Instituto Politécnico Nacional
Unidad Profesional Interdisciplinaria de Ingeniería campus Zacatecas

Área de ubicación para el desarrollo del trabajo
Ingeniería en Sistemas Computacionales

Línea de investigación
Inteligencia artificial

Título del proyecto de Trabajo Terminal
Reconocimiento de diagramas de flujo trazados a mano usando redes neuronales convolucionales.

Presenta(n):
Onder Francisco Campos García
David Betancourt Montellano

Director:
M. en C. Roberto Oswaldo Cruz Leija

Asesores:
M. en Ed. Karina Rodríguez Mejía

Zacatecas, Zacatecas a Noviembre de 2021.

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

Instituto Politécnico Nacional
"La Técnica al Servicio de la Patria"

85 Aniversario del Instituto Politécnico Nacional
70 Aniversario del CECyT 11 "Wilfrido Massieu"
60 Aniversario de la Escuela Superior de Física y Matemáticas
50 Aniversario del CECyT 12 "José Ma. Morelos" y del CECyT 13 "Ricardo Flores Magón"

Folio
UPIIZ/ESA/382/2021

Asunto
DESIGNACIÓN
ONDER FRANCISCO CAMPOS GARCÍA
INGENIERÍA EN SISTEMAS COMPUTACIONALES
BOLETA: 201767018198
GENERACIÓN: 2017-2020

Zacatecas, Zac., a 19 de octubre de 2021

C. ONDER FRANCISCO CAMPOS GARCÍA
PRESENTE

Mediante el presente se hace de su conocimiento que este Departamento acepta que **M. en C. Roberto Oswaldo Cruz Leija** sea **Director** y la **M. en ED. Karina Rodríguez Mejía** sea **Asesora** en el tema que propone usted a desarrollar como prueba escrita de la opción Curricular, con el título y contenido siguiente:

"Reconocimiento de diagramas de flujo trazados a mano usando redes neuronales convolucionales"

Se concede un plazo de máximo de un año, a partir de esta fecha, para presentarlo a revisión por el jurado asignado.

M. EN C. JULIA JANETH ROSALES MARES
Jefa del Departamento de Evaluación y Seguimiento Académico

SECRETARIA DE EDUCACIÓN PÚBLICA
INSTITUTO POLITÉCNICO NACIONAL
UNIDAD PROFESIONAL INTERDISCIPLINARIA
DE INGENIERÍA CAMPUS ZACATECAS
DIRECCIÓN

DR. FERNANDO FLORES MEJÍA
Director de la UPIIZ

México 2021 Año de la Independencia

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

Instituto Politécnico Nacional
"La Técnica al Servicio de la Patria"

85 Aniversario del Instituto Politécnico Nacional
70 Aniversario del CECyT 11 "Wilfrido Massieu"
60 Aniversario de la Escuela Superior de Física y Matemáticas
50 Aniversario del CECyT 12 "José Ma. Morelos" y del CECyT 13 "Ricardo Flores Magón"

Folio
UPIIZ/ESA/383/2021

Asunto
AUTORIZACIÓN DE IMPRESIÓN DE TRABAJO DE TITULACIÓN
ONDER FRANCISCO CAMPOS GARCÍA
INGENIERÍA EN SISTEMAS COMPUTACIONALES
BOLETA: 201767018198
GENERACIÓN: 2017-2020

Zacatecas, Zac., a 21 de octubre de 2021

El suscrito tengo el agrado de informar a usted, que habiendo procedido a revisar el trabajo de titulación que presenta con fines de titulación denominada:

"Reconocimiento de diagramas de flujo trazados a mano usando redes neuronales convolucionales"

Encontré que el citado **Trabajo de Titulación**, reúne los requisitos para **autorizar** la impresión y proceder a la presentación del Examen Profesional debiendo tomar en consideración las indicaciones y correcciones que al respecto se hicieron.

M. en C. Roberto Oswaldo Cruz Leija

M. en ED. Karina Rodríguez Mejía

Autorización de uso de obra

Instituto Politécnico Nacional
P r e s e n t e

Bajo protesta de decir verdad ***el*** que suscribe(n) ***Onder Francisco Campos García***, estudiante del programa de ***Ingeniería en Sistemas Computacionales***, con numero de boleta ***2017670181***, adscrito a la Unidad Profesional Interdisciplinaria de Ingeniería campus Zacatecas; manifiesto ser autor(a, as, es) y titular(es) de los derechos morales y patrimoniales de la obra titulada ***Reconocimiento de diagramas de flujo trazados a mano usando redes neuronales convolucionales.,*** en adelante el trabajo de titulación y de la cual se adjunta copia, por lo que por medio del presente y con fundamento en el artículo 27 fracción II, inciso b) de la Ley Federal del Derecho de Autor, otorgo a el Instituto Politécnico Nacional, en adelante El IPN, autorización no exclusiva para comunicar y exhibir públicamente total o parcialmente en medios digitales el trabajo de titulación por un periodo de ***INDEFINIDO*** contado a partir de la fecha de la presente autorización, dicho periodo se renovará automáticamente en caso de no dar aviso expreso a "El IPN" de su terminación.

En virtud de lo anterior, "El IPN" deberá reconocer en todo momento mi calidad de autor de el trabajo de Titulación.

Adicionalmente, y en mi calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación ,manifiesto que la misma es original y que la presente autorización no contraviene ninguna otorgada por el suscrito respecto del trabajo de titulación, por lo que deslindo de toda responsabilidad a El IPN en caso de que el contenido del trabajo de titulación o la autorización concedida afecte o viole derechos autorales, industriales, secretos industriales, convenios o contratos de confidencialidad o en general cualquier derecho de propiedad intelectual de terceros y asumo las consecuencias legales y económicas de cualquier demanda o reclamación que puedan derivarse del caso.

Zacatecas, Zac., a 3 de noviembre del 2021.

Atentamente

Nombre y firma del alumno

Firmas

En esta sección se mostrarán los nombres y las firmas de los alumnos responsables del desarrollo del proyecto de Trabajo Terminal.

C. Onder Francisco Campos García

C. David Betancourt Montellano

Autorización

Por medio del presente autorizo la impresión y distribución del presente reporte final de proyecto de Trabajo Terminal, toda vez que lo he leído, comprendido en su totalidad, y estoy de acuerdo con su contenido.

Atentamente:

M. en C. Roberto Oswaldo Cruz Leija

DIRECTOR

M. en Ed. Karina Rodríguez Mejía

ASESORA

Índices

Índice de contenido

Índice de tablas

Índice de figuras

Resumen

Actualmente la visión artificial se usa en un sin fin de tareas que van desde tareas domésticas hasta tareas industriales y educativas, ya que con ella se pueden agilizar estas tareas al tener un proceso automatizado. En este proyecto se aborda el problema de la recuperación de información trazada a mano, específicamente los diagramas de flujo usados en el área de programación y algoritmia por medio de visión artificial y se propone un pipeline capaz de reconocer los elementos de un diagrama de flujo trazado a mano usando redes neuronales convolucionales con los cuales generar código fuente en el lenguaje de programación C equivalente al diagrama reconocido, además de digitalizar el diagrama de flujo, esto es, reconstruir los elementos del diagrama de flujo reconocido y generar una imagen digital del mismo, automatizando así la dichas tareas, teniendo como resultado final un archivo con extensión .c con el código fuente y un imagen con formato .png con la digitalización del diagrama reconocido.

Palabras clave: Análisis de gramática, diagrama de flujo, procesamiento de imágenes, reconocimiento de patrones, reconocimiento de trazos, red neuronal convolucional.

Abstract

Currently, artificial vision is used in an endless number of tasks from domestic tasks to industrial and educational tasks since with it these tasks can be streamlined by having an automated process, this project explores the problem of handwritten information recovery, specifically the flowcharts used in the programming and algorithms area by means of artificial vision, and proposes a pipeline capable of recognizing the elements of an handwritten flowchart using convolutional neural networks with which to generate code source in the C programming language equivalent to the recognized diagram, in addition to digitizing the flow diagram, that is, reconstructing the elements of the recognized flow

diagram and generating a digital image of it, thus automating the various tasks, having as a final result a file with .c extension with the source code and an image in .png format with the digitization of the diagram recognized.

Keywords: Convolutional neural network, flowchart, grammar analysis, image processing , pattern recognition, sketches recognition.

Definición del problema

Contexto y antecedentes generales del problema

El reconocimiento de diagramas de flujo escritos a mano es una tarea que de forma general implica visión artificial para la detección y clasificación de sus elementos: texto, figuras y conectores, además de análisis de la gramática dado el contexto que los elementos tienen en el diagrama de flujo.

El reconocimiento es el problema central de aprender categorías y luego identificar nuevas instancias de esas categorías. En casi toda tarea de visión depende de la habilidad de reconocer objetos, escenas o categorías. “El reconocimiento visual en sí tiene una variedad de aplicaciones potenciales que tocan muchas áreas de inteligencia artificial y recuperación de información, que incluyen, por ejemplo, búsqueda de contenido en imágenes” [1].

Se pueden distinguir dos tipos de reconocimiento, en el primero se busca identificar instancias de un objeto, lugar o persona en particular, en el segundo se busca reconocer diferentes instancias de una categoría genérica como pertenecientes a la misma clase conceptual [1], la tarea de clasificación de diagramas de flujo cae en el primer tipo de reconocimiento ya que se busca identificar las figuras y conectores que el diagrama de flujo contiene de forma particular. Para realizar la tarea de reconocimiento es necesario implementar un clasificador el cual según [2] es un sistema capaz de dividir el espacio de características en las clases con las que se entrenan, para poder así asignar a un nuevo

objeto una clase a la que pertenece, de esta manera se tiene un sistema que como entrada tiene datos que describen a las clases que se van a clasificar y como salida se le da una etiqueta al objeto clasificado. Existen diversos tipos de clasificadores tales como mínima distancia, el K-NN, SVM, redes neuronales, etc.

Las redes neuronales convolucionales ya han dado lugar a una serie de avances en varios aspectos de la visión por computadora, incluida la clasificación de imágenes y detección de objetos [3]. Para una mejor eficacia en la clasificación de imágenes se han diseñado estructuras más complejas de redes neuronales convolucionales como la AlexNet [4] la cual usa una estructura de 8 capas que incluye cinco capas convolucionales y tres capas completamente conectadas, la VGG Net [5] utiliza núcleos de convolución del mismo tamaño, además usa de 16 a 19 capas en su red dos veces más que la AlexNet, otra estructura de redes de convolución es la GoogLeNet [6] la cual usa núcleos de convolución de diferentes tamaños y una arquitectura profunda de 22 capas, en la ResNet [7] lo que sorprende es su estructura de 152 capas.

El análisis de la gramática se ha usado por [8, 10, 11] para ayudar a la tarea de reconocimiento de diagrama de flujo trazados a mano y en [9] para reconocer diagramas de flujo en documentos de patentes, proponiendo posibles candidatos de figuras o conectores que se permiten en el flujo del diagrama y así poder considerar un diagrama válido, de esta forma se puede aumentar la eficacia en la tarea de reconocimiento.

Problema de investigación y situación problemática

El problema de investigación es sobre el reconocimiento de diagramas de flujo dibujados a mano, contemplando cada una de sus partes. Resolver tal problema es necesario para poder crear alguna aplicación como es el caso de generar el pseudocódigo [12] o código fuente [13] que vendría siendo la implementación de algún algoritmo, y por qué no, también generar el diagrama de flujo en formato digital, es decir, reconstruirlo en una imagen.

La necesidad de buscar métodos más eficaces en el reconocimiento del diagrama de flujo es relevante ya que el uso de diagramas de flujo es importante para matemáticos e informáticos. Los retos que se pueden encontrar son la adquisición de la información, por el ruido, niveles de brillo, niveles de exposición, superposición de los símbolos trazados y la forma de escribir de cada individuo [8].

Ahora bien, la situación problemática es la necesidad de automatizar la experimentación con la propuesta de solución para el problema de investigación; tal como se hizo en [8, 9, 10, 11, 12, 13] donde se automatizó con software la fase experimental en cada uno de los proyectos.

Estado del arte

A continuación, se muestra la investigación encontrada para el estado del arte considerando trabajos de investigación donde se han propuesto diversas soluciones con varias técnicas de reconocimiento de patrones para el problema en cuestión.

En [13] (*A Novel Pen-Based Flowchart Recognition System for Programming Teaching,* 2008) se trabajó en un sistema que provee al estudiante una interfaz intuitiva para el aprendizaje en el uso de los diagramas de flujo, donde se dibuja el diagrama en una tableta electrónica para posteriormente generar el código equivalente en el lenguaje de programación C, este sistema usa un modelo de reconocimiento de trazos híbrido: Support Vector Machine (SVM, por sus siglas en inglés) con Hidden Markov Model (HMM, por sus siglas en inglés), por el hecho de que HMM es bueno en el trato de datos secuenciales, y que con SVM se obtuvo un buen rendimiento en la clasificación de figuras, los resultados de la clasificación se encuentran en la tabla 1.

Symbol	SVM-HMM hybrid approach	Traditional HMM approach
operation	97.21	96.40
decision	96.30	95.27
I/O	96.31	94.89
connector	92.15	90.31
termination	93.28	91.44

Tabla 1. Eficacia del modelo híbrido SVM-HMM. Fuente: [13].

Un tiempo después en [8] (*Recognition of Hand Drawn Flowcharts*, 2013) el problema presentado fue el reconocimiento de diagramas de flujo dibujados a mano, aquí se propusieron dos enfoques para tratar el problema, *on-line* y *off-line*, el primer enfoque se refiere a la entrada interactiva de la información, es decir, que se hacen los trazos sobre un editor que contiene un lienzo para dibujar y así se va computando la entrada conforme se va recuperando la información, mientras que el *off-line* es cuando el diagrama de flujo ya fue hecho antes, siendo así que pueda estar en alguna fotografía o en papel, que significa que la información se captura en un solo paso. Se exponen algunas ventajas de cada enfoque, para el *on-line* no es necesario un preprocesamiento de la imagen, se pueden usar algoritmos de segmentación relativamente simples y habla de su relevancia para el contexto del proyecto dado que el número de dispositivos con pantalla sensibles al tacto fue en aumento; para el *off-line* es necesario el preprocesamiento para eliminar ruido y requiere de algoritmos de segmentación más complejos. Por otra parte, menciona que ambos enfoques pueden presentar problemas, por ejemplo, que las figuras estén sobrepuestas, que haya trazos imprecisos y la diversidad de los símbolos empleados [8], esto último quiere decir que puede o no usarse una simbología estándar. Por lo anterior, se piensa que debe invertirse tiempo suficiente para la etapa de preprocesamiento ya que no se desea que "entre basura y salga basura".

Como tal en [8] se implementó un sistema llamado "FlowChart Editor" (FCE) el cual trabaja con el enfoque *on-line* en una versión móvil y una versión de escritorio, permitiendo así almacenar los diagramas de flujo y guardarlos en un formato apropiado para usarlos en aplicaciones como Microsoft Word y Microsoft PowerPoint. Dicho sistema FCE trabaja con dos métodos para el reconocimiento de figuras y flechas, uno que pondera la similitud con

una figura modelo y otro basado en análisis probabilístico, alcanzando con este último la mayor eficacia (97.5%); la dirección y control del flujo se logra con análisis del contexto; para verificar la estructura del diagrama utiliza gramáticas gráficas que proporcionan un mecanismo en el cual la generación y transformación de objetos visuales se puede modelar con precisión de forma matemática. Finalmente, cabe decir que no se describen resultados de la implementación para el enfoque *off-line* y se menciona que para futuros trabajos se podrían juntar ambos enfoques en un solo sistema y además integrar un módulo para la generación automática de código fuente.

Otro problema atacado con el reconocimiento de diagramas de flujo es en [9] (*Flowchart recognition for non-textual information retrieval in patent search, 2013*) ya que propone una solución para la recuperación de información no textual en documentos de patentes, centrándose en los diagramas de flujo y permitiendo obtener una representación estructural del diagrama (archivo XML) que pueda ser consultada semánticamente. La solución al problema se construyó aplicando el estado del arte de las técnicas de reconocimiento de figuras y líneas; métodos de segmentación, descriptores de figuras de propósito general y una versión comercial de reconocimiento óptico de caracteres. A partir de los resultados se encontró que los métodos empleados son sensibles al ruido cuando los nodos se rompen ya sea por mala adquisición de la imagen o por causa del diseño. Por último, en las conclusiones menciona que por la evaluación se ha demostrado que el uso de descriptores dedicados para las tareas específicas de reconocer las formas posibles dentro del léxico del diagrama supera a los descriptores de forma genérica y que una investigación adicional sobre ellos podría mejorar el rendimiento de la propuesta de solución.

Sobre el proyecto anterior, probablemente si se hubiese hecho una etapa de preprocesamiento de imágenes adecuada se pudo haber mejorado el resultado. Por lo mismo que dice que los métodos empleados resultaron ser sensibles al ruido.

Un proyecto que propone algo diferente para el trabajo de reconocimiento es [10] (*Online recognition of sketched arrow-connected diagrams*, 2016). Aquí al igual que en otros el

problema consiste en el reconocimiento de trazos para diagramas hechos a mano bajo el enfoque *on-line*. El modelo que proponen es hacer el reconocimiento por medio de la selección de símbolos candidatos, basado en la evaluación de relaciones entre candidatos usando un conjunto de predicados, las partes centrales del modelo son la segmentación de texto/no texto, segmentación de símbolos, clasificación de símbolos y análisis de la estructura del diagrama. El canal de reconocimiento se muestra en la figura 1.

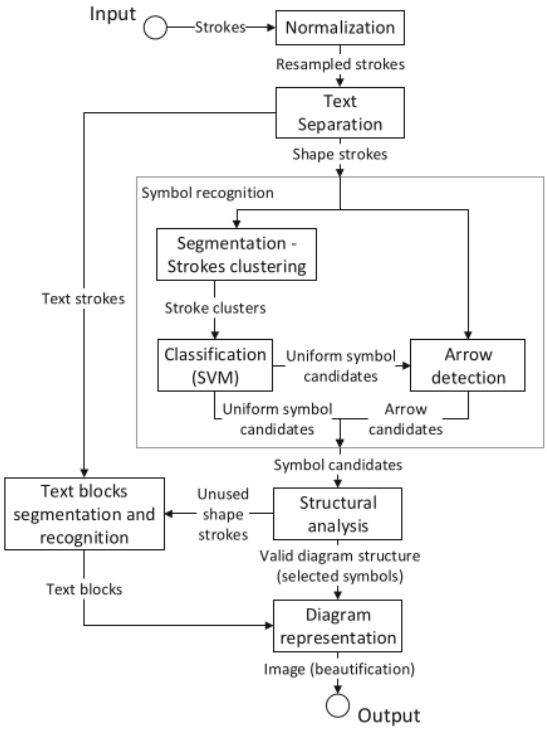


Figura 1. Canal de reconocimiento propuesto. Fuente: [10].

Para la clasificación entre texto y no texto se usaron dos redes neuronales del tipo Bidirectional Long Short Term Memory (BLSTM, por sus siglas en inglés) donde las etiquetas resultantes fueron texto y figuras, y para el lado de los caracteres se usó la base de datos IAMonDo [10] y de las figuras se usaron dos, un conjunto de datos previamente existente y uno creado por los autores para sus propios fines.

Para la separación del texto de las figuras describe dos procesos independientes, por un lado el reconocimiento del texto y por el otro el reconocimiento de la figura en sí. Para la

segmentación de las figuras se usó Single-Linkage Agglomerative Clustering (SLAC, por sus siglas en inglés) como clusterizador, una vez teniendo las figuras por separado procede la etapa de reconocimiento donde existen otras dos etapas:

1. Para figuras uniformes se usó un clasificador multiclase SVM.
2. Para las flechas (conectores) se llevan a cabo otras dos etapas:
    a. La primer etapa es encontrar el eje de una flecha que conecta dos figuras sin establecer dirección, posteriormente se busca la cabeza de la flecha para conocer la dirección de la misma.
    b. La segunda etapa es la del análisis estructural la cual es una tarea de optimización de puntajes de las relaciones entre los candidatos, estas relaciones se evalúan para dar un puntaje y se buscan las relaciones con el mayor puntaje posible.

Otro trabajo más de investigación que ataca solo el enfoque *on-line* es [11] (*Online flowchart understanding by combining max-margin Markov random field with grammatical analysis*, 2017) que usa el campo aleatorio de Markov de margen máximo con análisis gramatical. Los dos métodos de aprendizaje empleados son el Support Vector Machine (SVM, por sus siglas en inglés) y el Markov Random Field (MRF, por sus siglas en inglés). Menciona que su investigación supera a otras debido al procesamiento de símbolos en función de si tienen una apariencia regular, esto en específico en el campo del reconocimiento de diagramas de flujo. Por último, se menciona que para futuras investigaciones se explorará el potencial de MRF más profundamente para lograr mayor compatibilidad con la consistencia global del diagrama de flujo. Aquí la importancia del MRF parecer ser que éste es bueno etiquetando cuando existe una secuencia, en el caso del enfoque *on-line* es posible ir etiquetando conforme se va trazando el diagrama, cosa que no es posible en el enfoque *off-line* ya que cuando se captura en una imagen, el diagrama ya está completo.

Un trabajo reciente es [12] (*Flow2Code: from hand-drawn flowcharts to code execution,* 2017), ya que enfrenta el enfoque *off-line* desarrollando una aplicación móvil para el sistema operativo Android por medio de la cual se realiza el procesamiento para generar pseudocódigo y soportando la ejecución del mismo. En concreto, el objetivo que se persigue es que la curva de aprendizaje, para principiantes en el área de programación, disminuya, ya que cuando se está aprendiendo a programar el diseñar el diagrama en algún programa de aplicación implica aprender a usar tal programa. Uno de los principales obstáculos fue el reconocimiento de texto dentro de las figuras, se usó reconocimiento óptico de caracteres y menciona que podría mejorar con el tiempo la precisión si se opta por usar SVM o redes neuronales, sin embargo, esto necesitaría un entrenamiento para el reconocimiento del texto por parte de los usuarios antes de ver mejoras en la precisión, por lo que en el proyecto se integró un módulo para que el usuario corrigiera el texto y así garantizar una eficacia del 100% para el reconocimiento del texto, solo que esto hace más tardado el proceso de generar el pseudocódigo. Otra dificultad fue el brillo y grado de exposición de la imagen, ya que a veces no se lograban reconocer ciertas figuras cuando las fotos presentaban demasiado brillo o un grado de exposición elevado. Por otra parte, una deficiencia es que no toma en cuenta lo que sucedería si el usuario dibuja un diagrama de flujo incorrecto. Finalmente, menciona que algunas mejoras serían el ajuste dinámico de los umbrales utilizados en el reconocimiento en función de la forma de trazar de cada usuario, la posibilidad de digitalizar el diagrama, la predicción de trazos, situación que posiblemente mejoraría la experiencia de usuario y que sería deseable tener un sistema "full" que soporte ambos enfoques (*on-line y off-line*).

Considerando las investigaciones anteriores, es posible darse cuenta que el problema de reconocimiento de diagramas de flujo elaborados a mano tiene **dos enfoques**, *off-line* [8] (entrada no interactiva de la información) y *on-line* (entrada interactiva de la información), en la mayoría de los proyectos se trabajó el enfoque *on-line* ya que como se expone en [8] el enfoque *off-line* presenta más dificultades. Además, existe la necesidad por la orientación a educación como en [12], de obtener una eficacia global alta. En los trabajos mencionados

arriba se han intentado diversas técnicas del área de reconocimiento de patrones precisamente para la tarea de reconocer figuras, líneas y texto, como elementos del diagrama de flujo, por lo tanto, fue posible darse cuenta de que en ninguno de los proyectos se usaron redes neuronales convolucionales (CNNs por sus siglas en inglés) para el reconocimiento de los componentes trazados del diagrama de flujo, siendo lo anterior una oportunidad para explorar el uso del método antes mencionado.

A continuación, se hace una comparación de cada uno de los proyectos que fueron brevemente descritos arriba, en la tabla 2 se hace una relación de los proyectos y en la tabla 3 la comparación de los proyectos sobre varios aspectos.

ID	**Nombre del proyecto**
Pro1	*A Novel Pen-Based Flowchart Recognition System for Programming Teaching.*
Pro2	*Recognition of Hand Drawn Flowcharts.*
Pro3	*Flowchart recognition for non-textual information retrieval in patent search.*
Pro4	*Online recognition of sketched arrow-connected diagrams.*
Pro5	*Online flowchart understanding by combining max-margin Markov random field with grammatical analysis.*
Pro6	*Flow2Code: from hand-drawn flowcharts to code execution.*

Tabla 2. Relación de los proyectos de investigación de diagramas de flujo. Fuente: Elaboración propia.

ID	Enfoque	Método de reconocimiento		Máx. eficacia (%)	Diagrama digital	Generar código fuente	Análisis gramatical
		Figuras y conectores	Texto				
Pro1	*On-line*	SVM-HHM.	No menciona	95.05 (sin considerar texto)	No	Sí (C)	No
Pro2	*On-line* *Off-line*	Análisis probabilístico. Método de similitud.	No menciona	97.5 (sin considerar texto y solo *on-line*)	Sí	No	Sí
Pro3	*Off-line*	Clasificador de vecino más cercano. Descriptor basado en momentos geométricos. Descriptor del módulo de forma desenfocada.	Reconocimiento óptico de caracteres (OCR)	0.6054	No	Sí (XML)	Sí
Pro4	*On-line*	BLSTM NNs. SLAK. SVM.	BLSTM NNs	90.25	Sí	No	Sí
Pro5	*On-line*	SVM estructurado. MRF/M3N.	MRF	96.18	No	No	Sí
Pro6	*Off-line*	Axis Aligned Score. K-Vecino más cercano. Closed Shape Recognizer.	off-the-shelf OCR	95.00	No	Sí (pseudocódigo)	No

Tabla 3. Tabla comparativa de los proyectos que han abordado el problema de reconocimiento de diagramas de flujo. Fuente: Elaboración propia.

Como se puede comprobar en la tabla 3, tres de los seis proyectos llevados a cabo abordan el enfoque *on-line*, mientras que dos abordan el *off-line* [8], donde en Pro3 realmente se busca el reconocimiento de diagramas de flujo (no necesariamente del área de programación) en documentos de patentes y finalmente uno más, (Pro2) aborda ambos enfoques, solo que para el *off-line* [8] solo se propone un marco de trabajo mas no describe resultados de una implementación.

Para el método de reconocimiento de figuras y conectores trazados a mano se puede observar que para el enfoque *on-line* se opta en su mayoría por usar Support Vector Machine (SVM) y en general en ambos enfoques se suelen usar varias técnicas y algoritmos del área de reconocimiento de patrones, en la tabla solo se han puesto los que se han considerado principales en cada una de las propuestas de solución en los proyectos. Mientras que, para el reconocimiento del texto, en algunos se resuelve usando el OCR siendo una de las causas por las que Pro3 obtuviera una eficacia global baja, y es donde se piensa podrían proponerse, en un futuro, otros métodos de reconocimiento buscando una mejora sustancial y permita de esa manera aplicaciones más prácticas en el reconocimiento de diagramas de flujo trazados a mano, y poder evitar una etapa de corrección del texto como en [12].

En la misma tabla 3, es posible observar que solo en un proyecto (Pro1) se genera el código equivalente del diagrama de flujo en un lenguaje de programación y solo en dos proyectos se genera el diagrama de flujo en formato digital, pero ninguno, de los proyectos revisados, genera tanto código fuente así como diagrama digital al mismo tiempo.

Finalmente, es posible darse cuenta de la evolución que ha seguido el problema de investigación con base en el orden cronológico de los proyectos, esto es, que en primer lugar se ha abordado el enfoque *on-line* (Pro1) con una eficacia relativamente alta, luego se propuso un marco de trabajo para el *off-line* y no fue hasta Pro3 que se trabajó dicho

enfoque, no necesariamente orientado a los diagramas de flujo en programación, obteniendo una eficacia relativamente baja, y fue hasta Pro6 que gracias a una "buena" eficacia se pudo trabajar una aplicación móvil logrando vislumbrar una aplicación en el área educativa, sin embargo, el reconocimiento del texto parece ser algo que mejorar para mayor practicidad en una aplicación móvil por mencionar un ejemplo.

Descripción del proyecto

Mediante el uso de técnicas de procesamiento de imágenes, reconocimiento de patrones (redes neuronales convolucionales) y análisis de gramática se busca el reconocimiento de diagramas de flujo trazados a mano, enfrentando dicho problema bajo el enfoque *off-line* [8] encontrado así en el estado del arte.

Se diseñará e implementará un canal de reconocimiento (pipeline) destacando el uso de redes neuronales convolucionales (CNNs por sus siglas en inglés) para tratar de aumentar la precisión en el reconocimiento de figuras, conectores y texto, siendo éstos los componentes de un diagrama de flujo [14] y con ayuda del análisis de gramática se pretende verificar la estructura del diagrama de flujo con el fin de detectar errores en el mismo. Para poder ver el resultado de la propuesta en acción se desarrollará con el lenguaje de programación Python, mismo que recibirá de entrada una imagen digital del diagrama de flujo trazado a mano, hará el procesamiento de la imagen, el reconocimiento, análisis de gramática y finalmente regresará el resultado en una imagen del diagrama de flujo una vez que fue reconstruido en una imagen digital y su equivalente código fuente en C, esto si es que se ha logrado reconocer de forma global el diagrama de flujo con una estructura correcta, asimismo, es importante mencionar que los programas que se generen de pruebas seguirán el paradigma de programación estructurada y que se usará un conjunto definido de símbolos para la construcción del diagrama de flujo.

Finalmente, se deberá seleccionar un conjunto de datos, en este caso imágenes de los elementos, como son conectores y figuras, del diagrama de flujo, así como algunos diagramas de flujo para poder entrenar el modelo de detección que emplea una red neuronal convolucional y hacer pruebas con los programas de ejemplo que se usaron en [12], en este caso un simple “Hola mundo”, el cálculo del n-ésimo término de la sucesión de Fibonacci y el cálculo del factorial de un número entero dado. Por otra parte, se usará un modelo existente para la detección de texto, eligiendo uno que empareje el estado del arte y se adapte a nuestro problema en específico.

Objetivo general del proyecto

Diseñar e implementar un canal (pipeline) de reconocimiento usando técnicas de procesamiento de imágenes, redes neuronales convolucionales y análisis de gramática que sea capaz de reconocer completamente un diagrama de flujo trazado a mano.

Objetivos particulares del proyecto

- Hacer reconocimiento de figuras, texto y conectores plasmados en una imagen digital de un diagrama de flujo trazado a mano.
- Usar redes neuronales convolucionales (CNNs por sus siglas en inglés) para la tarea de detección (localización y clasificación).
- Usar análisis de gramática para verificar la estructura del diagrama de flujo.
- Poder generar código fuente de C a partir del diagrama de flujo.
- Poder digitalizar el diagrama de flujo, esto es, reconstruir los elementos del diagrama de flujo y generar una imagen digital.

Justificación

Aquí se menciona la principal motivación para llevar a cabo este proyecto, las ventajas y usos esperados de los resultados del presente proyecto de investigación.

La motivación general es dada por la importancia que tiene el uso de técnicas de inteligencia artificial para resolver problemas en diversos ámbitos [15]. Dado que se están viviendo tiempos interesantes, en los cuales el software está provocando una revolución, no solo económica, sino en todos los ámbitos de nuestra vida moderna. Cada vez son más y de mayor relevancia las empresas y sectores basados en software y que se comercializan como servicios online: desde películas, agricultura hasta defensa nacional [16].

La razón en concreto por la que se va a desarrollar el proyecto se soporta básicamente en investigar la aplicación de CNNs en la tarea de reconocimiento de los elementos de un diagrama de flujo trazado a mano bajo el enfoque *off-line* [8], ¿y por qué se piensa que es necesario investigar? Básicamente con la intención de buscar una mayor eficacia en el reconocimiento global del diagrama de flujo, ¿y por qué es necesario una mayor eficacia? Dado que asegurando una buena eficacia se podría aplicar la solución en un producto de software que ayude en el área de educación, como en [12], o áreas científicas. Por ejemplo, se sabe que hoy en día la mayoría de los programas de estudios de ingeniería requieren que los estudiantes sean capaces de programar en algún nivel [17]. Como representación visual del flujo de datos, los diagramas de flujo son útiles para diseñar un algoritmo y explicarlo a otros y/o permitir una efectiva colaboración. Así, se puede usar un diagrama de flujo para deletrear la lógica detrás de un programa antes de comenzar a codificarlo, asimismo a organizar el pensamiento general y proporcionar una guía a la hora de traducirlo a un lenguaje de programación. De esa forma, los estudiantes se enfocarán más en la resolución del problema antes que en la sintaxis de un determinado lenguaje de programación [18], reduciendo la curva de aprendizaje en etapas iniciales del aprendizaje de programación [12].

Las ventajas que aportaría este proyecto de investigación son:

- Propuesta de un método, que no se aplicó en ninguno de los proyectos explorados del estado del arte, para enfrentar el problema de reconocimiento de figuras, conectores y texto en un diagrama de flujo trazado a mano. Dicho método son las redes neuronales convolucionales.
- Los resultados obtenidos, del uso de CNNs, aportarán al conocimiento del campo de reconocimiento de trazos a mano con el fin de aplicarlo en otro tipo de diagramas, como pueden ser de química, electrónica digital, física, UML, entre otros [12].
- El usar análisis de gramática para comprobar la estructura del diagrama, con esto ya no se deja al aire la pregunta, ¿qué pasaría si el diagrama a reconocer es incorrecto?, en este caso determina si el diagrama de flujo es correcto o no, y en función de lo anterior poder generar el diagrama de flujo digital y su correspondiente código fuente en C, es una ventaja para quien desea ver el resultado de reconocer un diagrama de flujo en particular y no está seguro de que él mismo sea correcto.
- El hecho de que se podrá generar tanto el diagrama de flujo digital y su equivalente código fuente, lo anterior se propone para demostrar la utilidad de la solución al problema en cuestión, cabe mencionar que en los proyectos del estado del arte explorados sólo generaban el diagrama de flujo digital o bien el código fuente, pero no ambas cosas al mismo tiempo.
- El generar código fuente en lenguaje de programación C ya que con él se obtiene lo mejor de los lenguajes de alto nivel y de los lenguajes de bajo nivel proporcionando así facilidad para la comprensión a la hora de leer código, velocidad de ejecución alta y un tamaño de programa compacto [x, H. Chaudhary, The C programming language. First MIT createspace, 2014, p. 10.]

Como propuestas de trabajo futuro, respecto de [12] se pretende que la solución con redes neuronales convolucionales, en específico del texto, haga obsoleto el módulo de corrección del texto, logrando así que alguien en un futuro pueda hacer una aplicación móvil práctica y que permita al estudiante una realimentación rápida sobre el diseño de su algoritmo. Por

otra parte, que la solución se integre en un sistema completo que soporte ambos enfoques (*off-line* y *on-line*) ambos con una eficacia alta, dando más posibilidades de tener una mejor valoración en características cuantitativas y cualitativos de un sistema de software.

Hipótesis

Es posible reconocer los componentes (figuras, conectores y texto) de un diagrama de flujo bajo el enfoque *off-line* [8] usando redes neuronales convolucionales. Para aplicar el resultado del reconocimiento se busca generar código fuente en C y además digitalizar el diagrama de flujo.

Marco teórico

Algoritmo

Puesto que la razón de ser de los diagramas de flujo son el plasmar de forma gráfica los algoritmos es necesario entender qué son y qué características reúnen.

Un algoritmo es una manera de resolver un problema. Ejemplos de algoritmos son las recetas de cocina donde el problema que se resuelve es la preparación de un platillo.
En las recetas se especifica qué ingredientes se requieren, la forma en que se deben mezclar los ingredientes, así como tiempos de cocción para obtener una comida con buen sabor. Una definición más precisa de algoritmo es: “Un algoritmo es una secuencia precisa de pasos que nos permite alcanzar un resultado o resolver un problema” [14]. Ahora que se tiene presente lo que es un algoritmo se puede pensar en el uso de computadoras para solucionar problemas, esto involucra su codificación en un lenguaje de programación,

siendo éste un paso necesario como parte del proceso de hacer un programa computacional que resuelva el problema [19].

Las características que los algoritmos deben reunir son las siguientes:

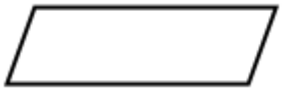
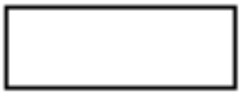
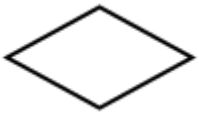
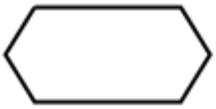
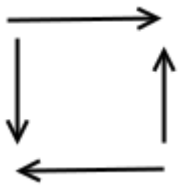
- **Precisión:** Los pasos a seguir en el algoritmo deben ser precisados claramente.
- **Determinismo:** El algoritmo, dado un conjunto de datos idénticos de entrada, siempre debe arrojar los mismos resultados.
- **Finitud:** El algoritmo, independientemente de la complejidad del mismo, siempre debe ser de longitud finita [14].

Diagrama de flujo

Se estudia de forma general los componentes de un diagrama de flujo, y su utilidad para la representación gráfica de los algoritmos, es necesario mencionar que el tipo de diagrama de flujo al que aquí se hace referencia es del área de programación y algoritmia.

Un diagrama de flujo representa la esquematización gráfica de un algoritmo. En realidad, muestra gráficamente los pasos o proceso a seguir para alcanzar la solución de un problema. Su correcta construcción es sumamente importante porque a partir del mismo se escribe un programa en algún lenguaje de programación. Si el diagrama de flujo está completo y correcto, el paso de éste a un lenguaje de programación es relativamente simple y directo.

Los símbolos presentados, colocados adecuadamente, permiten crear una estructura gráfica flexible que ilustra los pasos a seguir para alcanzar un resultado específico [14], en la siguiente tabla está cada uno de ellos que pertenecen a un conjunto en específico.

Representación gráfica	Descripción
	Símbolo utilizado para indicar el inicio y el fin del diagrama de flujo.
	Símbolo utilizado para lectura de los datos.
	Símbolo utilizado para representar un proceso. En su interior se expresan asignaciones, operaciones aritméticas, cambios de valor de celdas en memoria, etc.
	Símbolo utilizado para representar una decisión. En su interior se almacena una condición y dependiendo del resultado de la evaluación de la misma se sigue por una de las ramas o caminos alternativos.
	Símbolo utilizado para representar una decisión múltiple. En su interior se almacena un selector, y dependiendo del valor de dicho selector se sigue por una de las ramas o caminos alternativos. Este símbolo se utiliza en la estructura selectiva “Si” múltiple.
	Símbolo utilizado para representar la impresión de un resultado.
	Símbolos utilizados para expresar la dirección del flujo del diagrama (conectores).

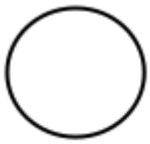
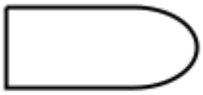
	Símbolo utilizado para expresar conexión dentro de una misma página.
	Símbolo utilizado para expresar conexión entre páginas diferentes.
	Símbolo utilizado para expresar un módulo de un problema. En realidad expresa que para continuar con el flujo normal del diagrama se debe primero resolver el subproblema que enuncia en su interior.

Tabla 4. Símbolos presentados en diagramas de flujo. Fuente: [14].

Reconocimiento de patrones

El área de reconocimiento de patrones es un área extensa de la cual se podría exponer mucha teoría y explicar diferentes algoritmos que existen para tareas de clasificación, sin embargo, solo se describen las partes teóricas, que se han considerado, más importantes para el presente proyecto, de esta forma se expone desde una perspectiva general hasta aterrizar en el uso de CNNs, mismas que representan el principal interés en el presente proyecto para la solución al problema de reconocimiento de diagramas de flujo trazados a mano.

El reconocimiento de patrones es una característica de todos los seres vivos, sin embargo, cada uno de ellos hace el reconocimiento de diferente manera con base a las necesidades que presenta, por ejemplo, para un perro no es capaz de reconocerse en un espejo ya que no le es útil esa capacidad.

Los seres humanos tienen la capacidad de identificar un rostro, entender las palabras que escuchan, leer caracteres escritos, identificar las llaves del carro, entre otras acciones que implican reconocer cosas con base a las experiencias pasadas, esta habilidad de los seres humanos que también puede ser adquirida por las computadoras se le llama reconocimiento de patrones el cual se basa en el problema universal de la clasificación y agrupamiento.

Definiendo un patrón como un conjunto de características o reglas que describen a un objeto, se puede entender el reconocimiento de patrones como el proceso de tomar datos de entrada con el propósito de extraer información que permita establecer relación entre conjuntos de dichos objetos y asignar a un objeto una clase predefinida.

Es natural para el humano que busque construir máquinas capaces de reconocer patrones, desde reconocimiento de voz, huellas digitales, caracteres, secuencia del ADN, entre otros. Está claro que el reconocimiento de patrones usando máquinas es útil y necesario, por el simple hecho de automatizar procesos o bien realizar tareas que al ser humano se le dificultan. Estas máquinas usan los datos para establecer las relaciones entre patrones para obtener un resultado, se pueden clasificar de la siguiente forma [2,22, 23, 25]:

Aprendizaje supervisado

Un aprendizaje supervisado requiere de un conjunto de patrones de los cuales se conoce la clase a la que pertenecen, a este grupo se le denomina conjunto de entrenamiento, un clasificador supervisado usa este conjunto para establecer relaciones entre las clases y así poder hacer una clasificación de un patrón dado [2,22].

Aprendizaje no supervisado

El aprendizaje no supervisado se realiza a partir de una agrupación de patrones de los cuales no se conoce la clase a la que pertenecen, con base a este conjunto se pretende

encontrar grupos de patrones que cumplen con características similares, a este proceso se le llama clusterización [2, 22].

Diseño de un sistema de reconocimiento de diagramas de flujo

El diseño de un sistema de reconocimiento de patrones consiste en clasificar un objeto dependiendo de sus características a partir de un algoritmo seleccionado. En seguida, se muestran los pasos que involucra el diseño de un sistema de clasificación y posteriormente se da una descripción de cada uno de ellos.

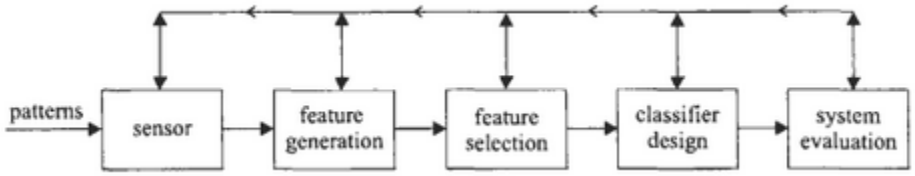


Figura 2. Diseño de un sistema computacional de clasificación. Fuente: [2].

- **Sensor:** Tiene como propósito extraer los elementos necesarios del medio para poder ser trabajados [2], por ejemplo, con la cámara del celular se puede tomar una fotografía a una persona para hacer reconocimiento facial.
- **La generación y selección de características:** Se refiere más que nada en cómo y cuáles características de los patrones se usarán para el proceso de clasificación y a su vez cuántos patrones se usarán como conjunto de entrenamiento, conviene que las características tengan una alta capacidad de discriminación entre clases.
- **Diseño de clasificación:** Se crea o selecciona un algoritmo de clasificación apropiado a la distribución de las clases, como ya se ha mencionado varias veces, en este proyecto se usarán CNNs.
- **Evaluación de sistema:** Se evalúa el rendimiento del sistema de clasificación, con preguntas como, ¿cuál es la tasa de error en la clasificación? [2].

Algoritmos de clasificación

Convolutional neural network

Las redes neuronales convolucionales (Convolutional Neural Networks, CNNs, ConvNet) es una clase de feed-forward neural network (red neuronal de avance, FFNN) no concurrente que está aplicada en análisis imágenes.

Una CNN es una red neuronal que generalmente está compuesta por las siguientes capas:

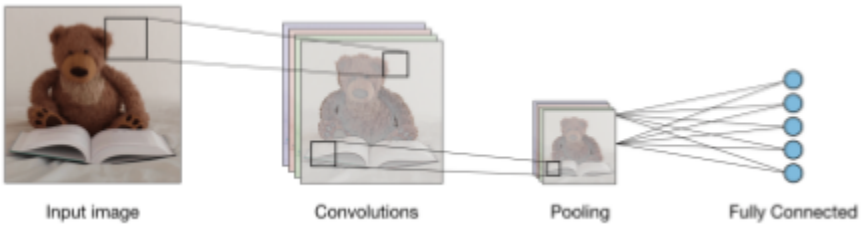


Figura 3. Capas de una CNN. Fuente: [20].

Convolutional layer (CONV): La capa de convoluciones usa filtros que usan operaciones de convolución ya que está escaneando la entrada con respecto a sus dimensiones. Sus hiperparámetros incluyen el tamaño del filtro o kernel y la cantidad de pixeles que avanza el filtro (paso). La salida resultante se denomina mapa de características o mapa de activación.

Pooling (POOL): La capa de agrupación (POOL) es una operación de disminución de resolución, generalmente aplicada después de una capa de convolución que hace invariancia espacial. En particular, el máximo y promedio son tipos especiales de agrupación donde se toma el valor máximo y media aritmética, respectivamente.

Fully Connected (FC): Esta capa opera en una entrada aplanada donde cada entrada está conectada a todas las neuronas. Las capas FC generalmente se encuentran hacia el final de arquitecturas CNN y se pueden utilizar para optimizar objetivos como los puntajes de confidencia de las clases.

Una red neuronal convolucional puede ser usada para clasificación de una imagen o clasificación y localización de objetos (detección de objetos) dentro de una imagen [20], por lo anterior es que se ha elegido esta arquitectura de redes neuronales para poder abordar el problema del reconocimiento de los elementos de un diagrama de flujo hecho a mano, siendo que el mismo se encuentra plasmado en una imagen.

Detección de objetos

Una de las aplicaciones de redes neuronales convolucionales es la detección de objetos, la detección de objetos consiste en localizar y clasificar un número variable de objetos de diferentes clases en una imagen. La detección de objetos es un problema más difícil que la localización de objetos por el número variable de salidas que se pueden tener. Existen varias técnicas para detección de objetos como la búsqueda selectiva, o la más simple como un deslizamiento de una ventana [21].

El reconocer los diagramas de flujo hechos a mano en una imagen es una tarea que se puede resolver con detección de objetos, se suele clasificar en dos tipos de modelos o detectores, el primer tipo es aquel que usa dos etapas para la detección, uno de los más representativos es Faster R-CNN y estos suelen tienen un alto rendimiento en la localización y clasificación de objetos; mientras que el otro tipo son los que emplean sólo una etapa para la detección, por ejemplo YOLO y SSD, su característica principal es que son más rápidos que los del primer tipo. Algo que tienen en común los detectores es que emplean lo que se denomina un backbone que actúa como el extractor básico de características que toma como entrada las imágenes y da de salida un mapa de características, es aquí donde se encuentran las capas de convolución y pooling [22].

Faster R-CNN

Este modelo de detección se basa en una versión más eficiente de sus modelos predecesores R-CNN y Fast R-CNN. Este modelo fue presentado en 2015 en un artículo llamado "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", su funcionamiento se basa en el uso de una red de propuestas de región (RPN, Region Proposal Network) que básicamente es una red neuronal convolucional (CNN) que en el artículo original usa de backbone una arquitectura llamada ZFNet o VGG, la salida de esa misma CNN es usada por el clasificador.

En la RPN se detectan regiones que podrían tener un objeto o no, y correspondientes coordenadas.

El clasificador o red de detección, emplea el mismo mapa de características y entonces las regiones propuestas se les aplica una técnica que se llama RoI pooling (con el fin de que cada región de interés tenga una entrada del mismo tamaño) y se conecta a una red fully-connected para así pasar la salida a un clasificador con soft-max y un método de regresión lineal (bbox regressor) para ajustar las coordenadas [23].

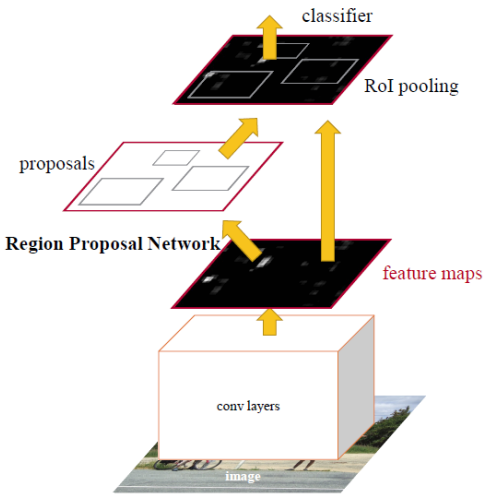


Figura 4. Modelo de detección de objetos Faster R-CNN. Fuente: [23]

Continual learning

Las redes neuronales han superado el rendimiento humano al realizar ciertas tareas como reconocimiento de voz, reconocimiento de objetos y videojuegos. Sin embargo, dicho éxito de las redes neuronales sigue siendo modesto en comparación con la inteligencia que el ser humano posee, la cual es capaz de aprender nuevas tareas y acumular conocimiento. El aprendizaje automático continuo tiene como objetivo un mayor nivel de inteligencia artificial al proporcionar a los agentes artificiales la capacidad de aprender en línea desde un flujo continuo e interminable de datos teniendo así la capacidad de integrar nuevos conocimientos y la estabilidad que retiene los conocimientos previos [24].

Procesamiento digital de imágenes

Para el presente proyecto es necesario abordar algunos tópicos del procesamiento de imágenes debido a que la entrada de información del canal de reconocimiento a desarrollar se encuentra en forma de imagen digital.

El procesamiento de imágenes digitales se refiere al procesamiento de imágenes en formato digital por medio de una computadora digital. Una imagen puede ser definida como una función bidimensional, $f(x, y)$, donde x & y son coordenadas espaciales (plano bidimensional), y la amplitud de f en cualquier par de coordenadas (x, y) es llamada la intensidad o nivel de gris de la imagen en ese punto. Cuando x, y y los valores de amplitud de f son todos finitos, se dice que la imagen es digital. Se debe tener en cuenta que una imagen digital está compuesta por un número finito de elementos, cada uno de los cuales tiene un valor y una ubicación particular. Estos elementos son referidos como píxeles. El píxel es el término más usado para denotar los elementos de una imagen digital [25].

Preprocesamiento

Pensando que es un paso realmente importante previo a la tarea de reconocimiento, se considera el preprocesamiento de una imagen como la preparación de datos que se refiere al proceso para transformar datos en bruto en un formato que puede ser usado para usarlo de manera efectiva en un modelo entrenado para el reconocimiento de patrones. Para una imagen, el preprocesamiento involucra por lo general una secuencia básica de transformaciones tales como recortar, filtrar, rotar, voltear [26], eliminar ruido, binarizar, adelgazar y vectorizar [8], en ese sentido, hay que probar alguna combinación de transformaciones con el fin de minimizar la pérdida de información y adecuar la imagen digital para ayudar la tarea de reconocimiento.

Segmentación

La segmentación es el proceso que subdivide una imagen en regiones uniformes. Cada región homogénea es una parte o un objeto de toda la imagen. En otras palabras, la segmentación de una imagen es definida por un conjunto de regiones que están conectadas y no se superponen, así cada píxel en un segmento de la imagen recibe una etiqueta de región única que indica la región a la que pertenece, por ejemplo, el método de umbralización donde una imagen en escala de grises se lleva a una escala con solo dos niveles, blanco o negro. La segmentación es uno de los pasos más importantes en el análisis de imágenes, debido a que los objetos u otras entidades de interés se extraen de una imagen para su posterior procesamiento, por ejemplo, la descripción y el reconocimiento [27]. Con esto se evitaría que a la entrada del modelo de predicción o clasificación se lleven componentes que juntos no resulta posible etiquetarlo.

Enmascaramiento de enfoque (unsharp masking)

Este algoritmo es empleado en la industria de impresión, básicamente es usada para realzar los bordes de una imagen digital; con esta técnica se puede obtener una mejor imagen de alto contraste, es por ello que también se conoce como un filtro de alto énfasis.

Matemáticamente hablando una versión de la imagen con filtro pasa-baja con ruido *g(m, n)* es restada de la imagen original *f(m,n)*, resultando así una imagen *v(m, n)* de alto contraste:

$$v(m, n) = f(m, n) - g(m, n)$$

Donde *m* y *n*, son dos enteros mayores a 0 que representan el número de píxeles de ancho y alto respectivamente.

Desde un punto de vista alternativo, un gradiente o filtro de señal pasa-alta podría ser sumado a la imagen original, así puede ser representado:
$v(m, n) = f(m, n) + alpha*h(m,n)$ donde $alpha > 0$ y $h(m, n)$ es un gradiente adecuadamente definido en (m, n) [27].

Esta técnica de procesamiento de imágenes es relevante para el presente proyecto, debido a que es requerido obtener la mayor cantidad de características de las imágenes con el fin de que el modelo de detección de objetos sea más eficaz, aún cuando la imagen pueda estar un tanto borrosa.

Gramática

Una de las formas en las que se pretende entender la estructura general de los diagramas de flujo es trabajando la semántica que es parte de la gramática que se encuentra en los diagramas de flujo.

Un lenguaje formal es un conjunto (finito o infinito) de cadenas finitas de símbolos primitivos, por ejemplo, el lenguaje “número” es simplemente el conjunto infinito de cadenas finitas formadas con los dígitos 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9. Dichas cadenas están formadas gracias a un alfabeto y a una **gramática** que están formalmente especificados. Se entiende por alfabeto a un conjunto no vacío de símbolos y entonces la gramática es un conjunto finito de reglas para formar cadenas finitas juntando símbolos del alfabeto. A cada

cadena de símbolos de un lenguaje formal se le llama fórmula bien formada (o palabra) del lenguaje [28].

Entendiendo lo anterior, se comprobará la estructura de los diagramas de flujo, con el fin de determinar si este es correcto o no.

Lenguaje de programación

Python

Se habla un poco de Python debido a que se ha elegido como el lenguaje de programación para poder llevar a cabo la investigación en este proyecto, en concreto para desarrollar un programa computacional en el cual se haga el preprocesamiento de la imagen, reconocimiento de los componentes del diagrama de flujo, análisis de gramática, generar el código fuente en C, la reconstrucción del diagrama de flujo en forma digital y diseño de la interfaz gráfica de usuario.

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python, http://www.python.org/, y puede distribuirse libremente. El mismo sitio contiene también distribuciones y enlaces de muchos módulos libres de Python de terceros, programas y herramientas, y documentación adicional [29].

El lenguaje de programación Python es ampliamente utilizado en ciencia debido a su gran cantidad de bibliotecas que proporcionan un conjunto completo de herramientas para análisis y manipulación de datos [30] tales como Numpy, Pandas, Keras, Sklearn y Matplotlib, además de la capacidad de interactuar con otros lenguajes de programación (C y Fortran) [30]. Esto último es la principal motivación de elegirlo como lenguaje de desarrollo ya que ese conjunto de bibliotecas proveen ya muchos algoritmos utilizados en el reconocimiento de patrones.

C

Como ya se mencionó antes, el código que se genere será en el lenguaje de programación C, es por ello que a continuación se define y se mencionan algunas de sus características.

C es us un lenguaje de programación de propósito general que ha sido usado para escribir compiladores o sistemas operativos así como programas en diversas disciplinas. Los tipos de datos fundamentales son caracteres, números enteros y números de punto flotante, además existe una jerarquía de tipos derivados, creados con apuntadores, arreglos, estructuras, y uniones. Las expresiones se forman a partir de operadores y operandos, cualquier expresión puede ser una proposición. C proporciona lo necesario para trabajar programación estructurada, al tener construcciones fundamentales de control de flujo, funciones y permitir que las variables sean declaradas en una modalidad estructurada [31]. Es por ello que se ha elegido este lenguaje, dado que se trabajará el paradigma de programación estructurada.

Marco metodológico

Con el fin de diseñar e implementar un pipeline de reconocimiento de diagramas de flujo trazadas a mano con redes neuronales convolucionales es prudente llevar este proceso bajo una metodología de desarrollo de software el cual ayude a estructurar, planear y controlar el desarrollo del mismo, para este proyecto se decidió usar la metodología en cascada, además en la parte de implementación de los modelos con redes neuronales convolucionales es necesario llevar una metodología de investigación que ayude a llevar el control de los ajustes de los parámetros de la red así como de los algoritmos de preprocesamiento de imágenes. Tal metodología de investigación es la metodología experimental ya que es usada en el campo de la ciencias de la computación [32].

Modelo en cascada

El modelo en cascada o modelo de la cascada, propone un enfoque sistemático y secuencial para el desarrollo del software: Especificación de requerimientos, planeación, modelado, construcción y por último despliegue [33], es relevante mencionar que en esta metodología una fase comienza cuando la anterior ha terminado [34].

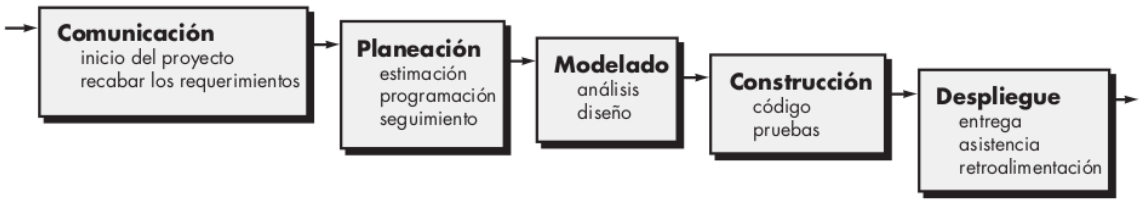


Figura 5. Metodología en cascada. Fuente: [33].

Productos de trabajo esperados

1. Comunicación.
    a. SRS.

2. Planeación.
    a. Marco metodológico.
    b. Cronograma.
    c. Matriz de riesgos.
    d. Recursos a utilizar.
    e. Productos de trabajo.
3. Modelado.
    a. Diagrama de arquitectura.
    b. Diagrama de pipeline.
    c. Diagrama de clases.
    d. Matriz de trazabilidad.
    e. Persistencia de datos.
        i. Dataset.
        ii. Resultados.
    f. Plan de pruebas de unitarias, de integración y de sistema.
4. Construcción.
    a. Diseño de la arquitectura de la CNN.
    b. Diagrama de flujo para preprocesamiento.
    c. Códigos:
        i. Preprocesamiento de imagen.
        ii. Segmentación.
        iii. Modelo de detección de figuras y conectores.
        iv. Modelo de detección de texto.
        v. Analizador de gramática.
        vi. Generador diagrama reconstruido en forma digital.
        vii. Generador código fuente.
        viii. Handler con GUI.
    d. Resultados de las pruebas.
    e. Reporte de la tarea de detección.

Metodología experimental

La metodología experimental muestra los experimentos que se realizarán para extraer resultados de implementaciones en el mundo real. Los experimentos pueden probar la veracidad de las teorías. Este método dentro de las ciencias computacionales se usa en varios campos diferentes como redes neuronales artificiales, automatización de pruebas de teoremas, procesamiento de lenguaje natural, análisis de desempeño y comportamientos, etc. Es importante reiterar que todos los experimentos y resultados deben ser reproducibles [35].

Para la experimentación, como un paso de realimentación, se diseña un sistema que tiene propiedades anticipadas, para después ser probadas experimentalmente. Si los resultados no coinciden con las expectativas, el diseño del sistema se modifica en consecuencia [36].

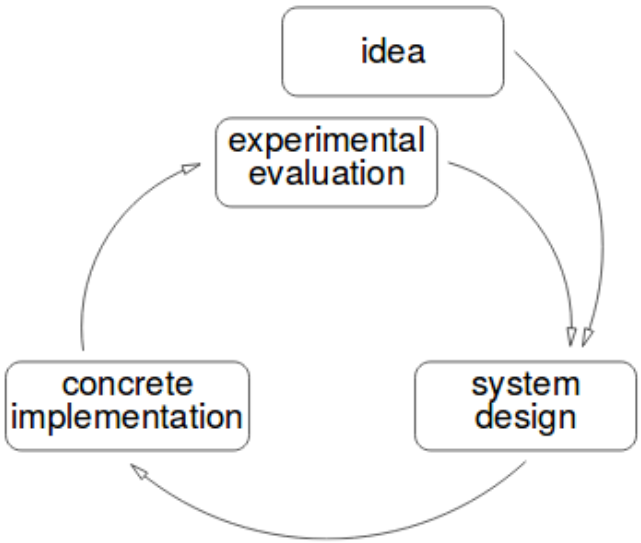


Figura 6. Experimentación en diseño de sistemas. Fuente: [36].

Productos de trabajo esperados

1. Idea.

a. Protocolo.
    i. Antecedentes y contexto del problema.
    ii. Definición del problema.
    iii. Estado del arte.
    iv. Justificación.
    v. Hipótesis.
    vi. Marco teórico.
2. Diseño del sistema.
    a. Diseño de la arquitectura de la CNN.
    b. Diagrama de flujo para preprocesamiento de imágenes.
3. Implementación en concreto.
    a. Software:
        i. Preprocesamiento de imagen.
        ii. Segmentación.
        iii. Modelo de detección de figuras y conectores.
        iv. Modelo de detección de texto.
        v. Analizador de gramática.
        vi. Generador diagrama reconstruido en forma digital.
        vii. Generador código fuente.
        viii. Handler con GUI.
        ix. Generador de reporte de evaluación del modelo.
4. Evaluación del experimento.
    a. Reporte de la tarea de detección de figuras y conectores:
        i. Mean Average Precision (mAP) en el dataset de validación.
        ii. Matriz de confusión.
        iii. Gráfico de la matriz de confusión.
        iv. Gráficas de la evolución de la pérdida en el entrenamiento.
        v. Pesos del modelo entrenado.

b. Pruebas del sistema con un conjunto de 56 imágenes de diagramas de flujo hechos a mano donde se incluyen los 3 algoritmos definidos para experimentar (imprimir “hola mundo”, cálculo del n-ésimo término de la sucesión de Fibonacci y cálculo del factorial).
c. Informe de resultados del experimento.

Ya que la metodología experimental se usará para la parte de desarrollo de la red neuronal convolucional y los algoritmos de preprocesamiento queda incrustado dentro de la metodología en cascada en la fase de **construcción**, en la siguiente figura se puede ver dicha fase en el rectángulo más grande.

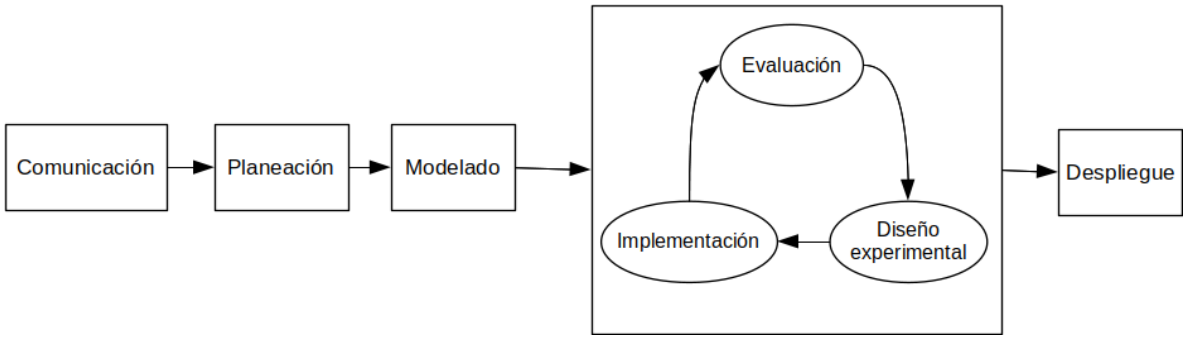


Figura 7. Incrustación de la metodología experimental en la metodología en cascada. Fuente: Elaboración propia.

Análisis y discusión de los resultados

Gestión del proyecto

Personas involucradas en el proyecto:

- Director: M. en C. Roberto Oswaldo Cruz Leija (ROCL).
- Asesora: M. en Ed. Karina Rodríguez Mejía (KRM).
- Desarrollador y tester: C. David Betancourt Montellano (DBM).
- Desarrollador y tester: C. Onder Francisco Campos García (OFCG).

1. Plan del proyecto

La forma de trabajo por cada uno de los desarrolladores se definió de la siguiente manera:

- Horas de trabajo por día: 4 horas.
- Días de trabajo: Lunes, martes, miércoles, jueves, viernes y sábado.
- Excepciones (días de descanso): Semana de Navidad, semana de Año Nuevo y la semana posterior a la Semana Santa en el mes de abril del año 2020.

A continuación se muestra el plan de actividades en dos tablas, lo planeado y lo real. Cada tabla contempla 6 columnas: Un identificador de la actividad, el nombre de la actividad, los responsables de llevarla a cabo (recurso), las horas, fecha de inicio y fecha final, que es cuando se comienza a hacer la actividad y cuando se concluye, respectivamente. Para la tabla de lo planeado se hizo una actualización a partir de la presentación de TT I, lo anterior con el fin de atender las observaciones realizadas en el reporte y presentación de TT I.

Plan de trabajo programado (planeado):

ID	Actividad	Recurso(s)	Horas	F. inicio	F. final
1	Protocolo	DBM, OFCG	216	05/08/2019	21/09/2019
1.1	Primera entrega protocolo	DBM, OFCG	136	05/08/2019	23/08/2019
1.2	Segunda entrega protocolo	DBM, OFCG	72	02/09/2019	11/09/2019
1.3	Correcciones finales	DBM, OFCG	32	17/09/2019	21/09/2019
2	Estructura del marco metodológico y plan de proyecto	DBM, OFCG	56	01/10/2019	08/10/2019
2.1	Marco metodológico	DBM, OFCG	8	01/10/2019	01/10/2019
2.2	Plan de proyecto (cronograma)	DBM, OFCG	48	02/10/2019	08/10/2019
3	SRS	DBM, OFCG	32	09/10/2019	12/10/2019
4	Observaciones marco metodológico	DBM, OFCG	24	14/10/2019	16/10/2019
5	Matriz de riesgos	DBM	16	17/10/2019	18/10/2019
6	Arquitectura de sistema	DBM	2	19/10/2019	19/10/2019
7	Diagrama conceptual	OFCG	6	19/10/2019	19/10/2019
8	Matriz de trazabilidad	DBM	8	19/10/2019	19/10/2019
9	Fase de estudio	DBM, OFCG	136	21/10/2019	09/12/2019
9.1	Metodología de investigación	DBM, OFCG	8	21/10/2019	21/10/2019
9.2	Procesamiento de imágenes	DBM, OFCG	24	04/11/2019	06/11/2019
10	Reporte de TT I	DBM, OFCG,	48	11/11/2019	16/11/2019

		ROCL, KRM			
11	Presentación final de TT I	DBM, OFCG, ROCL, KRM	64	18/11/2019	26/11/2019
9.3	Estudiar redes neuronales conv.	DBM, OFCG	104	27/11/2019	09/12/2019
12	Atender observaciones reporte TT I	DBM, OFCG, ROCL, KRM	64	10/11/2019	17/12/2019
13	Planeación	DBM, OFCG	64	18/12/2019	04/01/2020
13.1	Actualizar marco metodológico	DBM, OFCG	54	18/12/2019	04/01/2020
14	Modelado	DBM, OFCG	127	05/01/2020	05/02/2020
14.1	Actualizar matriz de riesgos	DBM	4	05/01/2020	05/01/2020
14.2	Listar productos de trabajo	DBM	4	06/01/2020	06/01/2020
14.3	Actualizar diagrama arquitectura	DBM	2	07/01/2020	07/01/2020
14.4	Diseñar pipeline	DBM, OFCG	6	07/01/2020	07/01/2020
14.5	Diseñar diagrama de persistencia de datos	OFCG	4	08/01/2020	08/01/2020
14.6	Actualizar matriz de trazabilidad	DBM	4	08/01/2020	08/01/2020
14.7	Diseñar diagrama de clases	DBM, OFCG	8	09/01/2020	09/01/2020

14.8	Diseñar plan de pruebas	DBM, OFCG	16	10/01/2020	11/01/2020
14.9	Construcción dataset	DBM, OFCG	79	12/01/2020	05/02/2020
14.9.1	Especificación	DBM, OFCG	8	12/01/2020	12/01/2020
14.9.2	Recuperación	DBM, OFCG, externos	41	13/01/2020	21/01/2020
14.9.3	Segmentación	DBM, OFCG, externos	30	22/01/2020	05/02/2020
15	Construcción	DBM, OFCG	720	13/01/2020	12/04/2020
15.1	Configuración de equipos	DBM, OFCG	64	13/01/2020	21/01/2020
15.2	Crear modelo de reconocimiento	DBM	60	22/01/2020	05/02/2020
15.3	Preprocesamiento	OFCG	16	22/01/2020	25/01/2020
15.4	Implementar modelo pre-entrenado de reconocimiento de texto	OFCG	44	26/01/2020	05/02/2020
15.5	Entrenar modelo de reconocimiento	DBM	40	06/02/2020	15/02/2020
15.6	Programar nodo	OFCG	4	06/02/2020	06/02/2020
15.7	Programar grafo	OFCG	12	07/02/2020	09/02/2020
15.8	Implementar analizador de gramática	OFCG	24	10/02/2020	15/02/2020
15.9	Evaluar modelo de reconocimiento de figuras	DBM	56	16/02/2020	29/02/2020
15.10	Programar generador de código	OFCG	56	16/02/2020	29/02/2020

15.11	Programar generador diagrama digital	DBM	20	01/03/2020	05/03/2020
15.12	Programar GUI principal	OFCG	20	01/03/2020	05/03/2020
15.13	Pruebas unitarias	OFCG, DBM	136	06/03/2020	22/03/2020
15.14	Integración	OFCG, DBM	72	23/03/2020	31/03/2020
15.15	Pruebas integración	OFCG, DBM	48	01/04/2020	06/04/2020
15.16	Pruebas de sistema	OFCG, DBM	48	07/04/2020	12/04/2020
15.17	Evaluación sistema	OFCG, DBM	48	13/04/2020	05/04/2020
15.18	Junta con asesores	DBM, OFCG, ROCL, KRM	4	20/04/2020	20/04/2020
15.19	Hacer informe de la evaluación	DBM, OFCG	40	21/04/2020	25/04/2020
16	Reporte final TT II	DBM, OFCG, ROCL, KRM	219	26/04/2020	20/05/2020
16.1	Hacerlo	DBM, OFCG	120	26/04/2020	10/05/2020
16.2	Revisarlo	DBM, OFCG, ROCL, KRM	80	10/05/2020	17/05/2020
16.3	Atender observaciones	DBM, OFCG	16	18/05/2020	19/05/2020
16.4	Imprimirlo	DBM, OFCG	3	20/05/2020	20/05/2020
17	Presentación TT II	DBM, OFCG,	48	21/05/2020	01/06/2020

		ROCL, KRM			

Tabla 5. Plan de proyecto (cronograma). Fuente: Elaboración propia.

El plan de trabajo en su versión final, contempla las modificaciones necesarias que se dieron durante el desarrollo del proyecto, además contiene el control sobre la ejecución de cada una de las actividades para observar las estimaciones exactas, sobreestimaciones y subestimaciones respecto a lo planeado.

Tanto para horas y fechas, se muestra lo planeado y lo real, con un color se marcan los datos reales para poder ver la desviación que existe en cuanto al tiempo y esfuerzo planeado. En los participantes, las iniciales marcadas en gris indican que la participación del recurso humano no se dio.

Las fechas y horas contadas marcadas en:

- Rojo significan que existe un retraso y subestimación, respectivamente.
- Verde quiere decir que se terminó o comenzó la actividad en la fecha planeada y para las horas que se realizó con el esfuerzo estimado.
- Amarillo, indica que se comenzó o terminó la actividad antes de lo planeado, o bien, para las horas que se empleó menos tiempo en horas respecto a lo estimado.

Las filas donde el texto está opacado representan actividades que no sea realizaron.

ID	Actividad	Recurso(s)	Horas	F. Inicio	F. Final
1	Protocolo	DBM, OFCG	216	05/08/2019	21/09/2019
1.1	Primera entrega protocolo	DBM, OFCG	136	05/08/2019	23/08/2019
1.2	Segunda entrega protocolo	DBM, OFCG	72	02/09/2019	11/09/2019
1.3	Correcciones finales	DBM, OFCG	32	17/09/2019	21/09/2019

2	Estructura del marco metodológico y plan de proyecto	DBM, OFCG	56 58	01/10/2019 01/10/2019	08/10/2019 11/10/2019
2.1	Marco metodológico	DBM, OFCG	8 4	01/10/2019 01/10/2019	01/10/2019 01/10/2019
2.2	Plan de proyecto (cronograma)	DBM, OFCG	48 54	02/10/2019 02/10/2019	08/10/2019 11/10/2019
3	SRS	DBM, OFCG	32 48	09/10/2019 12/10/2019	12/10/2019 21/10/2019
4	Observaciones marco metodológico	DBM	12 6	14/10/2019 16/10/2019	16/10/2019 17/10/2019
5	Matriz de riesgos	DBM	16 4	17/10/2019 18/10/2019	18/10/2019 22/10/2019
6	Arquitectura de sistema	DBM	2 2	19/10/2019 23/10/2019	19/10/2019 23/10/2019
7	Diagrama conceptual	OFCG	6 2	19/10/2019 19/10/2019	19/10/2019 19/10/2019
8	Matriz de trazabilidad	DBM	8 3	19/10/2019 23/10/2019	19/10/2019 23/10/2019
9	Reunión con director para validar cronograma y SRS	DBM, OFCG, ROCL	1 0.25	23/10/2019 25/10/2019	23/10/2019 25/10/2019
10	Reunión con asesor para validar cronograma y SRS	DBM, OFCG, KRM	1 0.283	23/10/2019 25/10/2019	23/10/2019 25/10/2019
11	Hacer acciones acordadas en junta de validación de SRS y cronograma con director	OFCG	0.1 0.166	25/10/2019 29/10/2019	25/10/2019 29/10/2019
12	Hacer acciones acordadas en junta de validación de SRS y cronograma con asesor	DBM	0.1 0.1	28/10/2019 29/10/2019	28/10/2019 29/10/2019
13	Reunión con asesor para	DBM,	1	29/10/2019	29/10/2019

	validar artefactos de diseño y análisis	OFCG, KRM	0.2	04/11/2019	04/11/2019
14	Reunión con director para validar artefactos de diseño y análisis	DBM, OFCG, ROCL	1 0.166	30/10/2019 04/11/2019	30/10/2019 04/11/2019
15	Hacer corrección de diagrama de arquitectura de sistema	DBM	1 1	05/11/2019 05/11/2019	05/11/2019 05/11/2019
16	Fase de estudio	DBM, OFCG	128 80.5	21/10/2019 21/10/2019	06/11/2019 26/01/2020
16.1	Metodología de investigación	DBM, OFCG	16 5	21/10/2019 21/10/2019	21/10/2019 21/10/2019
16.2	Procesamiento de imágenes	DBM, OFCG	24 26	22/10/2019 24/10/2019	24/10/2019 06/11/2019
17	Reporte de TT I	DBM, OFCG, ROCL, KRM	48 76	11/11/2019 05/11/2019	16/11/2019 19/11/2019
18	Presentación final de TT I	DBM OFCG, ROCL, KRM	64 17	18/11/2019 20/11/2019	26/11/2019 26/11/2019
16.3	Redes neuronales convolucionales	DBM, OFCG	80 49.5	27/11/2019 07/12/2019	07/12/2019 26/01/2020
19	Atender observaciones reporte TT I	DBM, OFCG, ROCL, KRM	64 7.6	10/12/2019 07/12/2019	17/12/2019 19/12/2019
20	Planeación	DBM, OFCG	64 23	18/12/2019 21/12/2019	04/01/2020 22/01/2019
20.1	Actualizar marco metodológico y plan de proyecto	DBM, OFCG	64 23	18/12/2019 21/12/2019	04/01/2020 22/01/2019
21	Modelado	DBM, OFCG	127 98.66	05/01/2020 21/01/2020	05/02/2020 14/02/2020
21.1	Actualizar matriz de riesgos	DBM	4 2	05/01/2020 21/01/2020	05/01/2020 21/01/2020
21.2	Listar productos de	DBM	4	06/01/2020	06/01/2020

	trabajo		1	21/01/2020	21/01/2020
21.3	Actualizar diagrama arquitectura	DBM	2 0.5	07/01/2020 22/01/2020	07/01/2020 22/01/2020
21.4	Diseñar pipeline	DBM, OFCG	6 1.5	07/01/2020 22/01/2020	07/01/2020 22/01/2020
21.5	Diseñar diagramas de persistencia de datos	OFCG	4 2	08/01/2020 22/01/2020	08/01/2020 22/01/2020
21.6	Actualizar matriz de trazabilidad	DBM	4 0.5	08/01/2020 27/01/2020	08/01/2020 27/01/2020
21.7	Diseñar diagrama de clases	DBM, OFCG	8 7.66	09/01/2020 22/01/2020	09/01/2020 27/01/2020
21.8	Diseñar plan de pruebas	DBM, OFCG	16 17.5	10/01/2020 26/01/2020	11/01/2020 01/02/2020
21.9	Construcción dataset	DBM, OFCG	79 66	12/01/2020 21/01/2020	05/02/2020 14/02/2020
21.9.1	Especificación	DBM, OFCG	8 6.5	12/01/2020 21/01/2020	12/01/2020 23/01/2020
21.9.2	Recuperación	DBM, OFCG, externos	41 24	13/01/2020 23/01/2020	21/01/2020 27/01/2020
21.9.3	Segmentación	DBM, OFCG, externos	30 35.5	22/01/2020 28/01/2020	05/02/2020 14/02/2020
22	Construcción	DBM, OFCG	720 1024.25	13/01/2020 20/01/2020	12/04/2020 27/06/2020
22.1	Configuración de equipos	DBM, OFCG	64 15	13/01/2020 20/01/2020	21/01/2020 26/01/2020
22.2	Crear modelo de reconocimiento	DBM	60 115.32	22/01/2020 01/02/2020	05/02/2020 14/03/2020
22.3	Preprocesamiento	OFCG	16 4	22/01/2020 25/01/2020	25/01/2020 25/01/2020
22.4	Implementar/personalizar modelo pre-entrenado de	OFCG	44 86.5	26/01/2020 04/02/2020	05/02/2020 23/04/2020

	reconocimiento de texto				
22.5	Entrenar modelo de reconocimiento	DBM	40 72.5	06/02/2020 14/02/2020	15/02/2020 28/03/2020
22.6	Programar nodo	OFCG	4 0.5	06/02/2020 26/02/2020	06/02/2020 26/02/2020
22.7	Programar grafo	OFCG	12 16.5	07/02/2020 26/02/2020	09/02/2020 24/03/2020
22.8	Implementar analizador de gramática	OFCG	24 20	10/02/2020 26/02/2020	15/02/2020 24/03/2020
22.9	Evaluar modelo de reconocimiento de figuras	DBM	56 53.83	16/02/2020 15/03/2020	29/02/2020 09/04/2020
22.10	Programar generador de código	OFCG	56 19.5	16/02/2020 25/03/2020	29/02/2020 06/04/2020
22.11	Programar generador diagrama digital	DBM	20 22	01/03/2020 10/04/2020	05/03/2020 14/04/2020
22.12	Programar GUI principal	OFCG	20 4	01/03/2020 15/03/2020	05/03/2020 18/03/2020
22.13	Pruebas unitarias	OFCG, DBM	136 52.33	06/03/2020 15/04/2020	22/03/2020 25/04/2020
22.14	Integración	OFCG, DBM	72 21	23/03/2020 20/04/2020	31/03/2020 29/04/2020
22.15	Pruebas integración	DBM	48 20.33	01/04/2020 25/04/2020	06/04/2020 12/05/2020
22.16	Pruebas de sistema	OFCG, DBM	48 9.5	07/04/2020 01/05/2020	12/04/2020 10/05/2020
22.17	Evaluación sistema	OFCG, DBM	48 5.66	13/04/2020 02/05/2020	05/04/2020 14/05/2020
15.18	Junta con asesores	DBM, OFCG, ROCL, KRM	4	20/04/2020	20/04/2020
22.18	Corregir modelo de reconocimiento de figuras.	DBM	10 21.83	02/05/2020330/ 04/2020	09/05/2020 09/06/2020

22.19	Reorganización del dataset para mejorar modelo de figuras	DBM	10 63.5	03/05/2020 03/05/2020	07/05/2020 27/05/2020
22.20	Corregir modelo de texto	OFCG, DBM	10 233.7	03/05/2020 03/05/2020	08/05/2020 20/06/2020
22.21	Entrenar modelo de reconocimiento de figuras	DBM	16 32.75	09/05/2020 17/05/2020	10/05/2020 10/06/2020
22.22	Preprocesamiento de la imagen	DBM	4 3	22/05/2020 22/05/2020	23/05/2020 24/05/2020
22.23	Evaluar modelo de reconocimiento de figuras	DBM	8 46.5	11/05/2020 19/05/2020	11/05/2020 13/06/2020
22.24	Corregir y adaptar generación de grafo	OFCG, DBM	8 5	05/06/2020 20/06/2020	05/06/2020 20/06/2020
22.25	Adaptar clasificador de figuras	DBM	2 7.5	06/06/2020 10/06/2020	06/06/2020 12/06/2020
22.26	Pruebas unitarias	OFCG, DBM	8 15	06/06/2020 13/06/2020	06/06/2020 25/05/2020
22.27	Integración	OFCG, DBM	4 23	07/06/2020 12/06/2020	07/06/2020 25/06/2020
22.28	Pruebas integración	OFCG, DBM	8 3	08/06/2020 25/06/2020	08/06/2020 25/06/2020
22.29	Pruebas de sistema	OFCG	4 10	09/06/2020 26/06/2020	09/06/2020 26/06/2020
22.30	Evaluación sistema v2	OFCG, DBM	10 10	09/06/2020 26/06/2020	10/06/2020 26/06/2020
22.31	Hacer informe de los resultados	DBM, OFCG	16 6	11/06/2020 27/06/2020	11/06/2020 27/06/2020
23	Reporte final TT II	DBM, OFCG, ROCL, KRM	216 +78	12/06/2020 14/06/2020	30/06/2020 30/06/2020
23.1	Hacerlo	DBM, OFCG	120 78	12/06/2020 14/06/2020	20/06/2020 29/06/2020

23.2	Revisarlo	DBM, OFCG, ROCL, KRM	50 --	21/06/2020 29/06/2020	28/06/2020 30/06/2020
23.3	Hacer video de funcionamiento del proyecto	DBM, OFCG	8 --	27/06/2020	27/06/2020 30/06/2020
23.4	Atender observaciones	DBM, OFCG	16	29/06/2020 --	29/06/2020 30/06/2020
16.4	Imprimirlo	DBM, OFCG	3	?/06/2020	?/06/2020
24	Presentación TT II	DBM, OFCG, ROCL, KRM	70	30/06/2020	08/06/2020
24.1	Hacerla	DBM, OFCG	32	30/06/2020	03/07/2020
24.2	Revisar presentación	DBM, OFCG, ROCL, KRM	16	04/07/2020	05/07/2020
24.3	Ensayar presentación	DBM, OFCG	20	05/07/2020	07/07/2020
24.4	Presentar	DBM, OFCG, ROCL, KRM	2	08/07/2020	08/07/2020

Tabla 6. Plan de proyecto en su versión final. Fuente: Elaboración propia.

Sobre el plan de proyecto en su versión final

A partir de la presentación de Trabajo Terminal I (actividad 18) se empleó una bitácora de actividades en donde cada día de trabajo se registraba lo que cada desarrollador y tester hacía, con el fin de llevar una medición del esfuerzo implicado en cada actividad, en el apéndice A (A.1 y A.2) se pueden encontrar gráficas derivadas del registro llevado a cabo.

También se puede apreciar en el plan de proyecto que para la fase de construcción (del modelo en cascada), se realizan dos versiones del sistema, razón por la cual en las evidencias de ejecución se hablará de versión inicial (propuesta), versión de primer

iteración (versión 1 del sistema) y versión 2 (final). Y finalmente que se ha trazado el control hasta la realización de este reporte.

Evidencias de ejecución

El seguimiento de las actividades se comenzó a realizar a partir de la actividad 2 (”Estructura del marco metodológico y plan de proyecto”). Enseguida se listan las actividades y descripción de lo realizado y/o referencia para encontrar evidencia:

- 2.1 - Marco metodológico: La evidencia es la investigación que se hizo sobre la metodología experimental y el modelo en cascada, información que se ha plasmado en la sección de “Marco metodológico”.
- 2.2 - Plan de proyecto (cronograma): Ver tablas 5 y 6.
- 3 - SRS (Software Requirements Specification): Ver el documento completo en el apéndice B.
- 4 - Observaciones del marco metodológico: Para atenderlas fue necesario actualizar el plan de proyecto, agregando las reuniones con director y asesor, y algunos artefactos de análisis y diseño que son requisito como parte de la materia de Trabajo Terminal I.
- 5 - Matriz de riesgos: Se puede consultar en el apéndice C (C.1).
- 6 - Arquitectura del sistema: La versión inicial se puede consultar en el apéndice D (D.1).
- 7 - Diagrama conceptual: Se puede ver más adelante, en el punto ”Desarrollo de proyecto”, específicamente, “Resumen del análisis del sistema” (Figura 8), mientras que la versión inicial se puede encontrar en el apéndice E (E.1).
- 8 - Matriz de trazabilidad: Ver el punto ”Desarrollo de proyecto”, específicamente, “Diseño del sistema” > “Matriz de trazabilidad” (Tabla 10).
- 9, 10, 11, 12, 13, 14, 15 - Reuniones para validar diagrama conceptual, SRS, cronograma, arquitectura del sistema, matriz de riesgos y matriz de trazabilidad: Ver apéndice F (F.1, F.2, F.3 y F.4) donde se encuentran las minutas de reuniones con asesor y director, donde se acordaron acciones y se les dio seguimiento.

- 16.1 - Metodología de investigación: Se volvieron a leer las referencias [36,37,38,39,40] de forma detenida.
- 16.2 - Procesamiento de imágenes: Se estudió con la sección 5 del curso "Deep Learning Computer VisionTM CNN, OpenCV, YOLO, SSD & GANs" que se compró en Udemy. Se puede revisar el certificado de finalización de curso en el apéndice G (G.1), además algunos códigos generados para esta actividad se encuentran en los siguientes enlaces de repositorios de GitHub: https://github.com/dbetm/learning-opencv (commits: 01/11/2019 y 04/11/2019), https://github.com/dbetm/processing-images/tree/master/opencv y https://github.com/Wolfteinter/Deep-Learning-Computer-Vision-CNN-OpenCV-YOLO-SSD-GANs.
- 16.3 - Redes neuronales convolucionales: Ver certificado de finalización del curso en el apéndice G (G.1), algunos proyectos realizados en dicho curso se encuentran en los siguientes enlaces: https://github.com/dbetm/david-ml-and-dl/tree/master/projects y https://github.com/Wolfteinter/Deep-Learning-Computer-Vision-CNN-OpenCV-YOLO-SSD-GANs. Además, se generó un documento donde se concentraron enlaces a recursos de distintos tipos para estudiar el tema de redes neuronales convolucionales, ver apéndice G (G.2).
- 17 - Reporte de TT I: Se encuentra en el siguiente enlace: https://bit.ly/316gCxm, además las reuniones con asesor y director donde se trató sobre la revisión del mismo, ver apéndice F (F.5 y F.6).
- 18 - Presentación final de TT I: Se hicieron diapositivas (https://bit.ly/317pva0) de apoyo para la presentación que se hizo de manera presencial.
- 19 - Atender observaciones del reporte TT I - Se concentraron las observaciones hechas del reporte final de TT I y se generó una versión corregida del mismo reporte con el fin de considerarse en este reporte final de TT II.
- 20.1 - Actualizar marco metodológico y plan de proyecto: Se cambió la 'metodología' de desarrollo de software que antes se había propuesto (*Iterativo e*

incremental) por el modelo en cascada, así como el plan de proyecto agregando las actividades faltantes de modelado. Además, la forma en cómo la metodología experimental de investigación encaja con el modelo en cascada, ver figura 7.

- 21.1 - Actualizar matriz de riesgos: Se puede consultar en el apéndice C (C.2).
- 21.2 - Listar productos de trabajo: Ver sección anterior, "Marco metodológico".
- 21.3 - Actualizar diagrama de arquitectura del sistema: La versión final se puede consultar en el apéndice D (D.2).
- 21.4 - Diseñar pipeline: La versión final se puede ver más adelante en "Diseño del sistema" > "Diseño detallado" (Figura 18) o bien, en el apéndice H (H.3), mientras que versiones anteriores se pueden consultar en el mismo apéndice H (H.1 y H.2).
- 21.5 - Diseñar diagramas de persistencia de datos: Ver apartado, "Diseño del sistema" > "Manejo de archivos" donde se encuentra la versión final o bien en el apéndice I (I.3), mientras que versiones anteriores se pueden consultar en el mismo apéndice I (I.1 y I.2).
- 21.6 - Actualizar matriz de trazabilidad: Lo que se hizo fue completarla, la misma está en "Diseño del sistema" > "Matriz de trazabilidad" (Tabla 10).
- 21.7 - Diseñar diagrama de clases: Está en "Diseño del sistema" > "Diseño detallado" (figuras 16 y 17), donde se encuentra la versión final o bien en el apéndice J (J.3), mientras que versiones anteriores se pueden consultar en el mismo apéndice J (J.1 y J.2).
- 21.8 - Diseñar plan de pruebas: Se puede consultar más adelante *Seguimiento al plan de pruebas* donde se muestra un resumen de los resultados de la ejecución de pruebas, en el anexo K (K.1, K.2, K.3) se pueden consultar completas los 3 tipos de pruebas (unitarias, de integración y de sistema).
- 21.9.1 - Especificación del dataset: Revisar el apartado "Diseño del sistema" > "Diseño de base de datos" además en el apéndice L (L.1) se muestra completa la especificación del dataset (conjunto de datos).
- 21.9.2 y 21.9.3 - Recuperación del dataset y segmentación de las imágenes del dataset: Se solicitó ayuda a alumnos de la escuela de otros semestres, gracias a ellos

se logró hacer los trazos a mano de figuras en hojas con 3 tipos de fondos (cuadrícula, raya y hoja de máquina). Con otro grupo de alumnos se hizo la segmentación en las imágenes, el dataset en su versión inicial se encuentra disponible para descargar en https://bit.ly/3doOxDR.

- 22.1 - Configuración de equipos: Se instaló un conjunto de bibliotecas de Python para manipulación de imágenes y de aprendizaje profundo (para implementar los modelos con redes neuronales), en el apéndice M (M.1) se encuentra una lista de paquetes necesarios para correr el sistema.
- 22.2, 22.3. 22.4, 22.5, 22.6, 22.7, 22.8, 22.9, 22.10, 22.11, 22.12 y 22.14 - Crear modelo de reconocimiento, preprocesamiento de la imagen, implementar/personalizar modelo pre-entrenado de reconocimiento de texto, entrenar modelo de reconocimiento, programar nodo, programar grafo, implementar analizador de gramática, evaluar modelo de reconocimiento de figuras, programar generador de código, programar generador diagrama digital, programar GUI principal (handler) e integración de los módulos: Los códigos generados para las anteriores actividades de codificación (de la primer versión completa) se pueden encontrar en el repositorio público de GitHub del proyecto: https://github.com/dbetm/handwritten-flowchart-with-cnn/tree/v1.0**,** el resultado de la evaluación de los modelos de detección de figuras y conectores, para una primer versión, se encuentra en el apéndice N (N.1).
- 22.13, 22.15 y 22.16 - Pruebas unitarias, de integración y de sistema: Se puede consultar más adelante “Seguimiento al plan de pruebas” donde se muestra un resumen de los resultados de la ejecución de pruebas, en el anexo K (K.1, K2 y K.3) se pueden consultar completas los 3 tipos de pruebas (unitarias, de integración y de sistema).
- 22.17 - Evaluación del sistema: Se evalúa con el fin de determinar qué tan bien resuelve el problema de la detección de los componentes del diagrama de flujo, ver evaluación para la versión 1 del sistema en el apéndice O (O.1).

- 22.18, 22.20, 22.21, 22.22, 22.23, 22.24, 22.25 y 22.27 - Corregir modelo de reconocimiento de figuras, corregir modelo de texto, entrenar modelo de reconocimiento de figuras, preprocesamiento de la imagen, evaluar modelo de reconocimiento de figuras, corregir y adaptar generación de grafo, adaptar clasificador de figuras e integración de los módulos: Los códigos generados para las anteriores actividades de codificación (versión final) se pueden encontrar en el repositorio público de GitHub del proyecto: https://github.com/dbetm/handwritten-flowchart-with-cnn, el resultado de la evaluación de los modelos de detección de figuras y conectores se encuentra en el apéndice N (N.2).
- 22.19 - Reorganización del dataset para mejorar modelo de figuras: Se pidió ayuda a personas externas para dibujar diagramas de flujo, a su vez que se dibujaron más plantillas de figuras y conectores para balancear la cantidad de instancias por cada clase de símbolo, la versión final se puede descargar aquí https://n9.cl/6kgs la estructura del dataset final es la que se encuentra en el apartado "Diseño del sistema" > "Diseño de base de datos".
- 22.26, 22.28 y 22.29 - Pruebas unitarias, de integración y de sistema: Se puede consultar más adelante "Seguimiento al plan de pruebas" donde se muestra un resumen de los resultados de la ejecución de pruebas, además, en el anexo K (K.1, K2 y K.3) se pueden consultar completas los 3 tipos de pruebas (unitarias, de integración y de sistema).
- 22.30 - Evaluación sistema v2: Se evalúa con el fin de determinar qué tan bien resuelve el problema, ver evaluación para la versión 2 (final) del sistema en el apéndice O (O.2).
- 22.31 - Hacer informe de los resultados: Se puede consultar en el apéndice P (P.1), donde se muestran resultados de los modelos de redes neuronales empleados y su efectividad resolviendo el problema, con el fin de determinar el cumplimiento de la hipótesis.
- 23.1 - Hacer reporte final de TT II: La evidencia es este mismo documento.

- 23.2 y 23.4 - Revisión del reporte final de TT II y atender observaciones: La evidencia es que se han generado correcciones y ha sido aceptado (firmas) por los asesores.
- 23.3 - Hacer video de funcionamiento del proyecto: El video se hará al finalizar el presente reporte.
- 24 - Presentación de TT II: La presentación se realizará después de la entrega de este reporte.

2. Manejo de desviaciones en la ejecución del plan

Se evidencia que existieron desviaciones en la ejecución del plan de proyecto al comparar entre sí las tablas de los cronogramas, en su versión planeada y versión final, tabla 5 y tabla 6 respectivamente. A continuación se puntualizan las desviaciones más relevantes:

- Se realizaron juntas con director de proyecto y asesora de proyecto para validar SRS, cronograma, diagrama conceptual, arquitectura del sistema, matriz de riesgos y matriz de trazabilidad, estas juntas y acciones derivadas corresponden a las actividades 9, 10, 11, 12, 13 y 15 del plan de proyecto en su versión final, de igual forma se pueden consultar las minutas de las juntas en el apéndice F (F.1, F.2, F.3 y F.4).
- Las observaciones del marco metodológico (en la versión final del plan de proyecto es la actividad 4) se atendieron solo por DBM, cuando lo planeado era que se hiciera por DBM y OFCG, esto fue debido al retraso considerable que se tuvo al hacer el SRS, de esta manera OFCG podría terminar algunas partes del SRS mientras que DBM comenzaba a atender las observaciones del marco metodológico.
- Cambió orden de las subactividades en la fase de estudio, ya que primero se estudió el procesamiento de imágenes antes que redes neuronales convolucionales, esta modificación se generó desde la materia de Trabajo Terminal I donde al abordar el

curso “Deep Learning Computer VisionTM CNN, OpenCV, YOLO, SSD & GANs” fue posible darse cuenta que era deseable primero reforzar y aprender nuevas técnicas de procesamiento de imágenes que preparan la imagen para la tarea de reconocimiento, y es por eso que se actualizaron los cronogramas quedando el estudio de redes neuronales después de la presentación de TT I. Se movió en ese periodo de tiempo (27/11/2019 - 07/12/2019), ya que justamente, en la planeación inicial, se había acordado dejar un espacio como comodín por si llegaba a ocurrir algún imprevisto. Sobre esta misma actividad de estudio de redes neuronales convolucionales es posible observar (en la versión final del plan de proyecto es la actividad 16.3) un retraso considerable respecto de la fecha de comienzo y terminación, esto es debido a asuntos personales de los desarrolladores y que se atendieron al mismo tiempo las observaciones del reporte de TT I.

- Debido a que se cambió el marco metodológico, fue necesario actualizarlo (en la versión final del plan de proyecto es la actividad 20.1), esta actividad tuvo un retraso considerable ya que se llegó a un punto de incertidumbre sobre la metodología de desarrollo de software a utilizar y no fue hasta que se habló con asesores para consensuar usar el modelo en cascada (sugerencia de uno de los revisores), ver sección anterior “Marco metodológico”.
- Las actividades de la fase de modelado (en la versión final del plan de proyecto las actividades 21.x) y de la fase de construcción (en la versión final del plan de proyecto las actividades 22.x) se comenzaron con significativo retraso respecto a lo planeado (ver versión final del plan de proyecto para constatar). Para manejar tal retraso se estuvo trabajando los días de descanso (los domingos y días de la Semana Santa), esto se puede ver en el apéndice A (A.1).
- Se llevó más tiempo (de forma significativa) de esfuerzo planeado el crear el modelo de reconocimiento de figuras (en la versión final del plan de proyecto la actividad 22.2), la implementación del modelo de texto (en la versión final del plan de proyecto la actividad 22.4) y el entrenamiento del modelo de reconocimiento de figuras (en la versión final del plan de proyecto la actividad 22.5), para manejar tal

retraso se estuvo trabajando los días de descanso (los domingos y días de la Semana Santa), esto se refleja las gráficas en el apéndice A (A.1 y A.2).

- No se llevó a cabo la junta con asesores (en la versión planeada del cronograma es la actividad 15.18) debido a que el objetivo era informar de los resultados de la evaluación de la primer versión del sistema y determinar si hacer otro ciclo o no, pero no fue necesario hacer dicho análisis en conjunto con los asesores, ya que los resultados para la primer versión no fueron aceptables, ver apéndice O (O.1) donde está la evaluación del sistema para la primer versión.
- Dado que la primer versión del sistema no obtuvo resultados aceptables resolviendo el problema, se realizó otra iteración de la fase de construcción, permitiendo esto por la naturaleza de la metodología experimental que define ciclos hasta que se resuelve el problema; por lo tanto, se agregaron nuevas actividades, que son el mejorar el dataset, detección del modelo de figuras, modelo de texto, preprocesamiento de imagen, adaptar el grafo, adaptar el clasificador, la segunda versión de las pruebas y finalmente se volvió a evaluar el sistema (en la versión final del plan de proyecto las actividades 22.18, 22.19, 22.20, 22.21, 22.22, 22.23, 22.24, 22.25, 22.26, 22.27, 22.28, 22,29 y 22.30). Para avanzar más rápido se optó por usar una plataforma (*Colab* de *Google*) en la nube para realizar el entrenamiento del modelo de detección de figuras (ya que debido al confinamiento por la pandemia por el *COVID-19* el usar la computadora remotamente era mucho más tardado, por la mala calidad de conexión de internet, que hacerlo en *Colab*), para ello se generó un script para realizar los entrenamientos que puede ser encontrado aquí: https://github.com/dbetm/handwritten-flowchart-with-cnn/blob/master/model/training_f_rcnn_in_colab.ipynb. Por otra parte, para corregir el modelo de texto, se tuvo que modificar el modelo, hacer entrenamientos del mismo y agregar un módulo de aprobación del reconocimiento de texto y “continual learning”, esto se describe en la evaluación de la segunda versión del sistema, ver apéndice O (O.2).

- El reporte final de TT II ya no se imprime debido al confinamiento por la pandemia por el *COVID-19* que impidió tanto la entrega en físico de este reporte así como la presentación presencial de TT II. Además, se solicita realizar un video para mostrar el funcionamiento del sistema (en la versión final del plan de proyecto la actividad 23.3).

3. Plan de los riesgos del proyecto.

El documento de matriz de riesgos se puede consultar en el apéndice C.

En la siguiente tabla se muestran los riesgos que se presentaron durante el desarrollo del proyecto, la forma en que se mitigan y el origen de los mismos.

Riesgo	**Mitigación**	**Causa(s)**
Agotar el tiempo.	Se ha trabajado los días de descanso (domingos y vacaciones de Semana Santa), además algunos días se trabajó más del tiempo planeado de 4 horas.	Durante el periodo de TT I fue debido a la carga de trabajo de otra materias. Y mayormente por desviaciones en el plan de proyecto. Cabe destacar que la contingencia por la pandemia generó que se tuvieran que buscar alternativas para el entrenamiento del modelo de figuras.
El nivel de conocimientos	Investigar diversas técnicas	La poca o nula experiencia

de los involucrados no son suficientes para desarrollar el proyecto.	y algoritmos durante la fase de construcción. Preguntar en grupo de expertos, una duda sobre CNNs.	en algunos aspectos que tocó abordar
Se trunca el entrenamiento de un modelo.	Se programó para que en cada iteración del entrenamiento se guardará el mejor modelo en la unidad de almacenamiento secundario o en la nube. Y derivar una versión del script de entrenamiento para modificar parámetros que permitieran retomar el entrenamiento.	En ocasiones la GPU se quedó sin memoria principal, generando un error que impedía continuar el entrenamiento. En la nube (Colab), el servicio gratuito solo permite alrededor de 10 horas continuas de uso cada 24 horas, así que cuando se excedía se interrumpía la ejecución.
Dataset no adecuado	Agregar más imágenes, en este caso de diagramas de flujo. Y modificar el algoritmo para hacer aumentó de datos.	En el dataset las figuras y conectores tenían un contexto de estar aisladas, sin intersectar con ninguna otra figura. Había fotos donde sobre las figuras y conectores se apreciaba el “recalcado” de otro tipo de figura, induciendo esto confusión al modelo.

Sobreajuste del modelo de texto	No usar dicho modelo sobreajustado y modificar la arquitectura para evitar futuros modelos sobreajustados.	Sobreentrenamiento.
Baja eficacia de los modelos para generar código fuente y diagrama digital.	Generar una segunda iteración de la fase de construcción.	Mala calidad del dataset de figuras. El modelo de texto no reconocía caracteres especiales.

Tabla 7. Riesgos que se han presentado durante el proyecto. Fuente: Elaboración propia.

Sobre el truncamiento del entrenamiento de un modelo, en la matriz de riesgos se había considerado que solo un fallo eléctrico lo podía ocasionar, pero la realidad fue que se encontraron dos causas más, por lo que queda de aprendizaje conocer las otras 2 posibilidades.

Desarrollo del proyecto

1. Resumen del análisis del sistema

Diagrama conceptual

El siguiente diagrama conceptual del software a desarrollar permite visualizar las relaciones entre las áreas, ideas y conceptos que implica el software de la etapa experimental en el contexto de la metodología experimental y de la fase de construcción del modelo en cascada.

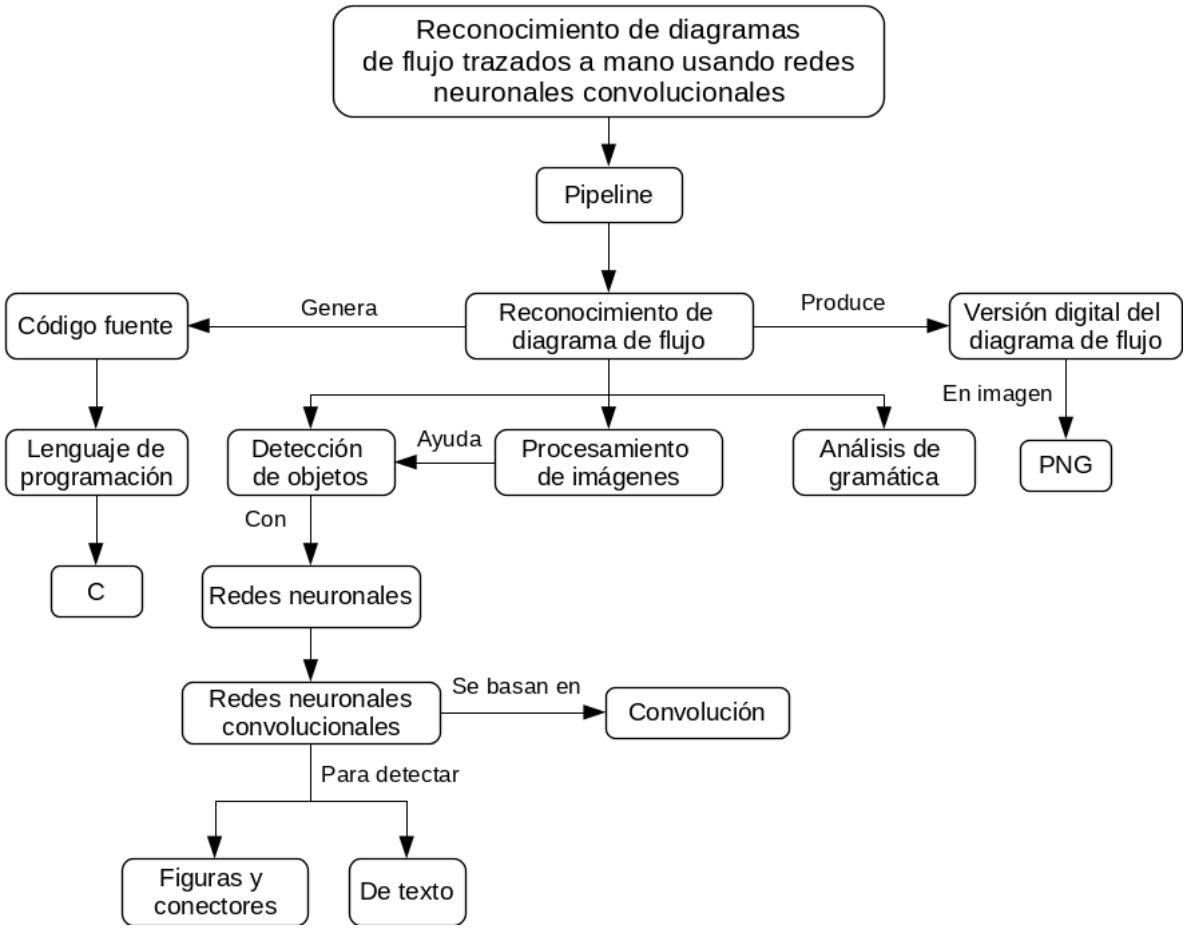


Figura 8. Diagrama conceptual del sistema para resolver el problema de reconocimiento de diagramas de flujo trazados a mano. Fuente: Elaboración propia.

Requerimientos funcionales

Aquí se muestran los requerimientos establecidos, el documento completo de especificación de requerimientos de software se encuentra en el apéndice B (B.1).

Los requerimientos se encuentran en dos tablas, en la primera se define un identificador para cada uno, un nombre, una descripción y los requerimientos funcionales relacionados. Para la segunda tabla, se modela el flujo de información que ocurre en cada uno de los requerimientos funcionales, es por ello que se define una entrada de datos, un proceso y una salida.

Identificador	**Nombre**	**Descripción**	**Requerimientos relacionados**
RF1	Cargar imagen	El sistema debe de cargar la imagen del diagrama de flujo del sistema de archivos de la computadora.	
RF2	Preprocesamiento de la imagen	El sistema debe de hacer un preprocesamiento de la imagen usando técnicas de procesamiento de imágenes.	RF1
RF3	Segmentación figura / texto	El sistema debe de ser capaz de segmentar entre lo que es una figura o conector y texto ya que cada uno se procesa de forma diferente.	RF2
RF4	Entrenar modelo de figuras	El sistema debe tener un modelo capaz de aprender de manera supervisada para diferenciar las símbolos a partir de un conjunto de imágenes proporcionadas.	
RF5	Clasificación de figuras y conectores	El sistema debe clasificar que tipo de figura o conector usando redes neuronales convolucionales.	RF3
RF6	Clasificación de texto	El sistema debe clasificar los caracteres del texto segmentados	RF3

		usando redes neuronales convolucionales.	
RF7	Análisis de gramática	El sistema debe usar análisis de gramática para verificar la estructura global del diagrama de flujo.	RF5
RF8	Generar versión digital del diagrama de flujo	El sistema debe de generar una versión digital del diagrama de flujo a partir de la clasificación de figuras, conectores y texto de la imagen.	RF1, RF2, RF3, RF5, RF6, RF7
RF9	Generar código fuente en C equivalente	El sistema debe de generar el código fuente en C equivalente al diagrama de flujo a partir de la clasificación de de figuras, conectores y texto de la imagen.	RF1, RF2, RF3, RF5, RF6, RF7
RF10	Compilar código fuente	Para probar que el código generado funciona, se hace una llamada al sistema operativo para compilar.	RF9
RF11	GUI de control	Poder reconocer diagramas de flujo trazados a mano así como hacer el entrenamiento del modelo de figuras, mediante ventanas, botones y cajas de texto.	RF3, RF4

Tabla 8. Requerimientos funcionales establecidos. Fuente: Elaboración propia.

Nota sobre RF3 - Segmentación figura / texto: Como tal no hay un algoritmo que haga segmentación a nivel de imagen, esto se hace simbólicamente al mandar la misma imagen a dos modelos diferentes, en este caso el modelo de reconocimiento de texto y el de figuras y conectores.

Flujo de información

Requerimiento	Entrada	Proceso	Salida
RF1	Imagen	1. Seleccionar archivo. 2. Leer archivo como objeto. 3. Seleccionar espacio de colores.	Objeto
RF2	Imagen	1. Seleccionar algoritmo(s) de preprocesamiento. 2. Aplicar algoritmo(s) de preprocesamiento a la imagen.	Imagen preprocesada
RF3	Imagen	1. Aplicar algoritmo(s) de segmentación a la imagen.	Imagen segmentada.
RF4	Conjunto de imágenes	1. Cargar el conjunto de imágenes. 2. Empezar proceso iterativo de disminución de la error en la red.	Red parametrizada (entrenada), gráficas de rendimiento y matriz de confusión.
RF5	Imagen segmentada	1. Probar en la red para etiquetar la imagen ya sea alguna figura o conector con modelo prenentrenado.	Objeto nodo con referencia en la imagen y la etiqueta.
RF6	Imagen segmentada	1. Hacer el reconocimiento de texto con modelo pre-entrenado.	Objeto nodo con referencia en la imagen y el texto.
RF7	Conjunto de nodos	1. Construir grafo dirigido a. Definir adyacencia.	Grafo dirigido o mensaje de error.
RF8	Grafo dirigido	1. Crear archivo. 2. Recorrer grafo.	Archivo con diagrama digital

		a. Traducir nodo a representación gráfica digital. 3. Mostrar resultado.	en .png.
RF9	Grafo dirigido	1. Crear archivo. 2. Recorrer grafo. a. Traducir nodo a código.	Archivo de código fuente .c.
RF10	Archivo de código fuente .c.	1. Llamar comando de compilación. 2. Verificar salida del proceso de compilación	Mostrar resultado dependiendo de la salida
RF11	Clicks	1. Si se elige entrenar, mandar a la ventana para entrenar modelo. 2. Si se elige reconocer diagrama de flujo, mandar a la ventana para hacer el reconocimiento.	Uso de las otras funcionalidades.

Tabla 9. Flujo de información para los requerimientos funcionales establecidos. Fuente: Elaboración propia.

Notas:

- En la implementación en el sistema el flujo para RF3 no se lleva a cabo según el proceso descrito, ya que no se aplica un algoritmo de segmentación como tal, en su lugar, la misma imagen preprocesada se manda a los dos modelos diferentes (texto y de figuras y conectores), logrando el mismo objetivo de la segmentación al final de cuentas que es reconocer por separado el texto del resto de símbolos.
- Para RF5 y RF6 la entrada en lugar de ser una imagen segmentada es una imagen preprocesada, es por el cambio descrito en el punto anterior.

2. Diseño del sistema

a. Arquitectura del sistema

El siguiente diagrama en lenguaje de modelado unificado (UML, por sus siglas en inglés) permite visualizar con cierto nivel de abstracción la relación entre los módulos del software a desarrollar. El preprocesamiento de imagen es una etapa necesaria para preparar la imagen para la fase de detección de los componentes del diagrama de flujo lo que va a permitir reconstruir el diagrama en formato digital, el archivo de código fuente y la compilación del mismo.

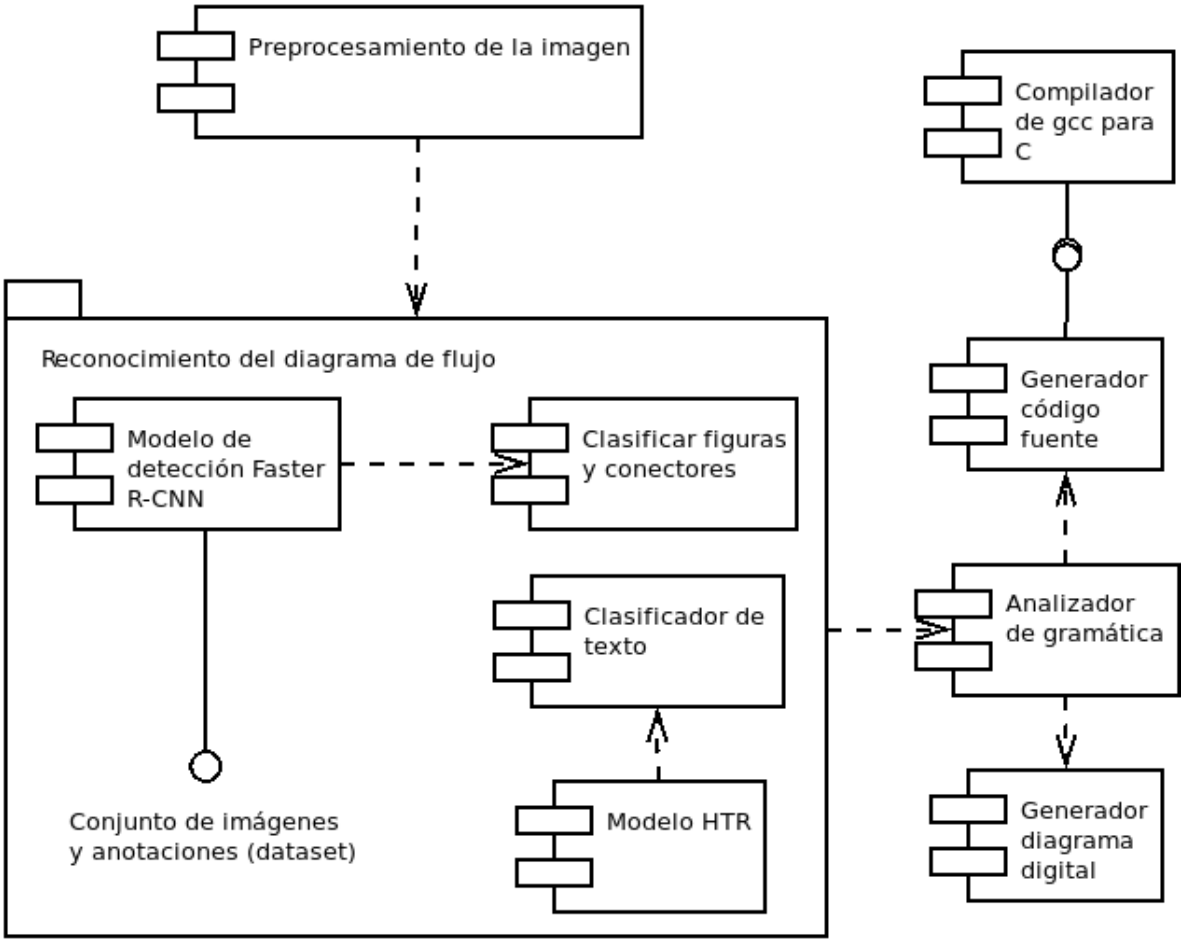


Figura 9. Arquitectura del sistema como diagrama (UML) de componentes. Fuente: Elaboración propia.

Como puede observarse se hace uso del compilador GCC instalado en el sistema operativo, esto se realiza a través de una llamada al sistema operativo donde se escribe el comando de compilación.

b. Diseño detallado

Prototipos de pantalla

Los prototipos de pantalla propuestos para la interfaz de usuario que se realizaron son los siguientes:

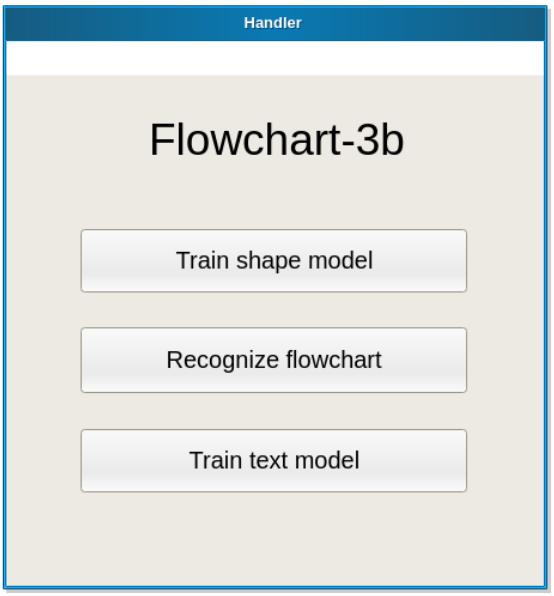


Figura 10. Ventana principal. Fuente: Elaboración propia.

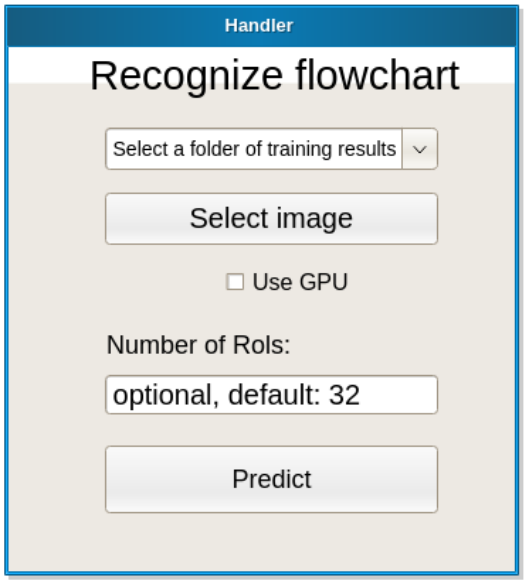


Figura 11. Ventana para reconocer un diagrama de flujo. Fuente: Elaboración propia.

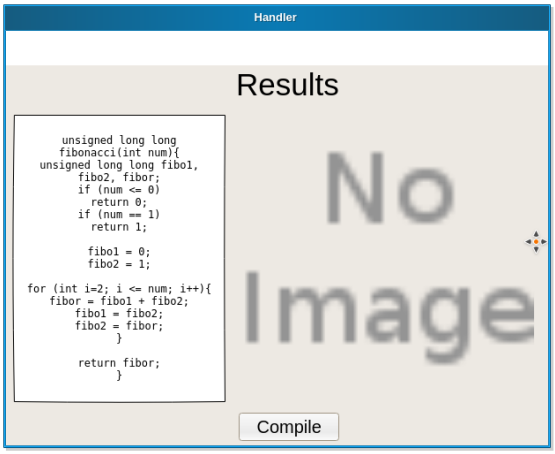


Figura 12. Ventana de resultados. Fuente: Elaboración propia.

Figura 13. Ventana auxiliar para entrenar modelo de texto. Fuente: Elaboración propia.

Figura 14. Ventana para corregir texto si es necesario. Fuente: Elaboración propia.

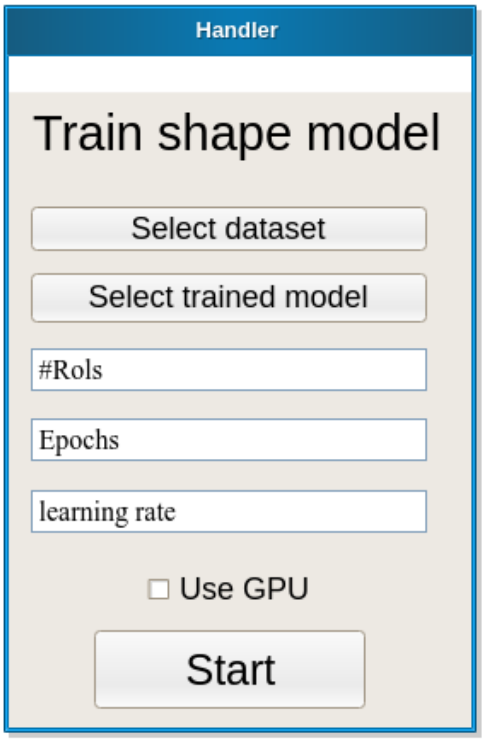


Figura 15. Ventana de entrenamiento del modelo de figuras. Fuente: Elaboración propia.

Diagrama de clases

A continuación se muestra el diagrama de clases de la versión final del sistema, ya que el diagrama es extenso se mostrará un diagrama con las relaciones entre las clases, posteriormente se mostrará cada una de las clases por separado para tener una mejor visualización de las mismas.

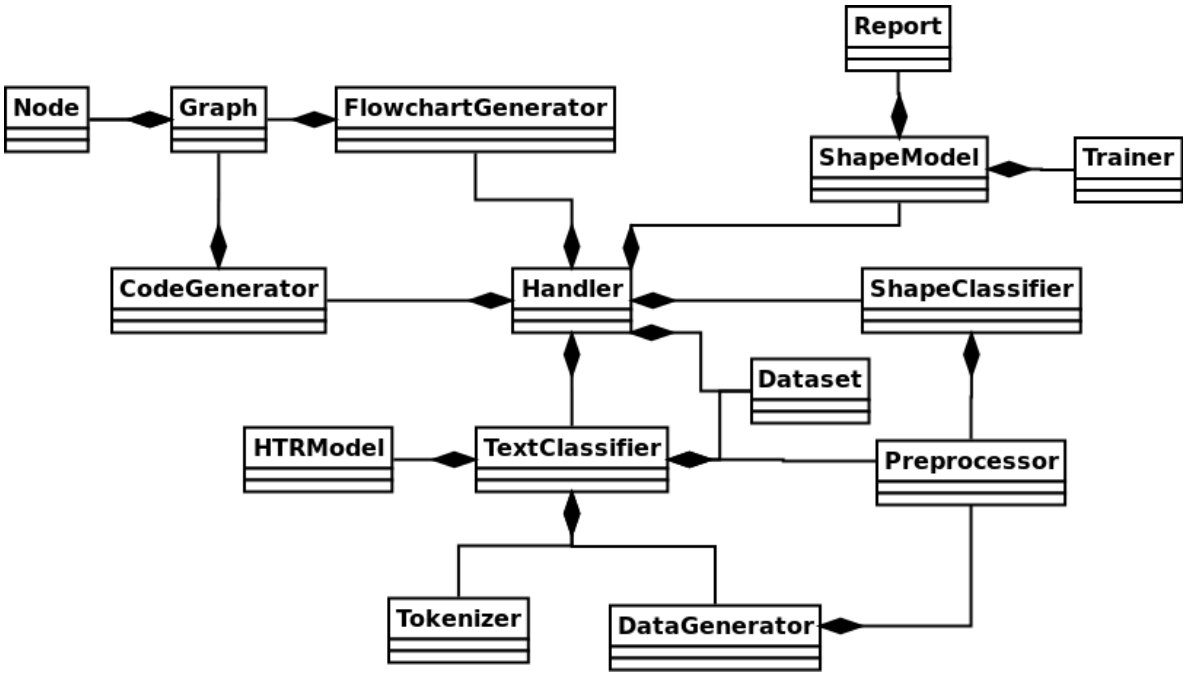
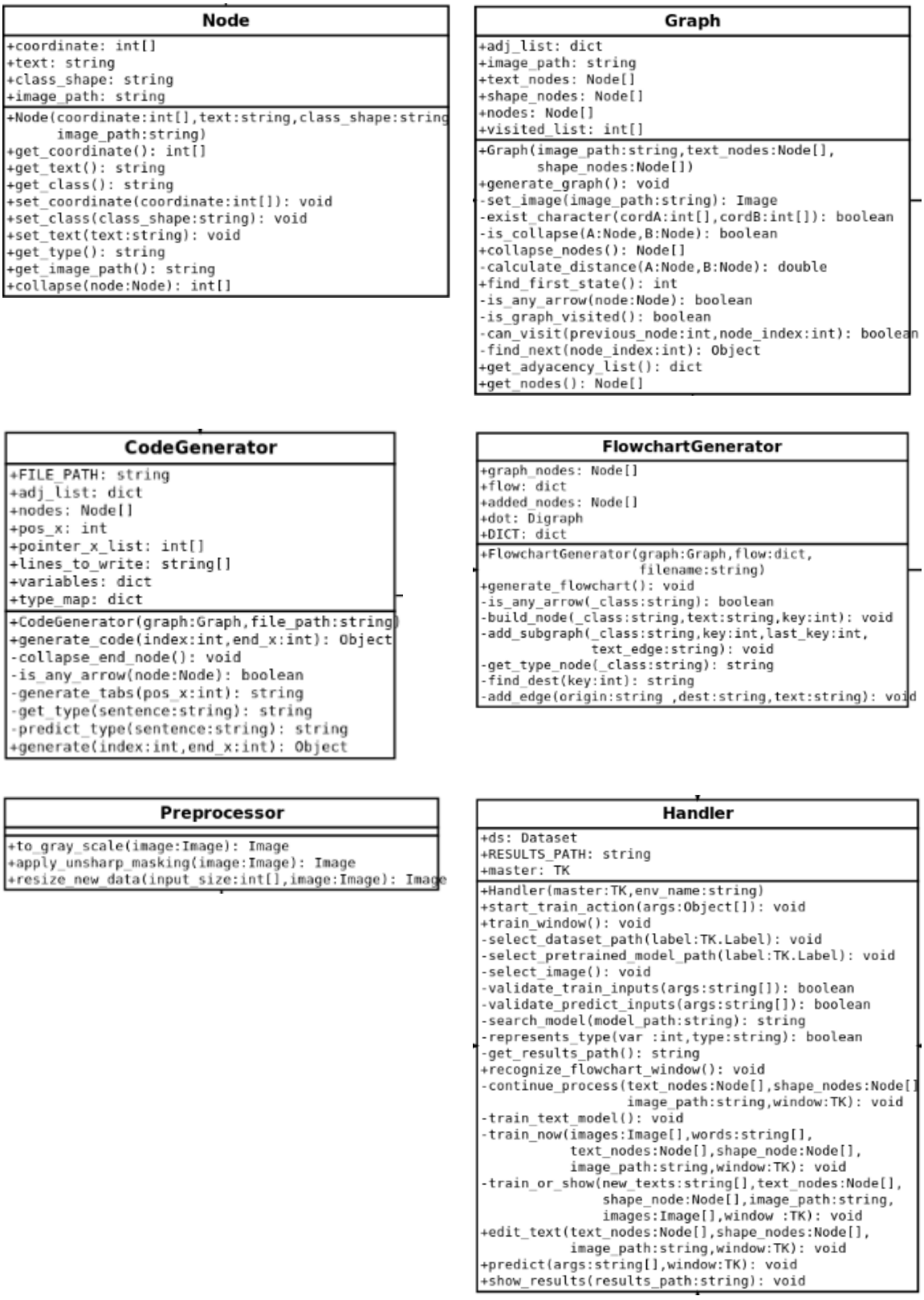


Figura 16. Representación de las relaciones del diagrama de clases. Fuente: Elaboración propia.


Node
+coordinate: int[]
+text: string
+class_shape: string
+image_path: string
+Node(coordinate:int[],text:string,class_shape:string
image_path:string)
+get_coordinate(): int[]
+get_text(): string
+get_class(): string
+set_coordinate(coordinate:int[]): void
+set_class(class_shape:string): void
+set_text(text:string): void
+get_type(): string
+get_image_path(): string
+collapse(node:Node): int[]
Graph
+adj_list: dict
+image_path: string
+text_nodes: Node[]
+shape_nodes: Node[]
+nodes: Node[]
+visited_list: int[]
+Graph(image_path:string,text_nodes:Node[],
shape_nodes:Node[])
+generate_graph(): void
-set_image(image_path:string): Image
-exist_character(cordA:int[],cordB:int[]): boolean
-is_collapse(A:Node,B:Node): boolean
+collapse_nodes(): Node[]
-calculate_distance(A:Node,B:Node): double
+find_first_state(): int
-is_any_arrow(node:Node): boolean
-is_graph_visited(): boolean
-can_visit(previous_node:int,node_index:int): boolean
-find_next(node_index:int): Object
+get_adyacency_list(): dict
+get_nodes(): Node[]
CodeGenerator
+FILE_PATH: string
+adj_list: dict
+nodes: Node[]
+pos_x: int
+pointer_x_list: int[]
+lines_to_write: string[]
+variables: dict
+type_map: dict
+CodeGenerator(graph:Graph,file_path:string)
+generate_code(index:int,end_x:int): Object
-collapse_end_node(): void
-is_any_arrow(node:Node): boolean
-generate_tabs(pos_x:int): string
-get_type(sentence:string): string
-predict_type(sentence:string): string
+generate(index:int,end_x:int): Object
FlowchartGenerator
+graph_nodes: Node[]
+flow: dict
+added_nodes: Node[]
+dot: Digraph
+DICT: dict
+FlowchartGenerator(graph:Graph,flow:dict,
filename:string)
+generate_flowchart(): void
-is_any_arrow(_class:string): boolean
-build_node(_class:string,text:string,key:int): void
-add_subgraph(_class:string,key:int,last_key:int,
text_edge:string): void
-get_type_node(_class:string): string
-find_dest(key:int): string
-add_edge(origin:string ,dest:string,text:string): void
Preprocessor
+to_gray_scale(image:Image): Image
+apply_unsharp_masking(image:Image): Image
+resize_new_data(input_size:int[],image:Image): Image
Handler
+ds: Dataset
+RESULTS_PATH: string
+master: TK
+Handler(master:TK,env_name:string)
+start_train_action(args:Object[]): void
+train_window(): void
-select_dataset_path(label:TK.Label): void
-select_pretrained_model_path(label:TK.Label): void
-select_image(): void
-validate_train_inputs(args:string[]): boolean
-validate_predict_inputs(args:string[]): boolean
-search_model(model_path:string): string
-represents_type(var :int,type:string): boolean
-get_results_path(): string
+recognize_flowchart_window(): void
-continue_process(text_nodes:Node[],shape_nodes:Node[]
image_path:string,window:TK): void
-train_text_model(): void
-train_now(images:Image[],words:string[],
text_nodes:Node[],shape_node:Node[],
image_path:string,window:TK): void
-train_or_show(new_texts:string[],text_nodes:Node[],
shape_node:Node[],image_path:string,
images:Image[],window :TK): void
+edit_text(text_nodes:Node[],shape_nodes:Node[],
image_path:string,window:TK): void
+predict(args:string[],window:TK): void
+show_results(results_path:string): void


TextClassifier
+trained_model_path: string
+dtgen: DataGenerator
+model: HTRModel
+recognizer: keras_ocr.Recognizer
+pipeline: keras_ocr.Pipeline
+TextClassifier()
+recognize(images:Image[],image_path:string): {Node[],
Images[]}
+train_new_data(): void
-set_image(image_path): Image
-exist_character_x(image:Image,x:int,ymin,
ymax:int): boolean
-exist_character_y(image:Image,y:int,xmin:int,
xmax:int): boolean
-is_collapse(A:int[],B:int[]): boolean
-merge_text_nodes(image_path:string,text_coord:int[][]): Node[]
-get_bbox(image_path:string): {int[],string[]}
+draw_boxes(image_path:string,boxes:int[]): void
-image_generator(image:Image): Image
HTRModel
+architecture: string
+input_size: int[]
+vocab_size: int
+model: Object
+HTRModel(architecture:string,input_size:int[],
vocab_size:int,greedy:boolean,beam_width:int,
top_paths:int)
+load_checkpoint(target:string): void
+get_callbacks_continue(logdir:string,checkpoint:string,
monitor:string,verbose:int): Object[]
+get_callbacks(logdir:string,checkpoint:string,
monitor:string,verbose:int): Object[]
+compile(learning_rate:float): void
+fit(x:{image[],string[]},y:{image[],string[]},
batch_size:int,epochs:int,verbose:int,
callbacks:Object[],validation_split:float,
validation_data:{image[],string[]},shuffle:boolean,
class_weight:Object,sample_weight:Object,
initial_epoch:int,steps_per_epoch:int,
validation_steps:int,validation_freq:int,
max_queue_size:int,workers:int,use_multiprocessing:boolean): Object
+predict(x:{image[],string[]},batch_size:int,
verbose:int,steps:int ,callbacks:Object[],
max_queue_size:int,workers:int,use_multiprocessing:boolean,
ctc_decode:boolean): {string[],int[]}
+puigcerver(input_size:int[],d_model:int): Object[]
Tokenizer
+vocab_size: int
+maxlen: int
+Tokenizer(chars:string,max_text_length:int)
+encode(text:string): int[]
+decode(text:int[]): string
+remove_tokens(text:string): string
DataGenerator
+tokenizer: Tokenizer
+batch_size: int
+partitions: string[]
+size: dict
+steps: dict
+index: dict
+new_dataset: dict
+DataGenerator(source:string,batch_size:int,
charset:string,max_text_length:int,
predict:boolean,load_data:boolean)
+load_data(): void
+new_next_train_batch(): {Image[],string[]}
+next_valid_batch(): {Image[],string[]}
+next_test_batch(): {Image[],string[]}
ShapeClassifier
+trained_model_path: string
+results_path: string
+class_mapping: dict
+bbox_threshold: float
+overlap_thresh_1: float
+config: Config
+overlap_thresh_2: float
+color_class: dict
+removable_threshold: float
+ShapeClassifier(results_path:string,bbox_threshold:float,
overlap_thresh_1:float,overlap_thresh_2:float,
use_gpu:boolean,num_rois:int)
-setup(): void
+predict_and_save(image:Image,image_name:string,
folder_name:string): void
+predict(image:Image,display_image:boolean): Node[]
-load_config(results_path:string): void
-get_real_coordinates(ratio:int,x1:int,y1:int,
x2:int,y2:int): int[]
-build_frcnn(): void
-load_weights(): void
-compile_models(): void
-apply_spatial_pyramid_pooling(roi:Object,
F:Object): {dict,
dict}
-generate_final_image(bboxes:dict,probs:dict,
img:Image,roi_helper:ROIHelpers,
ratio:int): {image,
dict}
-draw_rectangles(all_dets:dict,img:Image): Image
-fix_detection(all_dets:dict): dict
-is_bbox_removable(coords:int[],coords_2:int[]): boolean
+generate_nodes(dets:dict): Node[]
ShapeModel
+dataset_path: string
+num_rois: int
+weights_input_path: string
+ShapeModel(dataset_path:string,num_rois:int,
weights_input_path:string)
+train(data_augmentation:boolean,num_epochs:int,
epoch_length:int,learning_rate:float,
num_rois:int,use_gpu:boolean): void
-generate_results_path(base:string): string
+generate_classification_report(results_path:string,
generate_annotate:boolean,
use_gpu:boolean): void

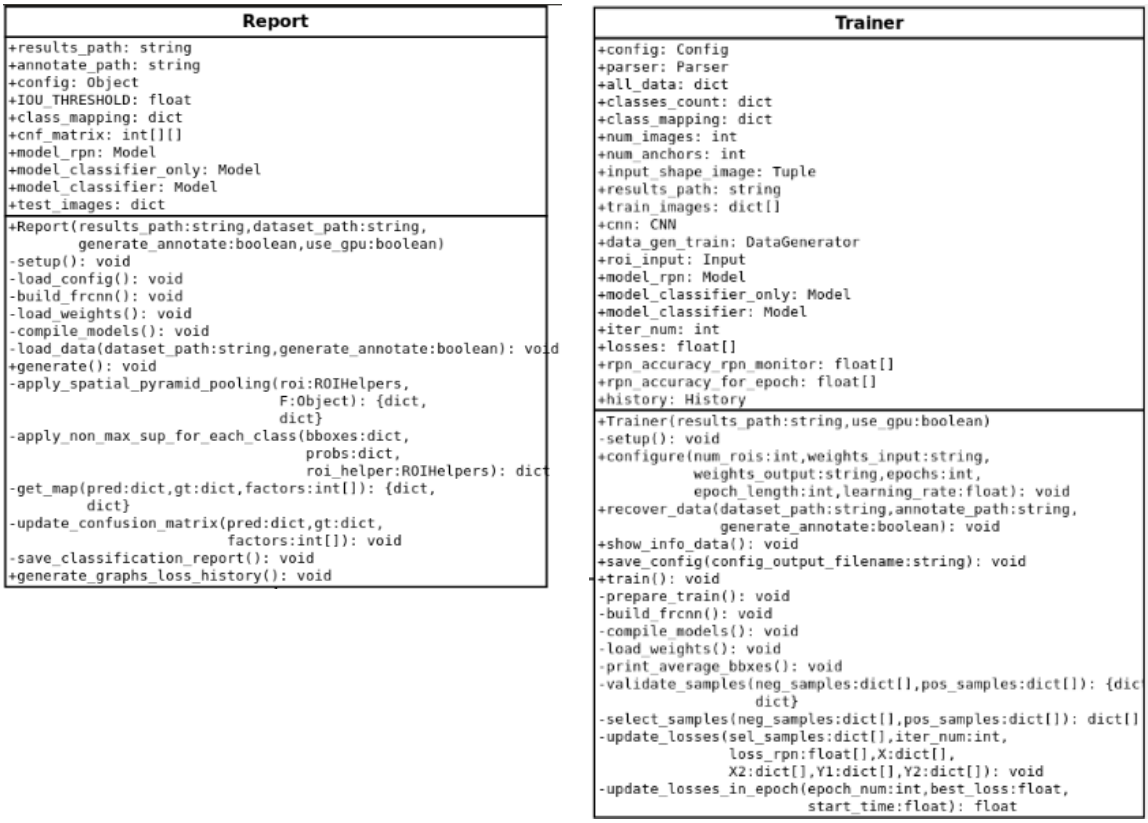


Figura 17. Clases con sus atributos y métodos. Fuente: Elaboración propia.

Pipeline

El canal de reconocimiento diseñado e implementado para resolver el problema de investigación representa una arquitectura en la cual existe un flujo de datos dentro de un proceso que se divide en varias fases secuenciales, siendo la entrada de cada una la salida de la anterior [41].

La versión inicial (planeada) y de la primer versión del sistema se pueden consultar en el apéndice H (H.1 y H.2). A continuación se muestra el pipeline utilizado para la versión 2 (final) del sistema.

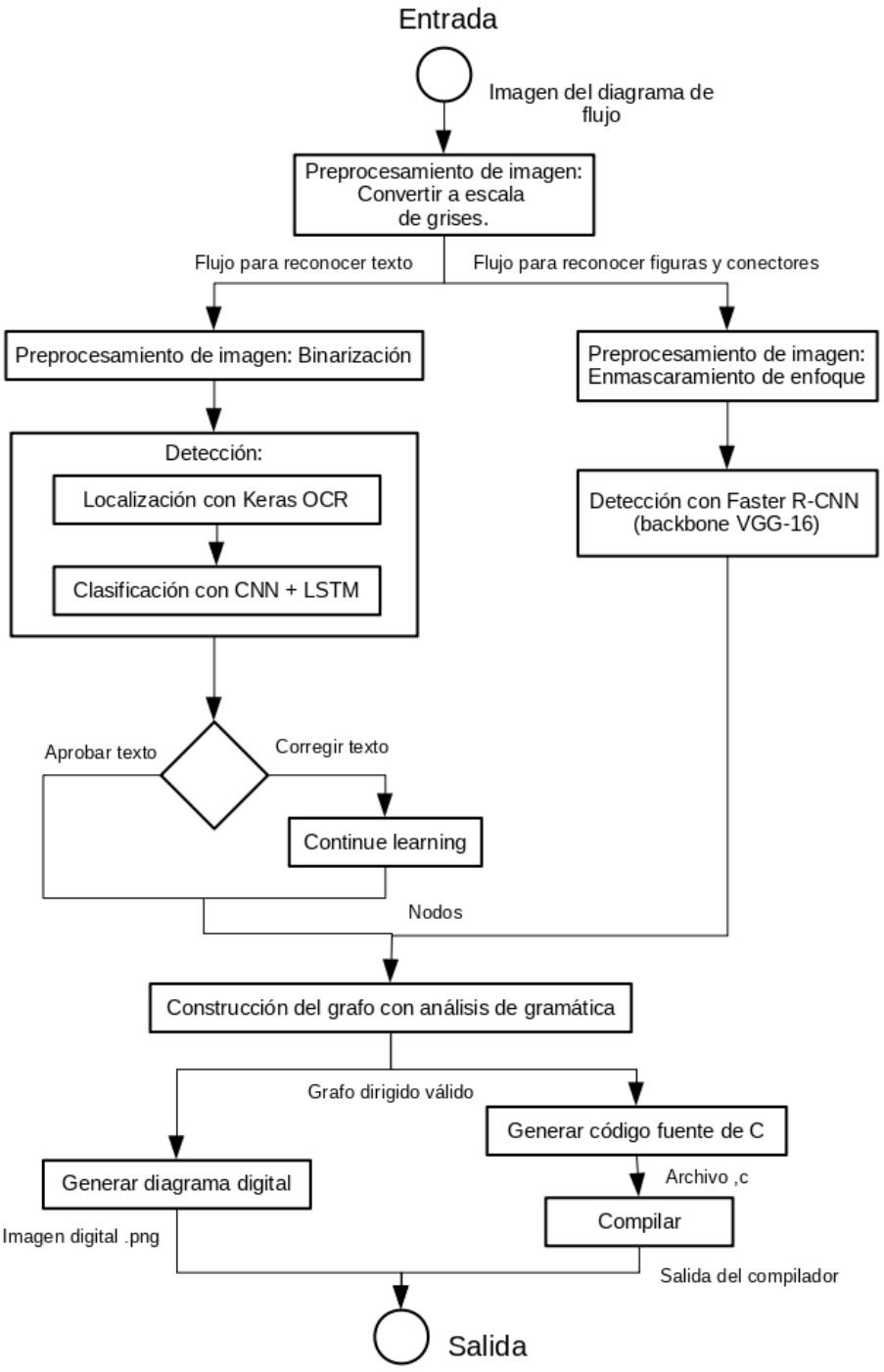


Figura 18. Canal de reconocimiento (pipeline), para la versión 2 (final) del sistema. Fuente: Elaboración propia.

Como entrada en el proceso se recibe una imagen digital de cualquier tamaño y no importando si está en escala de grises o no, dicha imagen debe ser de un diagrama de flujo hecho a mano (construido usando los símbolos definidos) que representa algún algoritmo del cual se desea generar su implementación en código fuente en C y la reconstrucción del mismo en una imagen digital. Como primer paso, se convierte la imagen a escala de grises, y la imagen resultante se manda por dos flujos diferentes que más adelante se volverán a unificar. Para el camino de detección de figuras y conectores pasa por otro algoritmo de preprocesamiento de imagen, en este caso el enmascaramiento de enfoque que ayuda a mejorar la nitidez de la imagen y luego pasa directamente al clasificador que emplea el modelo de detección Faster R-CNN el cual de forma unificada es capaz de localizar y clasificar las figuras y conectores, para así instanciar nodos por cada detección, conteniendo estos, la clase o figura de la clasificación y las coordenadas sobre la imagen de la detección. A continuación se retorna una lista de nodos. Por el lado del flujo de detección de texto, se binariza la imagen a blanco y negro, posteriormente se hace la localización de texto usando el paquete Keras OCR, con las coordenadas obtenidas se recortan las detecciones y se mandan a una red neuronal convolucional en conjunto con bloques BLSTM para hacer la clasificación; enseguida aparecerá un módulo para aprobar la detección del texto, si es necesario corregir manualmente se hará el proceso de *continual learning*, y de cualquier forma se retornará una lista de nodos donde cada uno tendrá coordenadas y texto. En el punto donde se tienen ambas listas de nodos se ha unificado el flujo nuevamente. Los nodos van a la entrada del análisis de gramática, donde se construye un grafo dirigido siempre que haya un flujo permisible y así se genera un grafo dirigido válido que va hacia los generadores de la salida, en este caso el generador de código fuente y el generador del diagrama de flujo en formato digital, donde si no hay problemas procederán a guardar sus salidas en una carpeta de resultados y mostrarse en una ventana de resultados.

Diseño del modelo de reconocimiento de figuras y conectores

La imagen de la estructura del modelo de detección Faster R-CNN con backbone VGG-16 (con el que se obtuvieron mejores resultados) se generó usando una utilidad de Keras que lo genera en una imagen.

El diseño del modelo consta de la capa de entrada, capas de convolución y pooling (usando la función de agrupación de máximo) que generan el mapa de características para el RPN que tiene un clasificador (de objeto o no objeto) y regresor para ajustar las coordenadas, la capa de RoI pooling y las capas de superiores FC donde hay dos capas de regularización (para evitar el problema del sobreajuste) "Dropout" y el clasificador, que tiene al final un capa llamada "dense_class_10" y "dense_regress_10" las cuales darán de salida la clase con mayor probabilidad y la coordenadas del objeto detectado, respectivamente.

En la figura 19, del diseño del modelo se pueden observar que las tablas que van conectadas por un flujo tienen valores "None" y otros numéricos, son atributos de la forma de los tensores de Keras, estos valores dependen del tamaño de la imagen de entrada, para un modelo entrenado va a depender de un hiperparámetro que se encuentra en la configuración del modelo, específicamente, el tamaño mínimo al cual se va redimensionar la imagen respecto de su lado más grande.

Cabe destacar que este modelo Faster R-CNN fue propuesto en [42] y su backbone es una arquitectura conocida que se presentó en [43].

Figura 19. Arquitectura completa del modelo Faster R-CNN. Fuente: Elaboración propia.

Diseño del modelo de reconocimiento de texto

El modelo utilizado para el reconocimiento de texto es una implementación en Keras del modelo publicado puigcerver[44]. La arquitectura del modelo está constituido por bloques de convolución donde cada bloque contiene una capa convolucional bidimensional, para reducir el sobreajuste se aplica "Dropout" (técnica de regularización para evitar sobreajuste) en la entrada de algunas capas convolucionales después de cada capa convolucional se aplica la normalización por lotes, además de unidades lineales rectificadoras con fugas (LeakyReLU) como función de activación en los bloques convolucionales. Por último, una capa de agrupación máxima (Maxpool) para reducir la dimensionalidad de las imágenes de entrada. Después de los bloques de convolución se aplican los bloques recurrentes los cuales están formados por capas bidireccionales 1D-LSTM que procesan la imagen de entrada en columna en orden de izquierda a derecha y de derecha a izquierda. En la salida de los bloques BLSTM se aplica nuevamente "Dropout".

Finalmente, después de los bloques recurrentes cada columna debe asignarse a una etiqueta de salida, para eso, la profundidad es transformada a un tamaño igual al número de caracteres + 1 para el símbolo de espacio en blanco. Todos los parámetros de la red neuronal son entrenadas para minimizar la pérdida CTC (Connectionist Temporal Classification), se usó el algoritmo RmsProp para incrementar la actualización de parámetros del modelo utilizando los gradientes de la pérdida de CTC. A continuación, se muestra visualmente la arquitectura antes descrita:

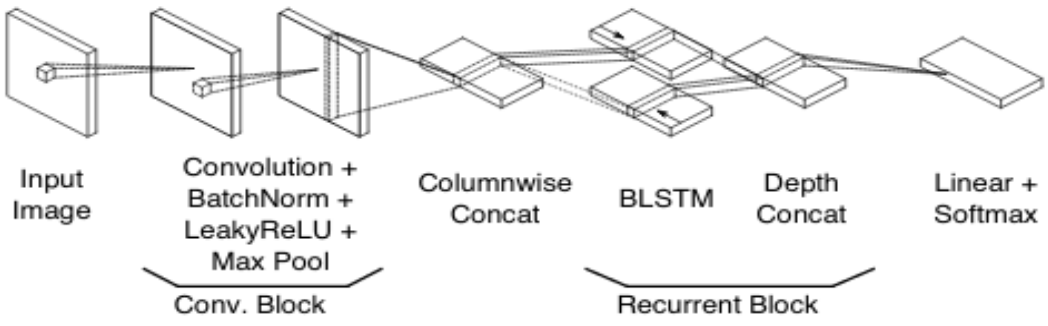


Figura 20. Arquitectura de la red neuronal del modelo puigcerver. Fuente: [44].

Construcción del grafo

El siguiente diagrama de Conway describe la secuencia que debe seguir un flujo correcto para generar la lista de adyacencia. Según la clase del nodo que se esté analizando se va tomando en cuenta las reglas que los diagramas de flujo como tal tienen para construirse.

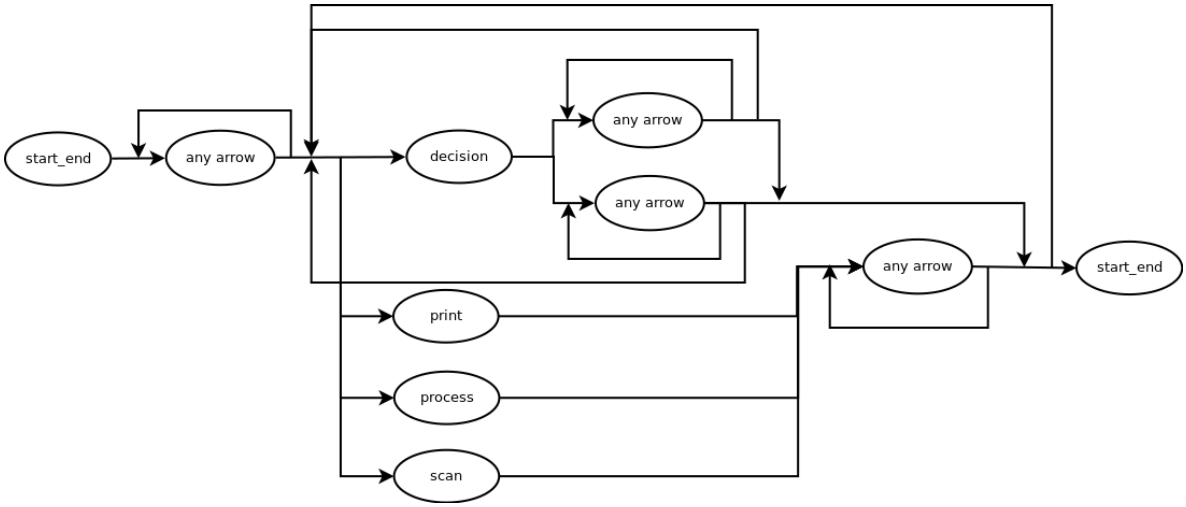


Figura 21. Diagrama de Conway que muestra la gramática para construir el diagrama de flujo. Fuente: Elaboración propia.

El proceso para generar la lista de adyacencia es el siguiente, se debe de considerar que este proceso trabaja con un objeto de tipo nodo a la vez, teniendo esto en consideración primero se colapsan los nodos generados por el modelo de reconocimiento de figuras que llamaremos “nodos figura” con los nodos generados por el modelo de reconocimiento de texto que llamaremos nodos texto, teniendo los nodos figura en una lista y los nodos texto en otra, para todo nodo figura se busca que nodo texto está colisionando en él y se convierte en un solo nodo el cual ya tiene el tipo de figura que es y el texto que contiene, a continuación, se busca cual de los nodos resultantes de colapsar las listas de nodos es de la clase “start_end” y tiene como texto “inicio”, una vez ubicado este nodo se empieza el proceso de construcción de la lista de adyacencia la cual es la estructura de datos que se eligió para almacenar las asociaciones de los nodos, el proceso como ya se mencionó

trabaja con un nodo a la vez pudiendo así trabajar como una máquina de estados, cuyos cambios en el estado están dictados por las reglas gramaticales que los diagramas de flujo tienen mostradas en la figura 21, se analiza el nodo actual verificando la clase a la que pertenece y calculando las distancias euclidianas entre el nodo analizado con los otros nodos de la lista y se verifica si el nodo más cercano, o sea, el nodo que tenga la menor distancia euclidiana es de la clase que puede seguirle a la clase del nodo analizado, si es el caso de que sí pertenezca a una clase posible se asocian los nodos y se marca como visitado al nodo que es más cercano al nodo analizado para minimizar la complejidad algorítmica ya que para los siguientes nodos por analizar ya no se tienen que analizar los nodos que ya fueron visitados. La excepción es cuando es un ciclo donde la tolerancia de visita para un nodo de clase "desicion" es de 2, este proceso se realiza hasta que todos los nodos fueron visitados o bien en algún nodo no se cumplió que nodo con menor distancia es de la clase a la que debería de seguir o cuando un nodo se visitó a un nodo más de tolerancia que tiene cada clase, la cual, como ya fue mencionado, para todas las clases es de tolerancia igual a 1 con excepción del nodo de "decision" que es de 2. Al final se tiene una lista de adyacencia con todas las relaciones de nodos o bien un mensaje indicando que el diagrama no es válido.

Diagrama de flujo del preprocesamiento de imágenes

Se diseña un diagrama de flujo para mostrar el proceso que se sigue para preparar las imágenes de los diagramas de flujo para la tarea de detección, tanto de texto así como de figuras y conectores. Las operaciones de mayor relevancia son las que se destacan en negritas.

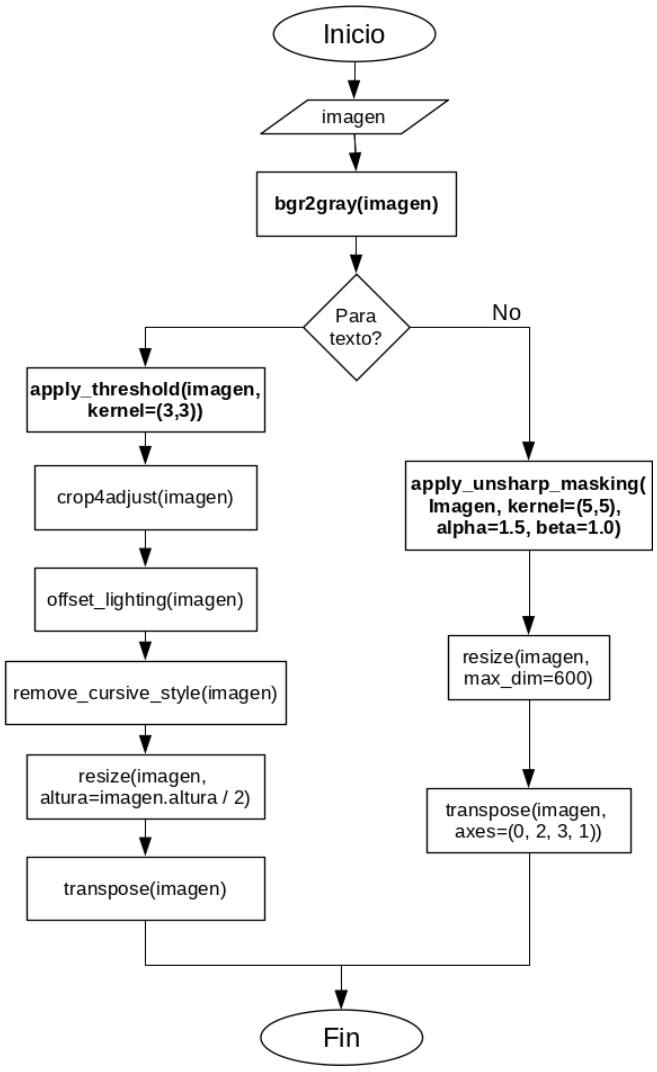


Figura 22. Diagrama de flujo para el preprocesamiento de la imagen. Fuente: Elaboración propia.

A grandes rasgos se puede observar que la imagen que se manda al modelo de texto se binariza mientras que la imagen que va al modelo de detección de figuras y conectores se le aplica enmascaramiento de enfoque, en ambos algoritmos, internamente se emplea desenfoque gaussiano.

c. Matriz de trazabilidad

La matriz de trazabilidad como parte del control del cumplimiento de requerimientos, se fue llenando la columna de pruebas una vez que estas se ejecutaron las mismas teniendo un status (ver sección más adelante “Seguimiento al plan de pruebas”) aceptable.

Objetivo	Requerimiento	Diseño	Componente	Prueba
Hacer reconocimiento de figuras y conectores plasmados en una imagen digital de un diagrama de flujo trazado a mano.	RF1 - Cargar imagen.	Diagrama de clases	Preprocesador	EPU 001
				EPI 001
	RF2 - Preprocesamiento de la imagen	Diagrama de clases	Preprocesador	EPU 001
				EPI 001
				EPI 003
				EPI 004
		Diagrama de flujo	Preprocesador	EPU 001
				EPI 001
	RF3 - Segmentar figuras/conectores del texto.	Diagrama de clases	Segmentador	n/a
	RF4 - Entrenar modelo de figuras.	Diagrama de clases	Modelo de figura	EPU 006
				EPS 002
		Arquitectura de la CNN		EPI 009
		Esquema de persistencia de datos	Dataset	EPU 005
	RF5 - Clasificación de figuras y conectores.	Diagrama de clases	Clasificador de figuras	EPU 004

		Pipeline		EPI 003
Hacer reconocimiento del texto manuscrito en los diagramas de flujo.	RF3 - Segmentar figuras/conectores del texto.	Diagrama de clases	Segmentador	n/a
	RF6 - Clasificación de texto.	Diagrama de clases	Clasificador de texto	EPU 007
				EPU 019
				EPU 020
				EPI 004
Usar análisis de gramática para verificar la estructura del diagrama de flujo.	RF7 - Análisis de gramática.	Diagrama de Conway	Grafo	EPU 011
				EPI 006
		Diagrama de clases	Analizador gramática	EPI 005
			Nodo	EPU 008
Poder digitalizar diagrama de flujo.	RF8 - Generar versión digital del diagrama de flujo.	Diagrama de clases	Generador de diagrama de flujo	EPU 013
				EPI 008
				EPS 001
		Esquema de persistencia de datos	Guardar imagen	EPS 001
Poder generar código fuente de C a partir del diagrama de flujo.	RF9 - Generar código fuente en C equivalente.	Diagrama de clases	Generador de código	EPU 014
				EPI 007
				EPS 001
		Esquema de persistencia de datos	Guardar archivo .c	EPS 001
	RF10 - Compilar código fuente.	Diagrama de clases	Generador de código	EPU 015
GUI de control (manejador).	RF11- Disponer de una GUI para la interacción con el sistema.	Diagrama de clases	Abrir las vistas para usar las funciones del sistema.	EPU 016
		Prototipos GUI		EPU 017
				EPU 018
				EPU 019
				EPU 020
				EPS 001
				EPS 002

Tabla 10. Matriz de trazabilidad. Fuente: Elaboración propia.

d. Diseño de la base de datos

Dataset para el modelo de reconoconocimiento de texto

A continuación se muestra un diagrama que describe la estructura del archivo iam.hdf5 donde se encuentra el conjunto de imágenes y sentencias del dataset IAM [45] que se usa para entrenar el modelo de reconoconocimiento de texto.

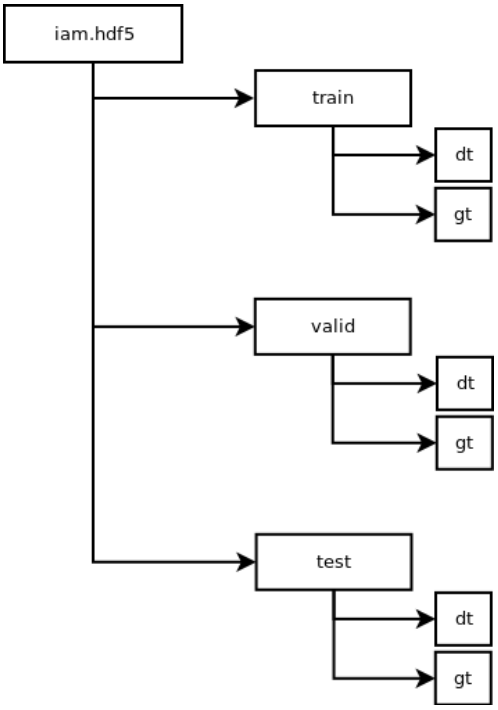


Figura 23. Organización del archivo para el dataset IAM. Fuente: Elaboración propia.

Donde dt es una lista de imágenes preprocesadas y gt es una lista de cadenas representando la sentencia de caracteres que la imagen asociada a su posición en el arreglo contiene.

Dataset para el modelo de reconocimiento de figuras y conectores

Para ver la especificación de dataset puede consultar el apéndice L (L.1), la estructura empleada para la versión final del conjunto de datos se puede observar en la siguiente figura:

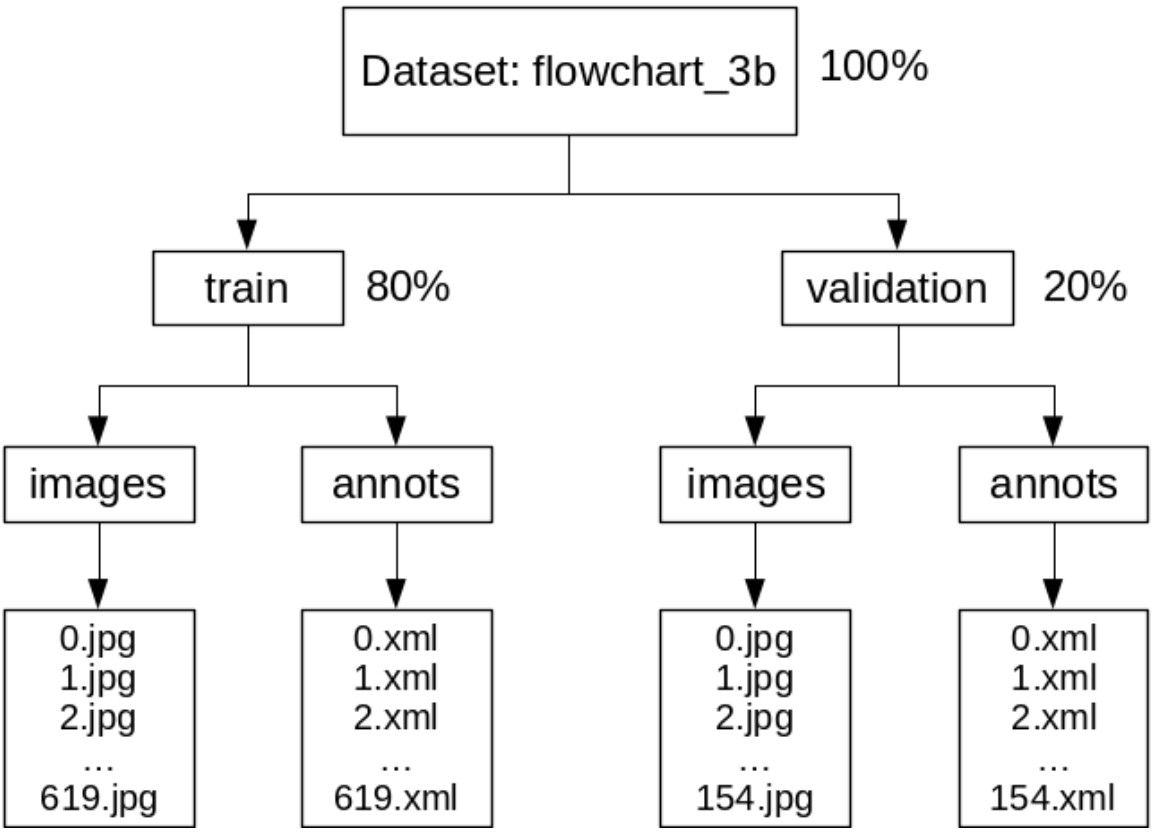


Figura 24. Estructura del dataset empleado para el entrenamiento del modelo de figuras y conectores. Fuente: Elaboración propia.

La base de datos está dividida en dos partes, conjunto de entrenamiento y de validación, siendo el 80% del total de dataset para el proceso de entrenar y el restantes 20% para la validación que es con el que se genera un reporte en el cual se encuentran métricas relevantes de la tarea de detección. Este formato sigue el tener un archivo de extensión .xml donde se encuentran las etiquetas o clases y coordenadas de cada objeto en la imagen correspondiente, que en este caso la correspondencia es por el nombre-id de los archivos, es decir, las anotaciones de objetos que se encuentran en 0.xml son de la imagen 0.jpg.

e. Manejo de archivos

A continuación se muestra un diagrama que describe la forma en que los archivos (código fuente en C y diagrama de flujo digitalizado) producidos son guardados después del proceso de reconocimiento.

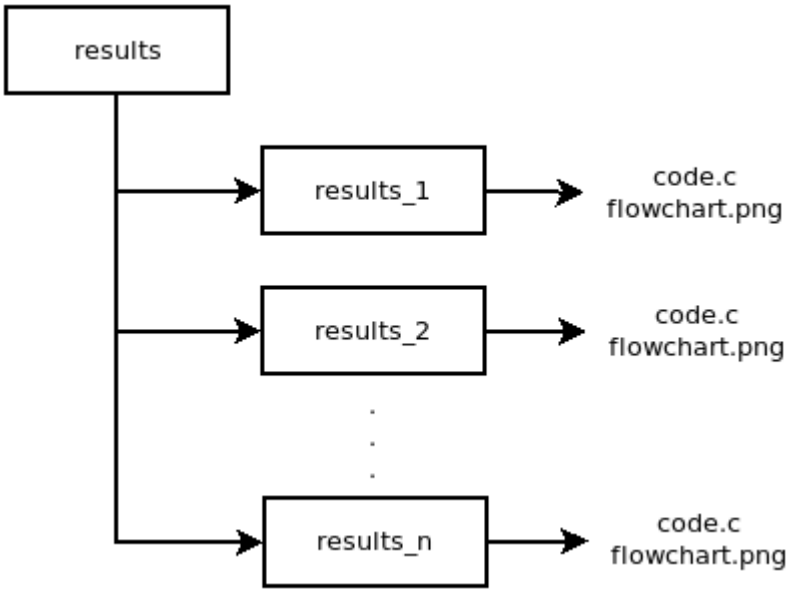


Figura 25. Organización del directorio para los resultados. Fuente: Elaboración propia.

La carpeta “results” se encuentra en la raíz de la carpeta del proyecto.

Por otra parte, para el entrenamiento del modelo de detección de figuras y texto se guardan los resultados en una carpeta llamada “training_results”, a continuación se muestra la estructura de este directorio y el contenido de una de las carpetas:

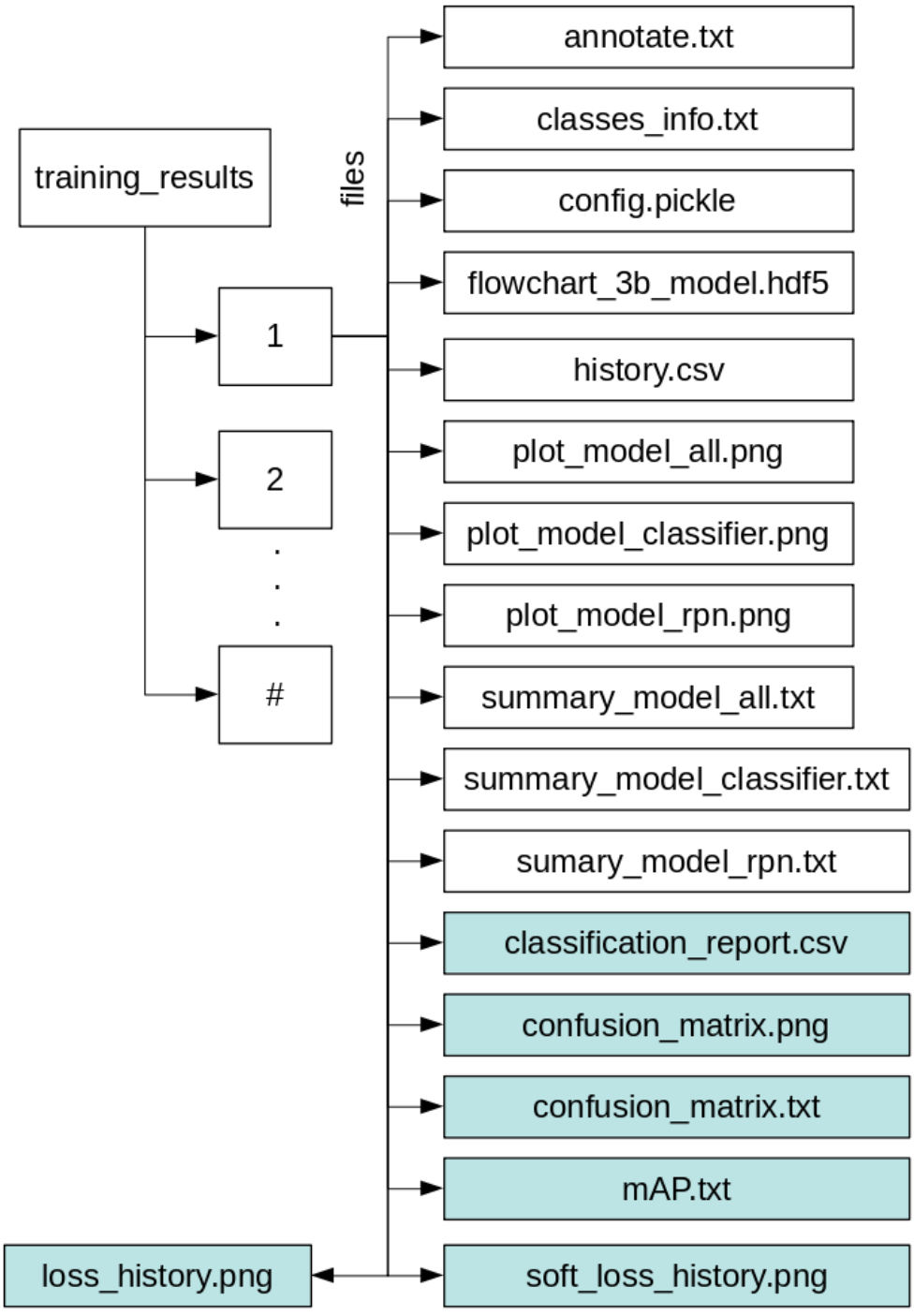


Figura 26. Organización del directorio donde se guardan los resultados del entrenamiento del modelo de detección de figuras y conectores. Fuente: Elaboración propia.

Los recuadros con fondo azul, son generados como parte del reporte para el modelo con el fin de evaluar qué tan bueno es el modelo para la tarea de detección en el dataset de validación, así como para generar las gráficas.

- annotate.txt: Se genera una lista a partir del dataset donde en cada ítem, se tienen los datos de objetos (nombre de la imagen donde está, coordenadas, clase y una bandera que indica si pertenece al conjunto de entrenamiento o validación).
- classes_info.txt: Guarda el número de objetos (figuras y conectores) que hay por cada clase para el entrenamiento del modelo, además del número de clases contando el fondo como una clase más.
- config.pickle: Guarda los hiperparámetros para entrenar el modelo y posteriormente usarlos en el "clasificador" (detector de figuras y conectores).
- flowchart_3b_model_hdf5: Guarda los pesos resultado del entrenamiento para el modelo de detección. Estos se cargan por el clasificador para hacer la tarea de detección.
- history.csv: Donde se registra métricas relativas al proceso de entrenamiento cada vez que se completa una época en el entrenamiento. Durante la generación del reporte de aquí se toma la pérdida para así visualizar la evolución en la convergencia del modelo.
- plot_model_all.png, plot_model_classifier.png, plot_model_rpn: Imágenes generadas que permiten observar la arquitectura completa del modelo, el clasificador y el RPN.
- summary_model_all.txt, summary_model_classifier.txt, summary_model_rpn.txt: Son archivos de texto plano donde se encuentra las capas que constituyen la arquitectura del modelo completo, solo el clasificador y el RPN, respectivamente. Adempas, en cada archivo se puede encontrar el número de parámetros que emplea el modelo.
- classification_report.csv: En este archivo se muestra una relación de las clases con dos métricas referentes a la clasificación.

- confusion_matrix.png y confusion_matrix.txt: Es una imagen donde se visualiza la matriz de confusión graficando por cada clase lo predicho vs la verdad fundamental, y la matriz de confusión en forma de archivo donde en lugar de una escala de colores hay valores numéricos, respectivamente.
- mAP.txt: Contiene los valores obtenidos respecto a la métrica AP (Average Precision) por cada clase y el promedio de dichos valores; además la cantidad de objetos detectados y los valores que en realidad existen, incluyendo el total de imágenes del dataset de validación.
- loss_history.png y soft_loss_history: Son dos gráficas que muestran la evolución de la convergencia del modelo entrenado respecto de la pérdida, soft_loss_history es la gráfica con las épocas y la menor pérdida obtenida hasta la i-ésima época.

La carpeta “training_results” se encuentra dentro de la carpeta “model” la cual se encuentra ubicada en la raíz de la carpeta del proyecto.

3. Construcción

Reconociendo un diagrama de “Hola mundo” con modelo de texto no reforzado con entrenamientos “Continual learning”

Figura 27. Reconociendo el “Hola mundo” sin haber hecho *continual learning*: Diagrama a reconocer. Fuente: Elaboración propia.

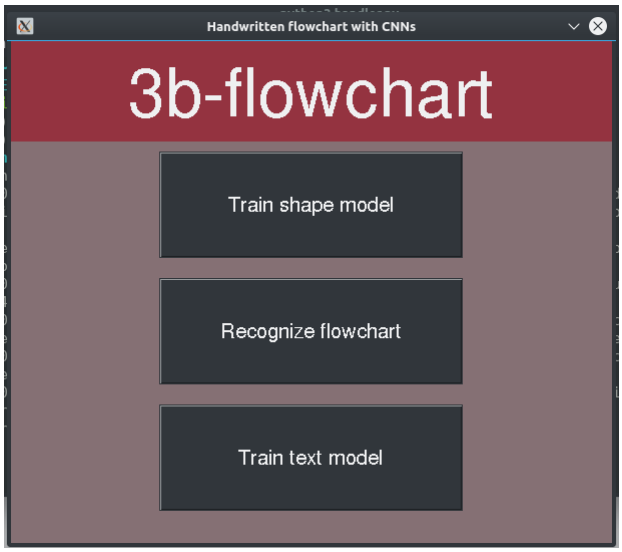


Figura 28. Reconociendo el "Hola mundo" sin haber hecho *continual learning*: Ventana principal. Fuente: Elaboración propia.

En la ventana principal se hace click sobre el botón "Recognize flowchart". Y aparece la siguiente ventana:

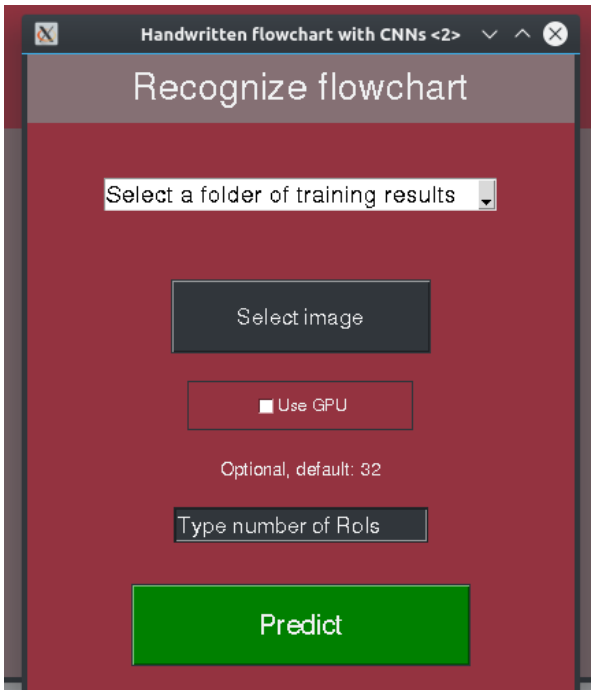


Figura 29. Reconociendo el "Hola mundo" sin haber hecho *continual learning*: Ventana para hacer el reconocimiento. Fuente: Elaboración propia.

Aquí se debe elegir la carpeta donde se encuentra alguno de los modelos entrenados, la imagen del diagrama, si se usará GPU y el número de regiones de interés (opcional).

Al darle click al botón “Select image” se abre un diálogo para elegir la imagen a reconocer.

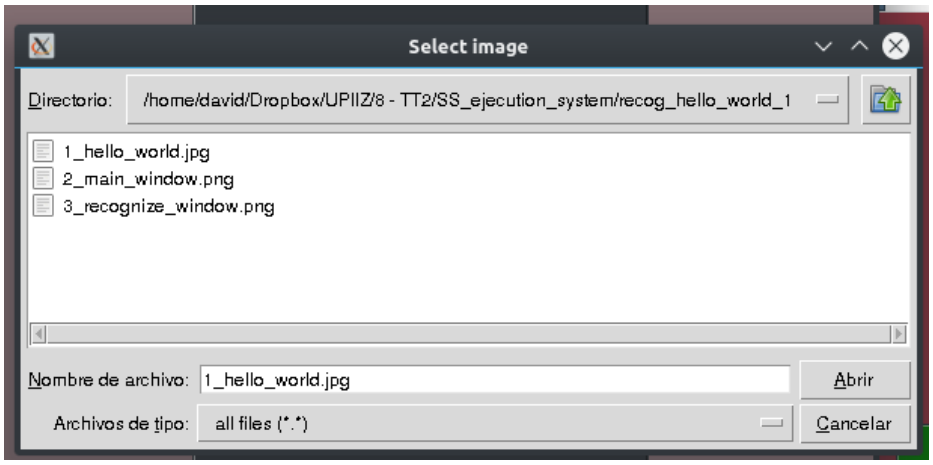


Figura 30. Reconociendo el “Hola mundo” sin haber hecho *continual learning*: Seleccionar imagen. Fuente: Elaboración propia.

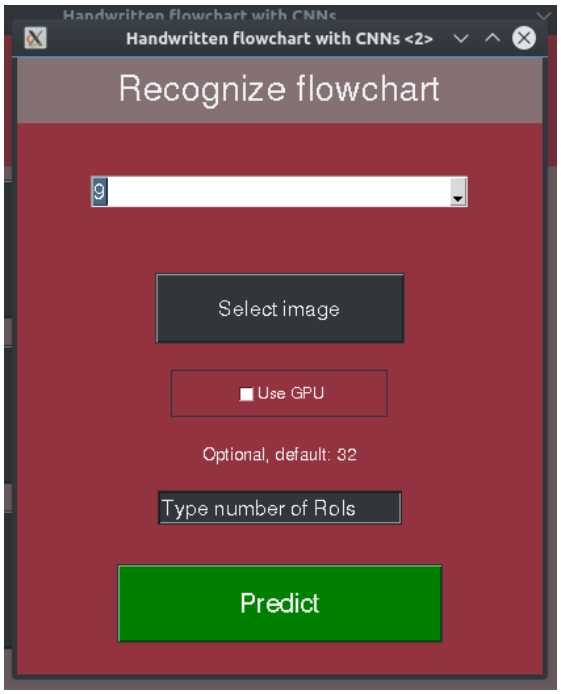


Figura 31. Reconociendo el “Hola mundo” sin haber hecho *continual learning*: Reconociendo. Fuente: Elaboración propia.

Una vez elegido la carpeta de resultados y la imagen, se da click en el botón “Predict” para generar los resultados.

Figura 32. Reconociendo el “Hola mundo” sin haber hecho *continual learning*: Editando el texto. Fuente: Elaboración propia.

Aquí se muestra el resultado de la predicción del texto, como se puede observar no son clasificaciones correctas (debido a que el modelo no ha seguido aprendiendo de acuerdo a las pruebas, es decir, no se ha aplicado “continual learning”), por lo tanto, se procederá a corregirlo manualmente.

Figura 33. Reconociendo el "Hola mundo" sin haber hecho *continual learning*: Texto corregido. Fuente: Elaboración propia.

Una vez editado, se da click en "Continue", y aparecerá la siguiente ventana para decidir hacer el proceso de "continual learning" o no.

Figura 34. Reconociendo el "Hola mundo" sin haber hecho continual learning: Entrenar o no. Fuente: Elaboración propia.

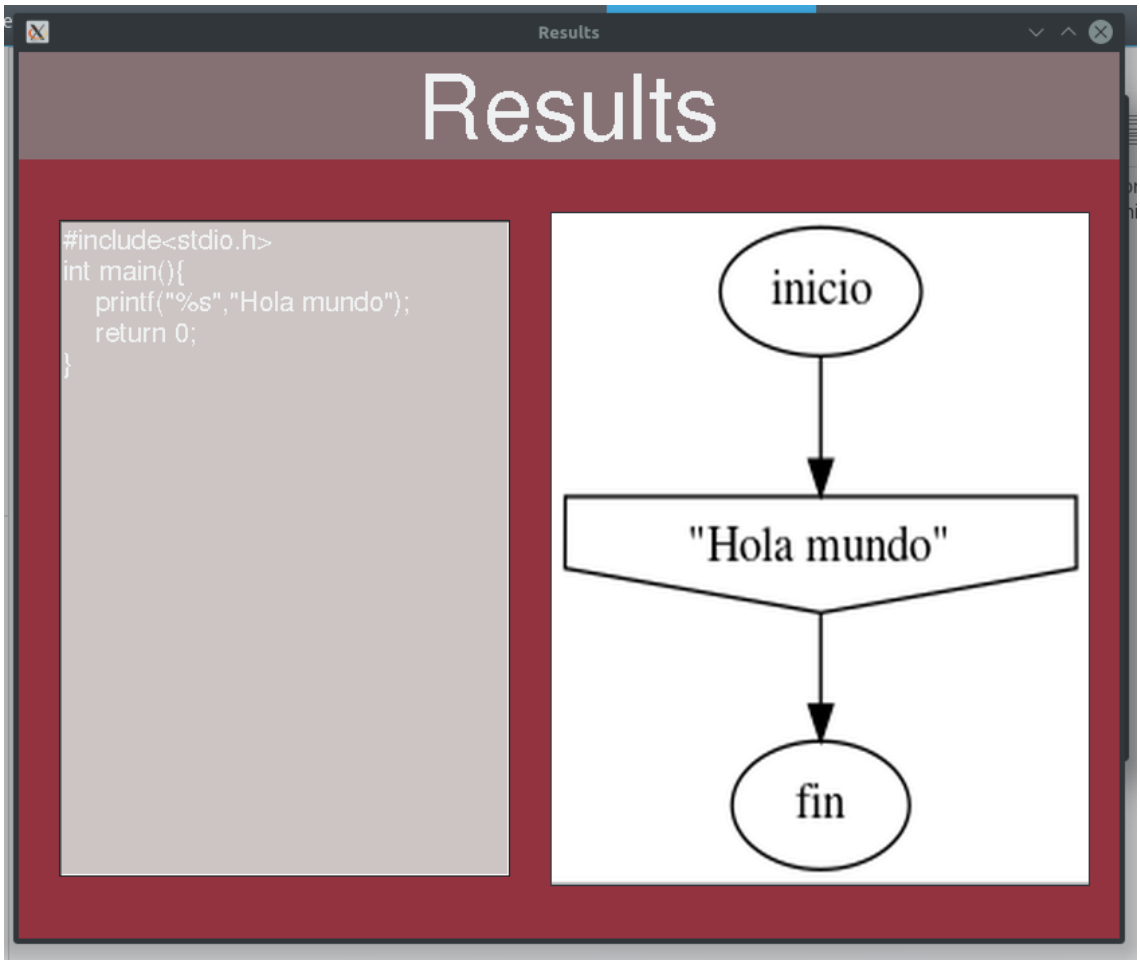


Figura 35. Reconociendo el “Hola mundo” sin haber hecho *continual learning*: Resultados. Fuente: Elaboración propia.

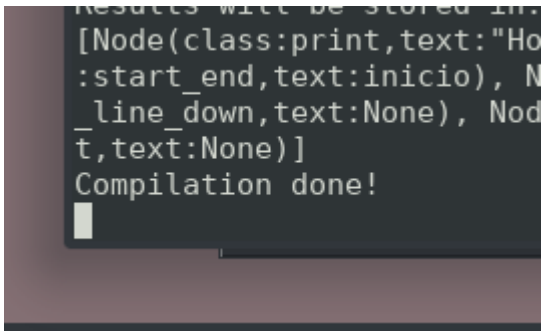


Figura 36. Reconociendo el “Hola mundo” sin haber hecho *continual learning*: Salida de la compilación. Fuente: Elaboración propia.

Finalmente, se muestran los resultados, en la ventana de “Results” está el código fuente en C y a la derecha la imagen del diagrama de flujo digital reconstruido.

En la figura 36. que es una captura de la Terminal se muestra la salida de compilación del código, pudiendo observar que no hay ningún mensaje de error y/o warning.

Reconociendo un diagrama de “Imprimir pares” con modelo de texto reforzado con entrenamientos “Continual learning”

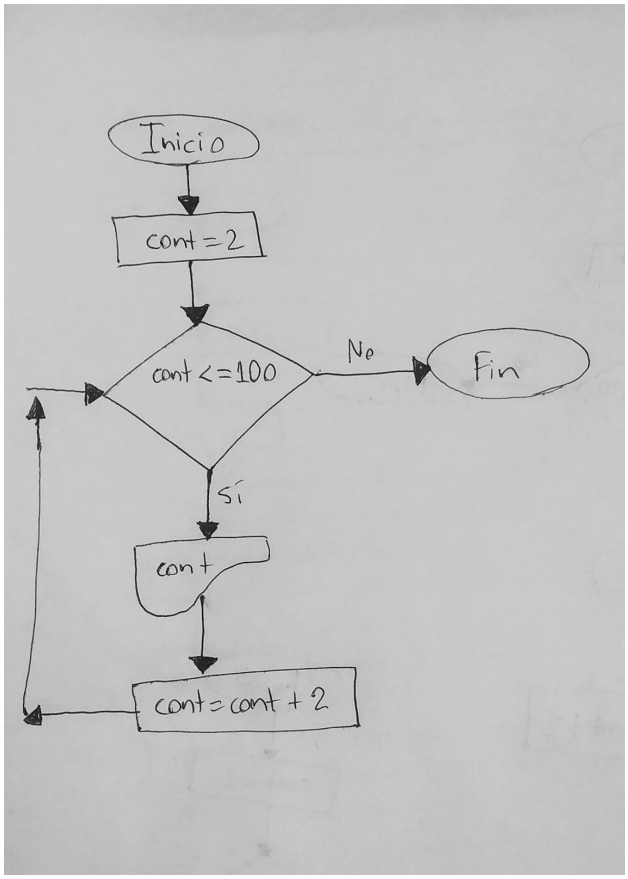


Figura 37. Reconociendo “Imprimir pares” con *continual learning*: Diagrama a reconocer. Fuente: Elaboración propia.

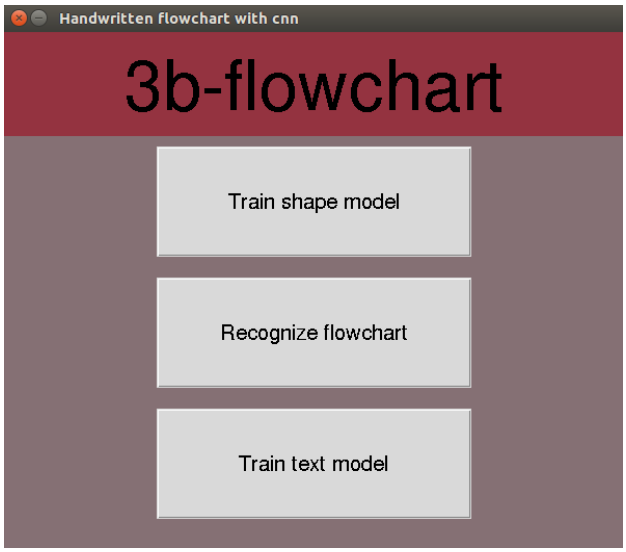


Figura 38. Reconociendo "Imprimir pares" con *continual learning*: Ventana principal. Fuente: Elaboración propia.

En la ventana principal se hace click sobre el botón "Recognize flowchart". Y aparece la siguiente ventana:

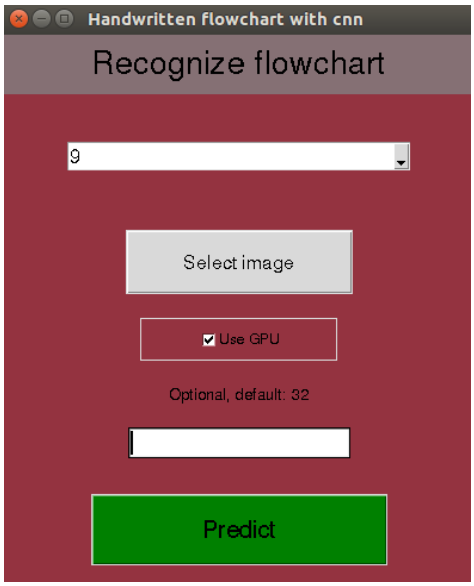


Figura 39. Reconociendo "Imprimir pares" con *continual learning*: Ventana para hacer el reconocimiento. Fuente: Elaboración propia.

Aquí se debe elegir la carpeta donde se encuentra alguno de los modelos entrenados, la imagen del diagrama de flujo a reconocer, si se usará GPU y el número de regiones de interés (opcional).

Al darle click al botón “Select image” se abre un diálogo para elegir la imagen a reconocer

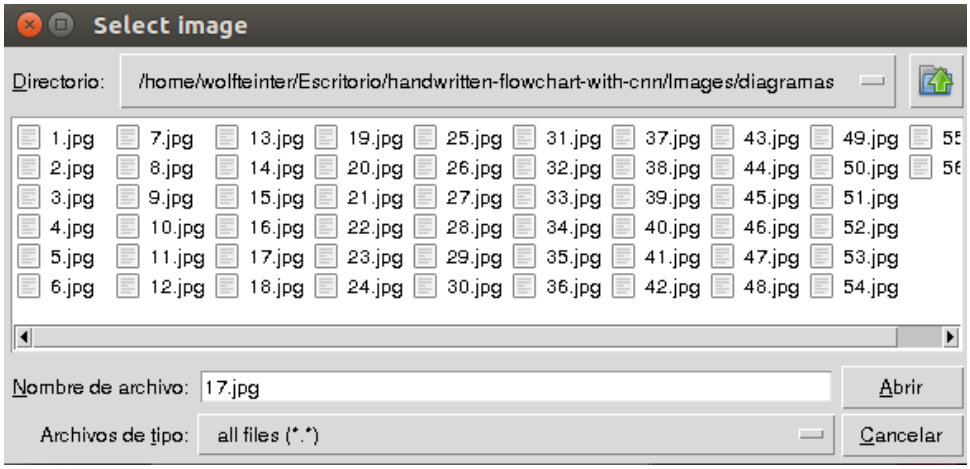


Figura 40. Reconociendo “Imprimir pares” con *continual learning*: Seleccionar imagen. Fuente: Elaboración propia.

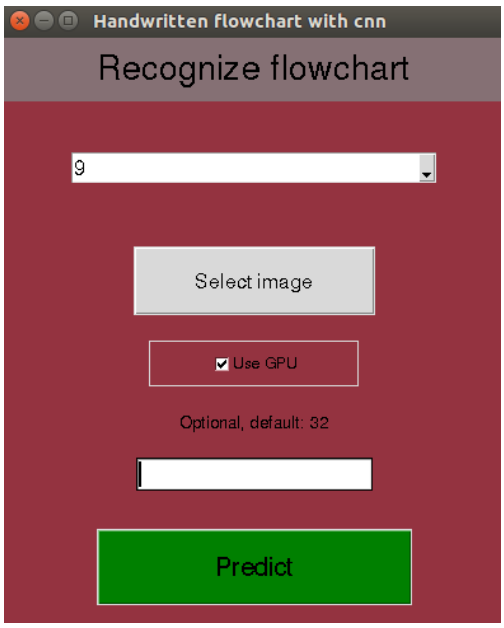


Figura 41. Reconociendo “Imprimir pares” con *continual learning*: Reconociendo. Fuente: Elaboración propia.

Una vez elegido la carpeta de resultados y la imagen, se da click en el botón “Predict” para generar los resultados.

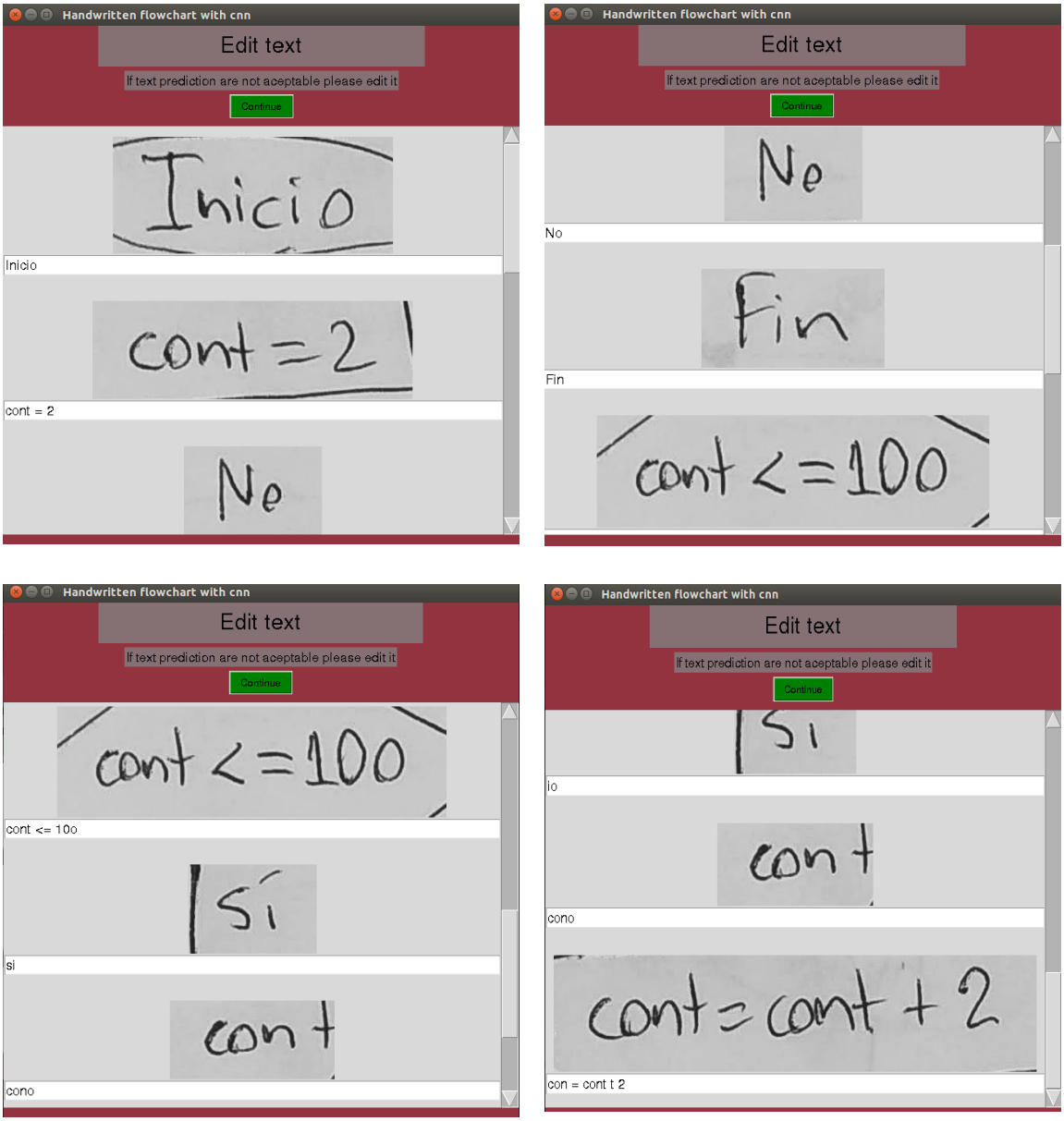


Figura 42. Reconociendo “Imprimir pares” con *continual learning*: Editando el texto. Fuente: Elaboración propia.

Aquí se muestra el resultado de la predicción del texto,como se puede observar las predicciones son más acertadas ya que el modelo usado en este ejemplo ya se le han aplicado varios procesos de “Continual learning”, aunque aún se observan algunos errores

como “cont <= 100” , “cont” , “cont = cont +2” no se predijo correctamente. Al corregirse el texto se presiona el botón de Continue.

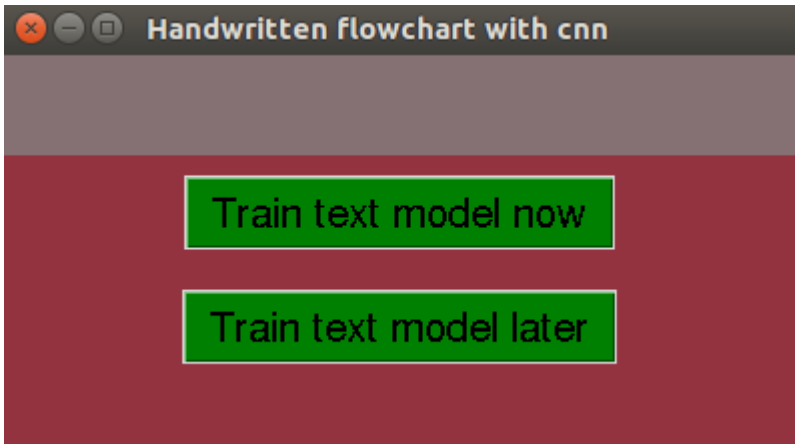


Figura 43. Reconociendo “Imprimir pares” con *continual learning*: entrenar o mostrar. Fuente: Elaboración propia.

Al presionar el botón de “Train text model now” empieza el proceso de entrenamiento.

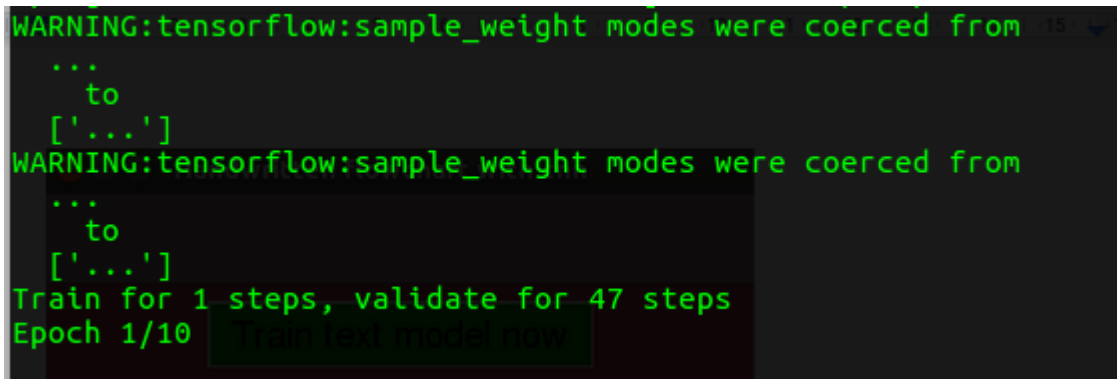


Figura 44. Reconociendo “Imprimir pares” con *continual learning*: Entrenando. Fuente: Elaboración propia.

Una vez terminado aparece la ventana de resultados usando los texto que se editaron.

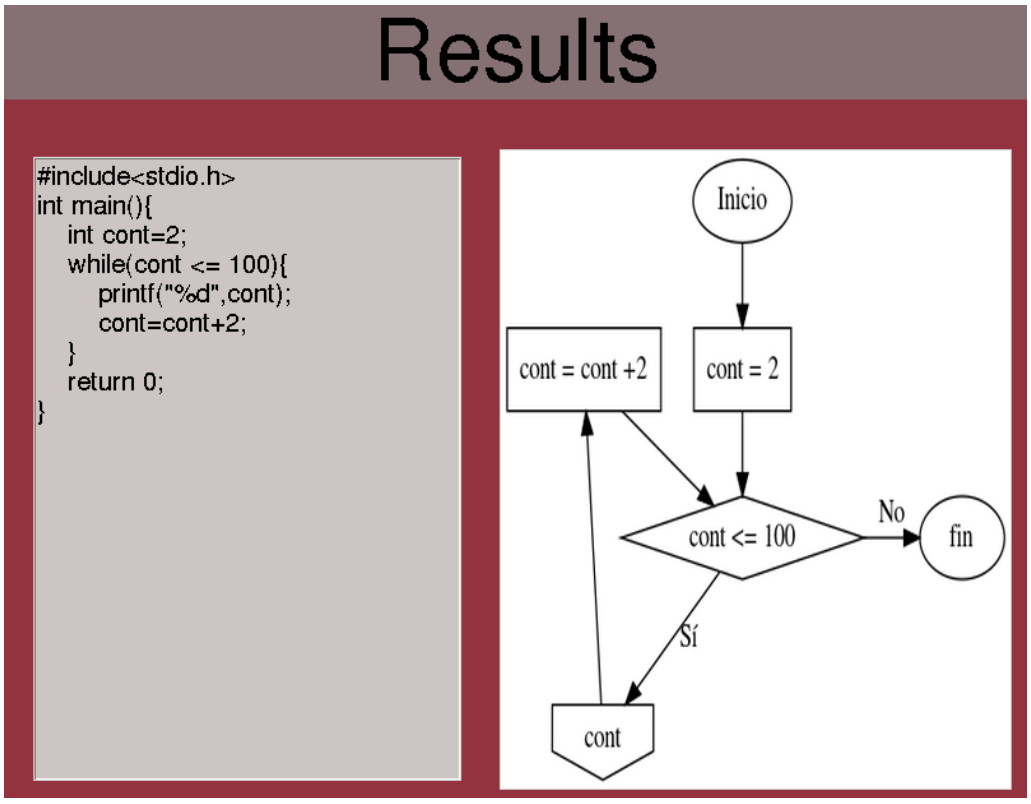


Figura 45. Reconociendo "Imprimir pares" con *continual learning*: Resultado. Fuente: Elaboración propia.

Entrenando el modelo de figuras y conectores

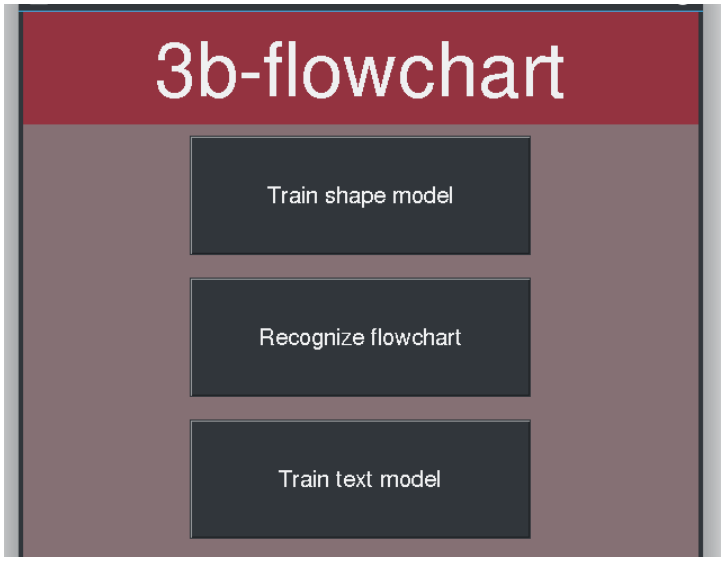


Figura 46. Entrenando modelo de figuras: Ventana principal. Fuente: Elaboración propia.

Ejecutando el script handler.py aparecerá la ventana principal donde se dará click en el primero botón para entrenar un nuevo modelo.

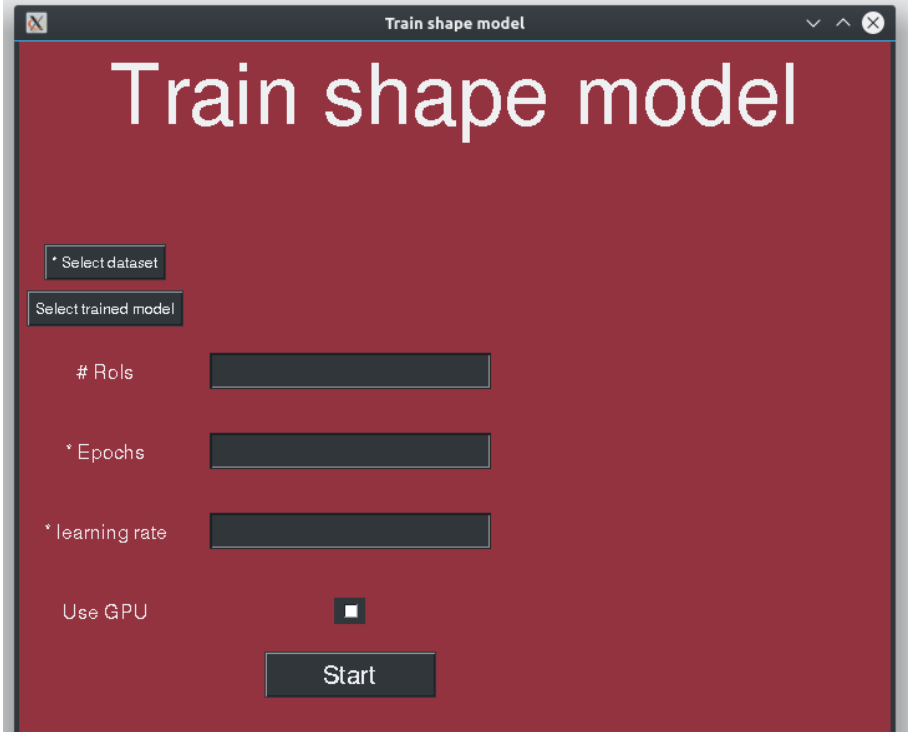


Figura 47. Entrenando modelo de figuras: Formulario para entrenar. Fuente: Elaboración propia.

Esta ventana contiene un formulario donde se puede elegir la ruta del dataset con el que se va entrenar, el modelo pre-entrenado para cargar de ahí los pesos, el número de regiones de interés a considerar dadas las detecciones del RPN (por defecto toma 32), número de épocas que durará el entrenamiento, la tasa de aprendizaje, un check box para elegir usar la GPU o no y el botón para comenzar el entrenamiento. Las entradas con un asterisco son obligatorias.

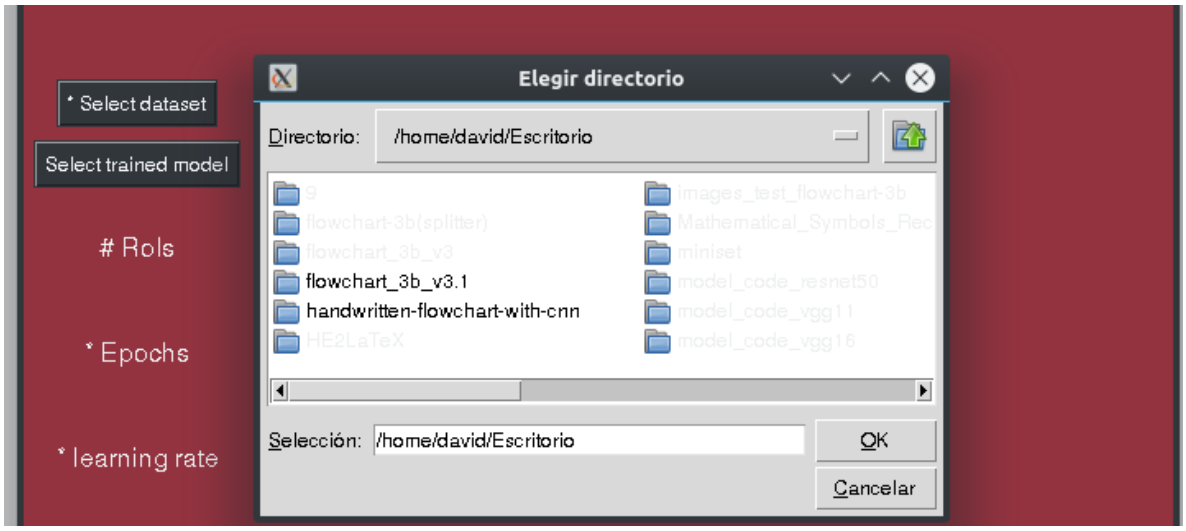


Figura 48. Entrenando modelo de figuras: Elegir directorio del dataset. Fuente: Elaboración propia.

Aquí se elige el directorio donde se encuentra el dataset, en este caso “flowchart_3b_v3.1”.

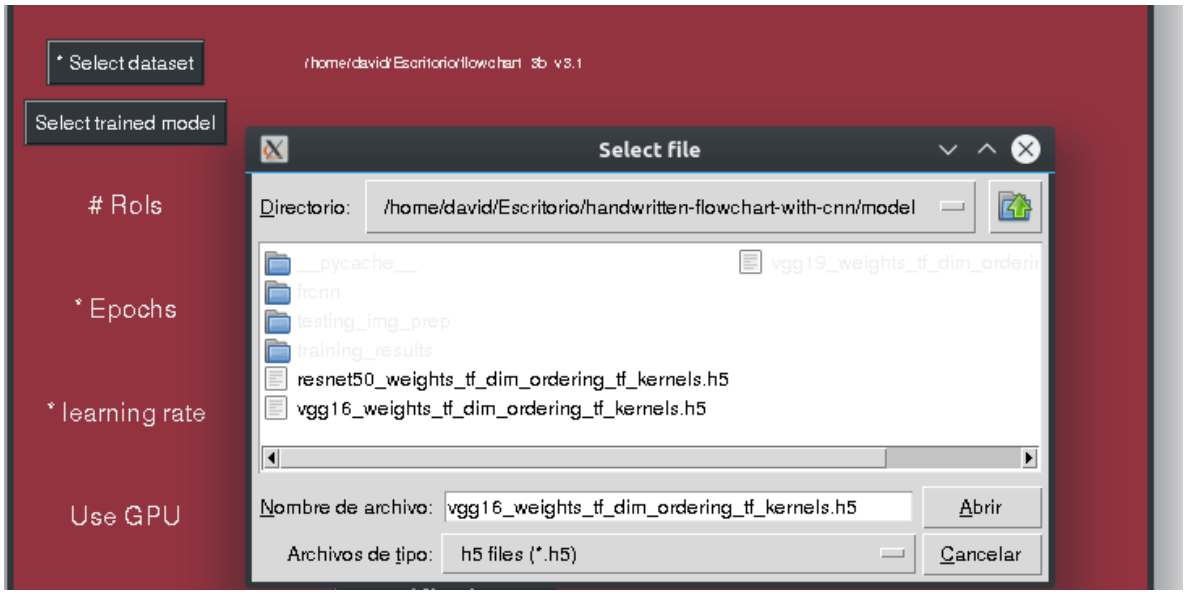


Figura 49. Entrenando modelo de figuras: Elegir archivo de pesos. Fuente: Elaboración propia.

Al darle al botón “Select trained model” aparecerá una ventana para elegir el archivo de pesos de un modelo pre-entrenado, para inicializar así los pesos de nuestro modelo antes de comenzar el proceso iterativo del entrenamiento.

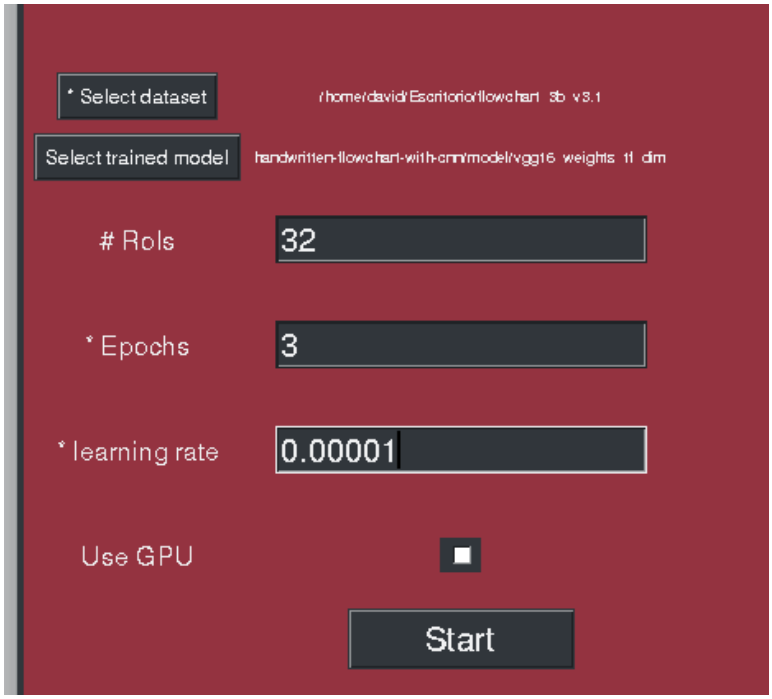


Figura 50. Entrenando modelo de figuras: Formulario completado. Fuente: Elaboración propia.

En esta captura se puede observar que se han completado todos los campos de entrada, se ha elegido no usar GPU ya que se está ejecutando en una computadora sin GPU. Ahora, sigue darle en el botón “Start” para comenzar el entrenamiento en un proceso independiente en una Terminal, esto con el fin de poder cerrar la GUI y se consuman menos recursos.

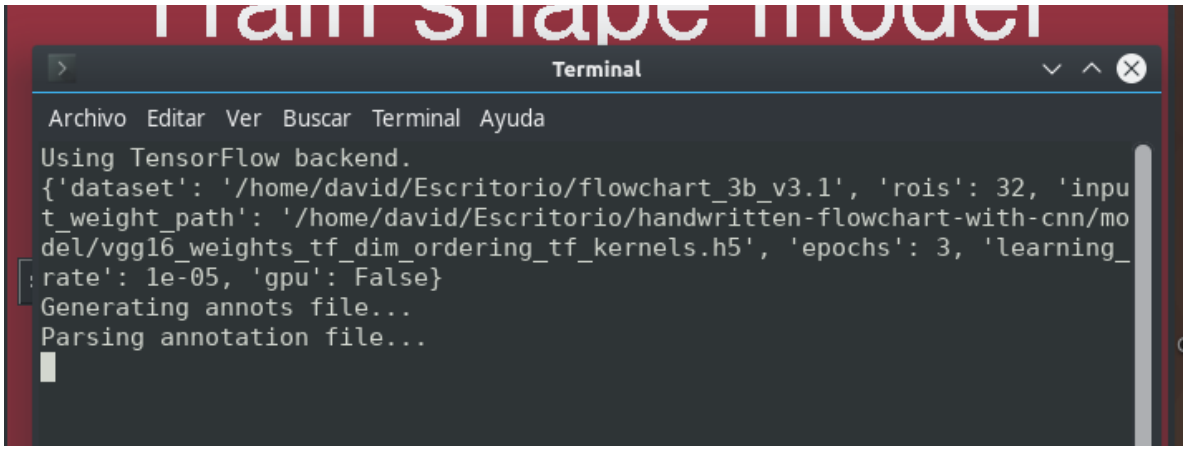


Figura 51. Entrenando modelo de figuras: Cargando información del archivo de anotaciones. Fuente: Elaboración propia.

Ya arrancada la Terminal del entrenamiento se comienzan a cargar los datos del dataset para generar el archivo de anotaciones y enseguida recuperar la información del mismo ("Parsing annotation file"). Cuando se finaliza se muestra la cantidad de objetos por cada clase y el número de clases encontradas, ver parte superior de la siguiente captura.

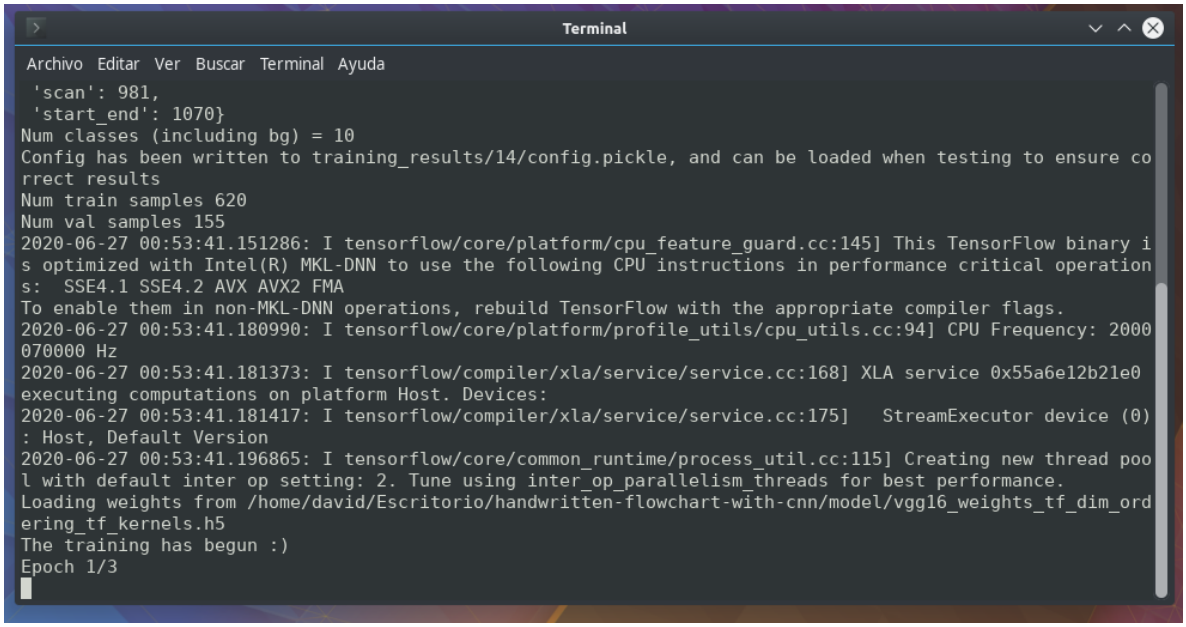


Figura 52. Entrenando modelo de figuras: Ciclo de entrenamiento ha comenzado. Fuente: Elaboración propia.

En este punto el proceso del entrenamiento ha comenzado, justo antes se han cargado los pesos del modelo pre-entrenado que se seleccionó (“Loading weights from ...”).

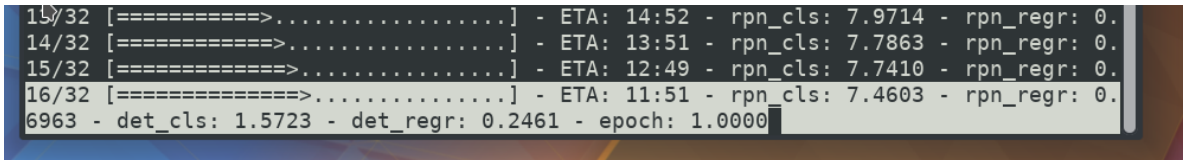


Figura 53. Entrenando modelo de figuras: Iteraciones de una época. Fuente: Elaboración propia.

Se realizan 32 iteraciones por cada época, y se busca optimizar (minimizar) la pérdida del clasificador del rpn (“rpn_cls”), el regresor del rpn (“rpn_regr”), clasificador del detector (“det_cls”) y regresor del detector (“det_regr”). ETA es la abreviación de “Estimated Time of Arrival”.

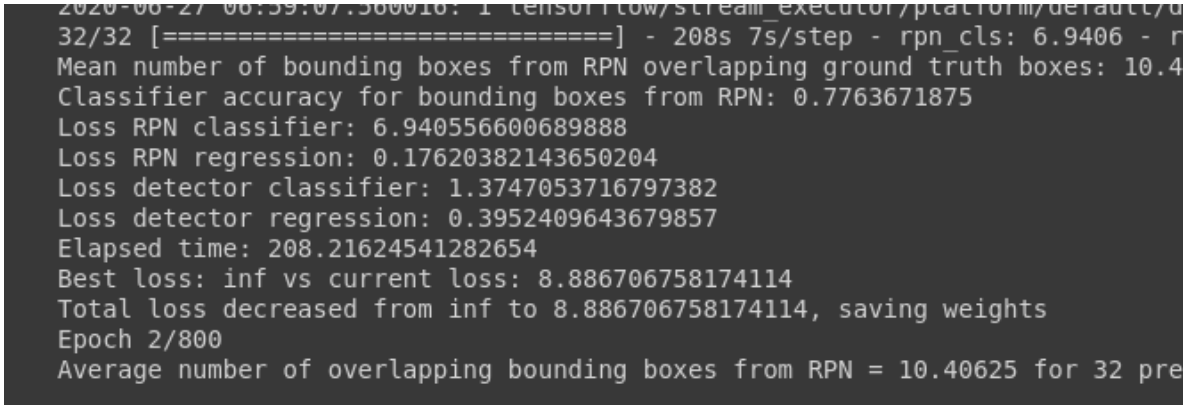


Figura 54. Entrenando modelo de figuras: Iteraciones completadas de una época. Fuente: Elaboración propia.

Una vez se completan las 32 iteraciones de una época, si las pérdidas sumadas del RPN y detector son menores a la mejor pérdida almacenada se actualizará la pérdida almacenada con dicha suma, ya que se ha encontrado un óptimo local mejor al anterior que bien podría ser el óptimo global; y de la misma manera se guardarán los pesos dado que se ha encontrado un mejor modelo. En la figura 54 el valor con el que se compara la suma es “inf”, debido a que apenas se completó la primer época, “inf” que es un valor relativamente

grande (una constante del paquete NumPy) con el que se ha inicializado antes de comenzar el ciclo del entrenamiento.

Algunos resultados obtenidos

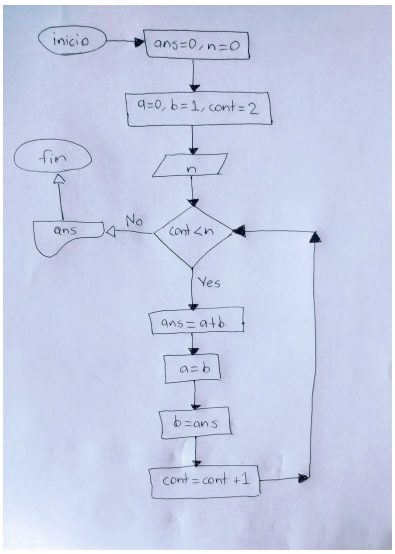
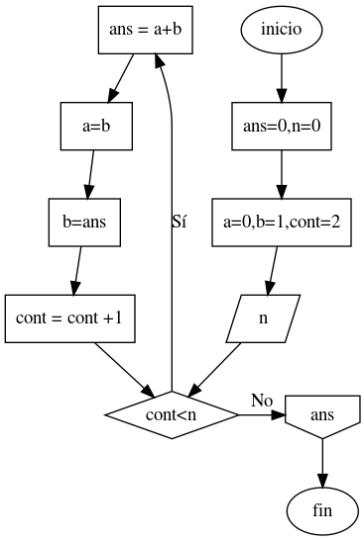
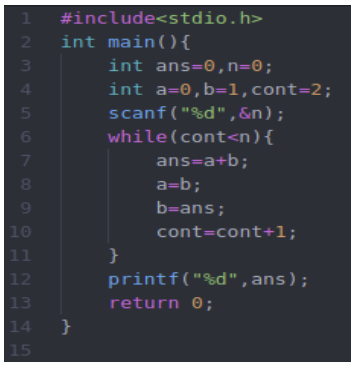
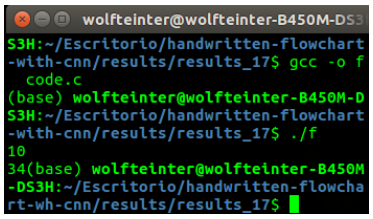
Diagrama por analizar	Diagrama digital generado	Código generado
		
		Resultado de la compilación
		

Tabla 11. Resultado completo para n-ésimo término de la sucesión de Fibonacci. Fuente: Elaboración propia.

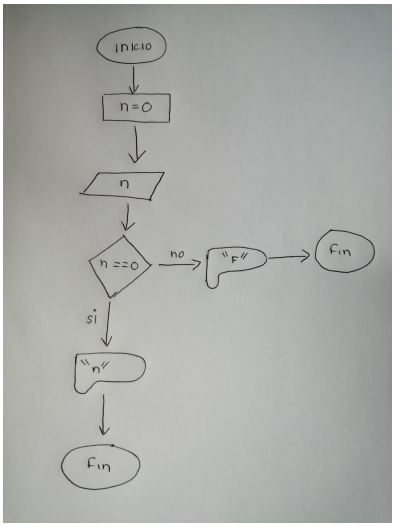
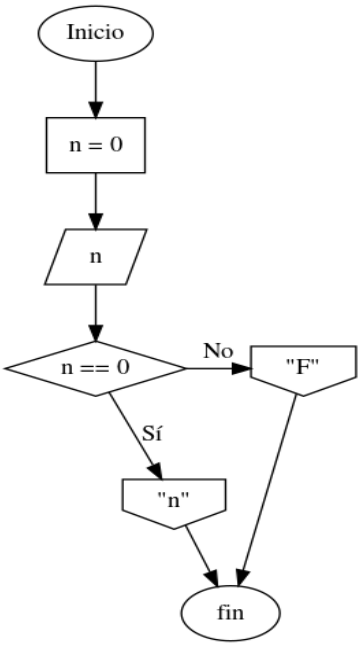
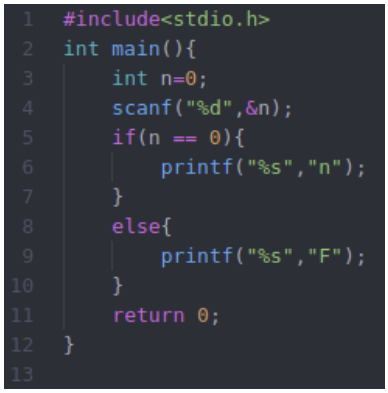
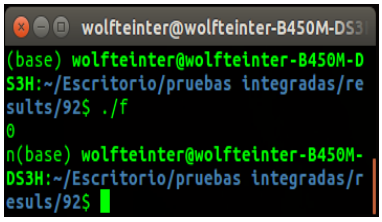
Diagrama por analizar	Diagrama digital generado	Código generado
		
		Resultado de la compilación
		

Tabla 12. Resultado completo para “Comparaciones”. Fuente: Elaboración propia.

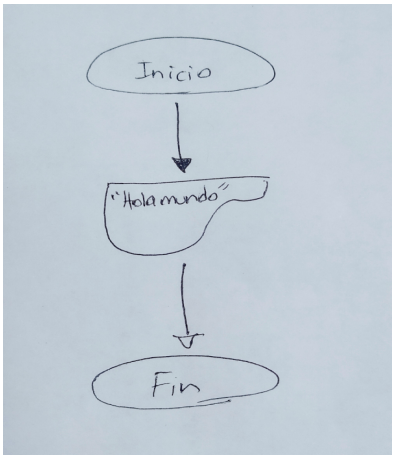
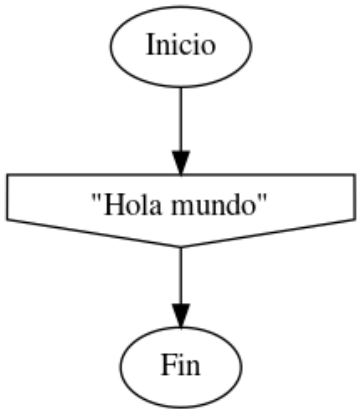
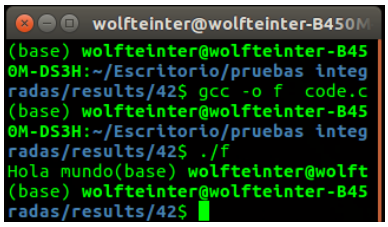
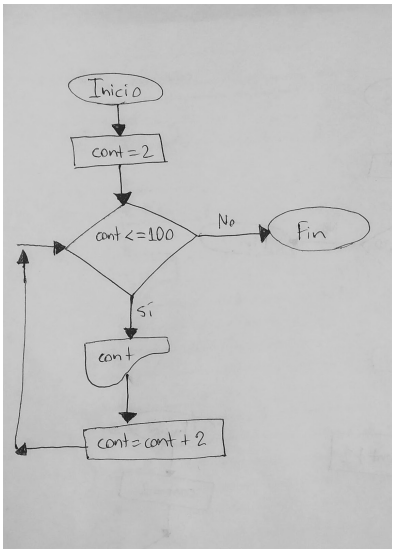
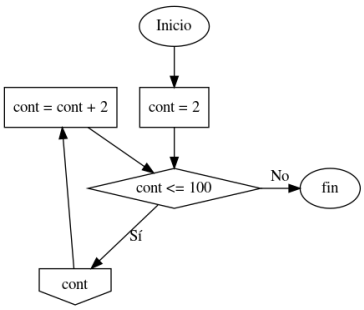
Diagrama por analizar	Diagrama digital generado	Código generado
		
		Resultado de compilación
		

Tabla 13. Resultado completo para el “Hola mundo”. Fuente: Elaboración propia.

<table>
<tr><th>Diagrama por analizar</th><th>Diagrama digital generado</th><th>Código generado</th></tr>
<tr>
<td> 
</td>
<td> 
</td>
<td>

```c
#include<stdio.h>
int main(){
    int cont=2;
    while(cont <= 100){
        printf("%d",cont);
        cont=cont+2;
    }
    return 0;
}
```

Resultado de compilación

```
wolfteinter@wolfteinter-B450M-DS3
-with-cnn/results/results_23$ gcc -o f
  code.c
(base) wolfteinter@wolfteinter-B450M-D
S3H:~/Escritorio/handwritten-flowchart
-with-cnn/results/results_23$ ./f
2468101214161820222426283032343638404244464850525456586062646668707274767880828486889092949698100(base) wolfteinte
r@wolfteinter-B450M-DS3H:~/Escritorio/
handwrith-cnn/results/results_23$
```

</td>
</tr>
</table>

Tabla 14. Resultado completo para impresión de pares hasta el 100. Fuente: Elaboración propia.

4. Seguimiento al plan de pruebas

Pruebas unitarias

Preprocessor		EPU 001
Descripción		
Verificar si los algoritmos para preparar la imagen generan la salida correcta.		
Versión	Argumentos	Status
V1	No se implementó	n/a
V2	Imagen, algoritmo a aplicar (convertir a escala de grises, enmascaramiento de enfoque, y preparación para detección de texto).	3/3

Tabla 15. Prueba unitaria 001. Fuente: Elaboración propia.

Node		EPU 008
Descripción		
Verificar si el tipo de nodo que se construye es texto, figura o conector.		
Versión	Argumentos	Status
V1	Coordenadas, texto, clase	5/5
V2	Coordenadas, texto, clase	5/5

Tabla 16. Prueba unitaria 008. Fuente: Elaboración propia.

Handler		EPU 016
Descripción		
Comprobar el buen funcionamiento de la ventana de start_gui.		
Versión	Argumentos	Status
V1	Oprimir botón "Entrenar" y “Reconocer diagrama de flujo".	2/2
V2	Oprimir botón "Entrenar", “Reconocer diagrama de flujo" y ”Entrenar modelo de texto”.	3/3

Tabla 17. Prueba unitaria 016. Fuente: Elaboración propia.

Handler		EPU 017
Descripción		
Comprobar el buen funcionamiento de la ventana train_model_gui.		
Versión	Argumentos	Status
V1	dataset_path, num_rois, pre-trained_model_path, num_epochs, learning_rate, use_gpu.	6/6
V2	dataset_path, num_rois, pre-trained_model_path, num_epochs, learning_rate, use_gpu.	6/6

Tabla 18. Prueba unitaria 017. Fuente: Elaboración propia.

Handler		EPU 018
Descripción		
Comprobar el correcto funcionamiento de la ventana "recognize_gui".		
Versión	Argumentos	Status
V1	Carpeta de resultado de entrenamiento, imagen, número de RoIs, use GPU.	5/5
V2	dataset_path, num_rois, pre-trained_model_path, num_epochs, learning_rate, use_gpu.	5/5

Tabla 19. Prueba unitaria 018. Fuente: Elaboración propia.

Handler		EPU 019
Descripción		
Comprobar el buen funcionamiento de la ventana edit_text.		
Versión	Argumentos	Status
V2	Corregir textos si es necesario	2/2

Tabla 20. Prueba unitaria 019. Fuente: Elaboración propia.

Handler		EPU 020
Descripción		
Comprobar el buen funcionamiento de la ventana train_or_show		
Versión	Argumentos	Status
V2	Oprimir botones “entrenar modelo de texto ahora” y entrenar modelo de texto después	2/2

Tabla 21. Prueba unitaria 020. Fuente: Elaboración propia.

Graph		EPU 011
Descripción		
Verificar el correcto funcionamiento de la construcción del grafo.		
Versión	Argumentos	Status
V1	Lista de Nodos de texto, lista de de nodos de figuras	3/3
V2	Lista de Nodos de texto, lista de de nodos de figuras	3/3

Tabla 22. Prueba unitaria 011. Fuente: Elaboración propia.

Code Generator		EPU 014
Descripción		
Verificar que se genera el código fuente en C.		
Versión	Argumentos	Status
V1	Grafo dirigido correcto, lista de nodos	0/3, 2/3, 3/3
V2	Grafo dirigido correcto, lista de nodos	3/3

Tabla 23. Prueba unitaria 014. Fuente: Elaboración propia.

Flowchart Generator		EPU 013
Descripción		
Comprobar el buen funcionamiento de la función generate_flowchart.		
Versión	Argumentos	Status
V1	Grafo dirigido correcto	9/9
V2	Grafo dirigido correcto	13/13

Tabla 24. Prueba unitaria 013. Fuente: Elaboración propia.

Shape Model		EPU 005
Descripción		
Verificar que se obtiene la referencia al dataset y de éste se indexan los datos de entrenamiento y validación.		
Versión	Argumentos	Status
V1	dataset_path, num_rois, horizontal_flips, vertical_flips, num_epochs, learning_rate, use_gpu	3/3
V2	dataset_path, num_rois, horizontal_flips, vertical_flips, num_epochs, learning_rate, use_gpu	4/4

Tabla 25. Prueba unitaria 005. Fuente: Elaboración propia.

Shape Model		EPU 006
Descripción		
Verificar que el error de la primer época a la última disminuye y la eficacia aumenta.		
Versión	Argumentos	Status
V1	dataset_path,num_rois,horizontal_flips,vertical_flips,num_epochs ,learning_rate,use_gpu,reporte: results_path	5/5
V2	dataset_path,num_rois,horizontal_flips,vertical_flips,num_epochs ,learning_rate,use_gpu,reporte: results_path	7/7

Tabla 26. Prueba unitaria 006. Fuente: Elaboración propia.

Shape Classifier		EPU 004
Descripción		
Verificar el comportamiento de la función "predict(image)".		
Versión	Argumentos	Status
V1	imagen,ruta de carpeta de resultados.	3/3
V2	imagen,ruta de carpeta de resultados.	13/13

Tabla 27. Prueba unitaria 004. Fuente: Elaboración propia.

Text Classifier		EPU 007
Descripción		
Prueba de unidad para la función de clasificación de texto.		
Versión	Argumentos	Status
V1	imagen	14/22
V2	imagen	4/6

Tabla 28. Prueba unitaria 007. Fuente: Elaboración propia.

Pruebas integradas

Handler y preprocessor		EPI 001
Descripción		
Comprobar la integridad del manejador con el preprocesador.		
Versión	Argumentos	Status
V1	NO SE USÓ EL MÓDULO DE PREPROCESAMIENTO	N/A
V2	Oprimir botón "Reconocer diagrama de flujo",Oprimir botón "Seleccionar imagen",Oprimir botón de "Seleccionar",imagen, Oprimir botón "Predecir".	9/9

Tabla 29. Prueba de integración 001. Fuente: Elaboración propia.

Handler, Preprocessor y Shape Classifier		EPI 003
Descripción		
Manejador con preprocesamiento y clasificador de figuras.		
Versión	Argumentos	Status
V1	Oprimir botón "Recognize flowchart", Modelo/9, flowchart_3b_model.hdf5,Oprimir botón "Select image",Oprimir botón "Abrir",imagen,Usar GPU,#	9/9

	RoIs,Oprimir botón "Predict".	
V2	Oprimir botón "Recognize flowchart", Modelo/9, flowchart_3b_model.hdf5,Oprimir botón "Select image",Oprimir botón "Abrir",imagen,Usar GPU,# RoIs,Oprimir botón "Predict".	9/9

Tabla 30. Prueba de integración 003. Fuente: Elaboración propia.

Handler, Preprocessor, Shape Classifier y Text Classifier		EPI 004
Descripción		
Comprobar la integridad del módulo Handler con el Preprocessor, Shape Classifier y Text Classifier.		
Versión	Argumentos	Status
V1	Oprimir botón "Recognize flowchart",Carpeta del modelo 9, flowchart_3b_model.hdf5,Oprimir botón "Select image",Oprimir botón "Abrir",imagen,Usar GPU,# RoIs,Oprimir botón "Predict"	9/9
V2	Oprimir botón "Recognize flowchart",Carpeta del modelo 9, flowchart_3b_model.hdf5,Oprimir botón "Select image",Oprimir botón "Abrir",imagen,Usar GPU,# RoIs,Oprimir botón "Predict", editar texto si es necesario, oprimir botón “Continue”	3/4

Tabla 31. Prueba de integración 004. Fuente: Elaboración propia.

Handler, Preprocessor, Shape Classifier, Text Classifier y Graph.		EPI 005
Descripción		
Comprobar la integridad del módulo Handler con el Preprocessor, Shape Classifier, Text Classifier y Graph.		
Versión	Argumentos	Status
V1	Oprimir botón "Recognize flowchart",Modelo 9, flowchart_3b_model.hdf5,Oprimir botón "Select image",Oprimir botón "Abrir",imagen,Usar GPU,# RoIs,Oprimir botón "Predict"	0/9
V2	Oprimir botón "Recognize flowchart",Modelo 9, flowchart_3b_model.hdf5,Oprimir botón "Select image",Oprimir botón "Abrir",imagen,Usar GPU,# RoIs,Oprimir botón "Predict", editar texto si es necesario,	3/4

	oprimir botón “Continue”.	

Tabla 32. Prueba de integración 005. Fuente: Elaboración propia.

Handler, Preprocessor Shape Classifier, Text Classifier, Graph y Code Generator		EPI 007
Descripción		
Comprobar la integridad del módulo Handler con el Preprocessor, Shape Classifier, Text Classifier, Graph y Code Generator.		
Versión	Argumentos	Status
V1	Oprimir botón "Recognize flowchart",Modelo 9, flowchart_3b_model.hdf5,Oprimir botón "Select image",Oprimir botón "Abrir",imagen,Usar GPU,# RoIs,Oprimir botón "Predict"	0/9
V2	Oprimir botón "Recognize flowchart",Modelo 9, flowchart_3b_model.hdf5,Oprimir botón "Select image",Oprimir botón "Abrir",imagen,Usar GPU,# RoIs,Oprimir botón "Predict", editar texto si es necesario, oprimir botón “Continue”	3/4

Tabla 33. Prueba de integración 007. Fuente: Elaboración propia.

Handler, Preprocessor, Shape Classifier, Text Classifier, Graph, Code Generator y Flowchart Generator.		EPI 008
Descripción		
Comprobar la integridad del módulo Handler con el Preprocessor, Shape Classifier, Text Classifier, Graph, Code Generator y Flowchart Generator.		
Versión	Argumentos	Status
V1	Oprimir botón "Recognize flowchart",Modelo 9, flowchart_3b_model.hdf5,Oprimir botón "Select image",Oprimir botón "Abrir",imagen,Usar GPU,# RoIs,Oprimir botón "Predict"	0/9
V2	Oprimir botón "Recognize flowchart",Modelo 9, flowchart_3b_model.hdf5,Oprimir botón "Select image",Oprimir botón "Abrir",imagen,Usar GPU,# RoIs,Oprimir botón "Predict", editar texto si es necesario, oprimir botón “Continue”.	3/4

Tabla 34. Prueba de integración 008. Fuente: Elaboración propia.

Handler y Shape Model		EPI 009
Descripción		
Comprobar la integridad del módulo Handler con el Shape Model.		
Versión	Argumentos	Status
V1	dataset_path,num_rois,pre-trained_model_path,num_epochs,learning_rate,use_gpu	5/5
V2	dataset_path,num_rois,pre-trained_model_path,num_epochs,learning_rate,use_gpu	5/5

Tabla 35. Prueba de integración 009. Fuente: Elaboración propia.

Pruebas de sistema

Prueba de sistema		EPS 001
Descripción		
Determinar si es posible generar el código en C y diagrama digital a partir de una foto de un diagrama de flujo trazado a mano.		
Versión	Argumentos	Status
V1	Oprimir botón "Reconocer diagrama de flujo",Oprimir botón "Seleccionar imagen",Oprimir botón de "Seleccionar",imagen,Oprimir botón "Predecir".	2/12
V2	Oprimir botón "Reconocer diagrama de flujo",Oprimir botón "Seleccionar imagen",Oprimir botón de "Seleccionar",imagen,Oprimir botón "Predecir",editar texto si es necesario, oprimir botón “Continue”	5/12

Tabla 36. Prueba de sistema 001. Fuente: Elaboración propia.

Prueba de sistema		EPS 002
Descripción		
Determinar si se puede entrenar la CNN para figuras y textos desde la GUI.		
Versión	Argumentos	Status
V1	dataset_path,num_rois,pre-trained_model_path,num_epochs,learning_rate,use_gpu	5/5
V2	dataset_path,num_rois,pre-trained_model_path,num_epochs,learning_rate,use_gpu	5/5

Tabla 37. Prueba de sistema 002. Fuente: Elaboración propia.

5. Entrega o liberación

El sistema se ha publicado en un repositorio en GitHub bajo la licencia MIT, este repositorio se creó desde el inicio del proyecto, a fin de ayudar en el control de versiones y trabajo en equipo. En este enlace https://github.com/dbetm/handwritten-flowchart-with-cnn puede ser encontrado en su versión final con su respectivo README en el cual de muestra una descripción general, forma de configurar para poder probarlo, cómo usarlo y algunas capturas de los resultados obtenidos, a fin de que cualquier persona interesada en el tema pueda experimentar con él.

Conclusiones y recomendaciones

Ahora que pertenece a la realidad el desarrollo del proyecto se puntualiza el alcance de los objetivos definidos en un modelo del mundo inicial para el proyecto, a fin de cuentas se ha perseguido un objetivo general (final) a través del cumplimiento de objetivos particulares (instrumentales).

El objetivo general se ha cumplido al haber diseñado e implementado un canal de reconocimiento (pipeline) con la capacidad de reconocer los componentes de un diagrama de flujo trazado a mano, a fin de generar la salida, en este caso el código fuente en C y el diagrama de flujo reconstruido en forma de imagen digital. Para alcanzarlo, fue necesario hacer modificaciones a la propuesta inicial del pipeline, satisfaciendo cada uno de los objetivos particulares y así demostrar el cumplimiento de la hipótesis al utilizar redes neuronales convolucionales para resolver el problema de reconocer cada uno de los componentes (figuras, conectores y texto) de un diagrama de flujo bajo el enfoque *off-line* [8]. Señalar el resultado obtenido respecto al objetivo general implica señalar los resultados obtenidos para cada uno de los objetivos particulares.

Para alcanzar el objetivo particular del reconocimiento de figuras, texto y conectores plasmados en una imagen digital de un diagrama de flujo trazado a mano, fue necesario dividir este mismo objetivo en dos subproblemas a resolver, así que se utilizaron dos modelos que emplean redes neuronales convolucionales para resolver cada uno, las métricas de cada uno de ellos resolviendo la tarea para la que fueron entrenados demuestran en qué grado son buenos para ello, dichas métricas son discutidas en el informe de resultados que se encuentra en el apéndice P (P.1). El mayor reto encontrado en la tarea de detección de texto fue la clasificación de los caracteres, siendo esto un paso crucial para así generar la salida del canal de reconocimiento, implicó agregar un proceso más al pipeline para aprobar el texto localizado en la imagen del diagrama de flujo y a fin de continuar mejorando la precisión del modelo de clasificación al implementar una técnica que hace

mejorar con el tiempo la precisión en la clasificación, por otra parte, sobre el modelo de detección de figuras y conectores, los mayores retos fueron la recuperación de los datos de entrenamiento y el encontrar la configuración para tener una métrica de reconocimiento suficiente para reconocer la mayoría de los diagramas de flujo probados, se solucionaron ambos retos al agregar más datos al conjunto de imágenes recuperadas inicialmente y el realizar diversas pruebas modificando parámetros propios del entrenamiento y realizar las evaluaciones de los modelos entrenados, respectivamente.

Usar redes neuronales convolucionales para la tarea de detección (localización y clasificación) ha tenido un resultado positivo, ya que fue necesario para detectar los componentes del diagrama de flujo tal como se ha explicado anteriormente.

El usar análisis de gramática para verificar la estructura del diagrama de flujo ha sido importante al momento de revisar si el flujo que se sigue en un diagrama es permisible, ayudando así a detectar cuando un diagrama tiene una estructura que no es la correcta o ha sido detectado de alguna manera que resulta incorrecto para así impedir la finalización del flujo de información en el pipeline a fin de evitar errores desconocidos en la generación del código y diagrama en forma digital.

Por la parte de generar el código fuente, el resultado es que en la mayoría de los casos donde el flujo de entrada es correcto, se ha sido capaz de generar el archivo con la implementación en código C del algoritmo que se representa en el diagrama de entrada, uno de los mayores problemas ha sido el considerar más casos posibles con fin de que sean una mayoría los casos donde se hace la generación correctamente, entendiendo así que el algoritmo se puede hacer más robusto al ir considerando más casos.

Para el objetivo particular de digitalizar el diagrama de flujo, esto es, reconstruir los elementos del diagrama de flujo y generar una imagen digital, se han tenido resultados en los que en la mayoría de los casos se ha generado correctamente.

Sobre las ventajas resultantes del desarrollo del proyecto se puede mencionar el aporte de un nuevo conjunto de datos para el estudio del problema bajo el enfoque *off-line* [8], de igual manera la propuesta e implementación de la solución al problema mediante el pipeline resultante. Además, la demostración práctica al solucionar el problema de recuperar la información para así generar una salida que podría integrarse en algún contexto, en este caso, la generación del código fuente y reconstrucción del diagrama del diagrama, respecto a esto último destacando el empleo de una técnica que permite la mejora continua de la implementación de la clasificación de los caracteres del texto, haciendo una analogía con la capacidad humana del aprendizaje, donde gracias a las sinapsis que conectan las neuronas que forman el cerebro pueden almacenar en algún factor más veces la información contenida en el ADN con el que se nace.

Los aspectos desafortunados encontrados en el proyecto realizado son principalmente el tiempo que tarda en completarse el reconocimiento completo de los componentes de un diagrama de flujo, aunque depende de los recursos de hardware de la computadora donde se prueba, el proceso subyacente para optimizarse no se ha trabajado, éste es la carga en memoria principal de los modelos de detección. Por otra parte, el hecho de que no deja de ser un proceso probabilístico para la solución del problema, lo que indica que siempre habrá casos donde no se obtengan un resultado satisfactorio.

Como propuestas de trabajo a futuro son el mejorar el aspecto estético (sobre la estructura del grafo) en el resultado de la reconstrucción del diagrama de flujo reconocido, la optimización en el flujo del canal, en sí, el uso de los modelos, como se ha mencionado antes, este es un aspecto que podría mejorarse y finalmente, el agregar más símbolos al conjunto permitido para la construcción del diagrama de flujo para otorgar más flexibilidad a la hora de dibujar un diagrama de flujo a reconocerse con el sistema.

Para terminar, las lecciones aprendidas desarrollando el proyecto son sobre el conocimiento obtenido en camino a alcanzar cada uno de los objetivos planteados, ya que ha implicado habilidades de un ingeniero de software, ingeniero de aprendizaje automático y de datos, concretamente, que un proyecto de inicio a fin como ha sido éste implica el conocimiento de diversas ramas; por otra parte la importancia del empleo de una metodología para no perder de vista el cómo hacer en cada una de las etapas del proceso de desarrollo. Cabe mencionar que se encontraron dificultades eligiendo las metodologías para encajar la orientación de proyecto de investigación con el desarrollo de software, representando esto algunos retrasos que finalmente se fueron tratando de resolver.

Fuentes de consulta

[1] K. Grauman and B. Leibe, Visual object recognition. [San Rafael, Calif.]: Morgan & Claypool, 2011.

[2] S. THEODORIDIS and K. KOUTROUMBAS, PATTERN RECOGNITION, 2nd ed. San Diego: ELSEVIER, 2003, pp. 1-2,77-87.

[3] Y. Jiang, F. Yang, H. Zhu, D. Zhou, and X. Zeng, "Nonlinear CNN: improving CNNs with quadratic convolutions," Neural Computing and Applications, 2019.

[4] Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105.

[5] Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[6] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D,Erhan D, Vanhoucke V, Rabinovich A et al (2015) Going deeper with convolutions. In: CVPR.

[7] He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778.

[8] W. Szwoch and M. Mucha, "Recognition of Hand Drawn Flowcharts" Advances in Intelligent Systems and Computing Image Processing and Communications Challenges 4, vol. 184, pp. 65–72, 2013.

[9] M. Rusiñol, L.-P. D. L. Heras, and O. R. Terrades, “Flowchart recognition for non-textual information retrieval in patent search” Information Retrieval, vol. 17, no. 5-6, pp. 545–562, 2013.

[10] M. Bresler, D. Průša, and V. Hlaváč, “Online recognition of sketched arrow-connected diagrams” International Journal on Document Analysis and Recognition (IJDAR), vol. 19, no. 3, pp. 253–267, 2016.

[11] C. Wang, H. Mouchère, A. Lemaitre, and C. Viard-Gaudin, “Online flowchart understanding by combining max-margin Markov random field with grammatical analysis” International Journal on Document Analysis and Recognition (IJDAR), vol. 20, no. 2, pp.

[12] J. I. Herrera Camara, "Flow2Code: from hand-drawn flowcharts to code execution", master thesis, Texas A&M University, 2017.

[13] Z. Yuan, H. Pan, and L. Zhang, “A Novel Pen-Based Flowchart Recognition System for Programming Teaching” Lecture Notes in Computer Science Advances in Blended Learning, pp. 55–64, 2008.

[14] O. Cairó Battistutti, Metodología de la programación, 3rd ed. México, D.F.: Alfaomega, 1995, pp. 3, 4-8.

[15] D. Rotman, "El verdadero poder de la IA: revolucionar nuestra forma de inventar", MIT Technology Review, 2019. [Online]. Available: https://www.technologyreview.es/s/10952/el-verdadero-poder-de-la-ia-revolucionar-nuestra-forma-de-inventar. [Accessed: 07- Sep- 2019].

[16] F. Assolini, “El software se está comiendo el mundo,” El Mundo, 19-May-2016. [Online].

Available:https://www.elmundo.es/economia/2016/05/19/573d9acf22601d21458b4666.html.

[17] M. Mccracken, et al., “A multi-national, multi-institutional study of assessment of programming skills of first-year CS students,” Working group reports from ITiCSE on Innovation and technology in computer science education - ITiCSE-WGR 01, 2001.

[18] "What is a Flowchart", Lucidchart.com. [Online]. Available: https://www.lucidchart.com/pages/what-is-a-flowchart-tutorial. [Accessed: 20- Aug- 2019].

[19] J. López García, Algoritmos y programación (Guía para docentes), 2nd ed. Fundación Gabriel Piedrahita Uribe, 2009, p. 11.

[20] "CS 230 - Convolutional Neural Networks Cheatsheet", Stanford.edu, 2018. [Online]. Available:https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks. [Accessed: 21- Aug- 2019].

[21] Aggarwal, C., 2018. Neural Networks And Deep Learning. Cham, Switzerland: Springer, pp.365-366.

[22] L. Jiao et al., "A Survey of Deep Learning-Based Object Detection," in IEEE Access, vol. 7, pp. 128837-128868, 2019, doi: 10.1109/ACCESS.2019.2939201.

[23] Faster r-cnn: Towards real-time object detection with region proposal networks
S Ren, K He, R Girshick, J Sun - Advances in neural information processing systems, 2015

[24] Aljundi, R. (2019). Continual Learning in Neural Networks. *ArXiv, abs/1910.02718*.

[25] R. Gonzalez and R. Woods, Digital image processing, 2nd ed. Pearson Prentice Hall, pp. 1-2, 88.

[26] T. N. Minh, et al., “Automated Image Data Preprocessing with Deep Reinforcement Learning,” CoRR, pp. 1, 2018.

[27] T. Acharya and A. Ray, Image processing: Principles and Applications. Wiley, 2005, pp. 109 ,110.

[28] F. Peinado and J. Sierra, *Repaso. Lenguajes formales*. Madrid, España, 2009, pp. 2.

[29] G. Rossum, Python tutorial. New York, NY: To Excel, 1999, p. 1.

[30] F. Nelli, Python data analytics, 2nd ed. Roma Italia: Apress, 2018, p. 17.

[31] D. M. Ritchie and N. Gomez Muños, El lenguaje de programacion C, 2nd ed. México: Prentice Hall, 1991, p. 1.

[32] Amaral, J. N., Buro, M., Elio, R., Hoover, J., Nikolaidis, I., Salavatipour, M., Stewart, L. y Wong, K. About Computing Science Research Methodology. p. 1

[33] R. Pressman, Ingeniería del software, 7th ed. Mc Graw Hill, 2010, p. 34.

[34] I. Sommerville and M. Alfonso Galipienso, Ingeniería del software, 7th ed. Madrid: Pearson Educación, 2005, p. 63.

[35] G. DODIG-CRNKOVIC, “Scientific Methods in Computer Science.”, pp. 3, 9, 13

[36] Feitelson, D. G. Experimental Computer Science:The Need for a Cultural Change.

[37] Gutiérrez Aranzeta, C. Introducción a la metodología experimental

[38] E. P. Sánchez Femat, “Aplicación móvil para el conteo automático e identificación preliminar de colonias de bacterias mediante reconocimiento de patrones,” thesis, DSpace Tesis IPN, 2018.

[39] V. Gómez, "Modelo Iterativo Incremental - Instinto Binario", Instinto Binario, 2016. [Online]. Available: https://instintobinario.com/modelo-iterativo-incremental/. [Accessed: 21- Aug- 2019].

[40] "Desarrollo iterativo e incremental", Proyectos Ágiles. [Online]. Available: https://proyectosagiles.org/desarrollo-iterativo-incremental/. [Accessed: 21- Aug- 2019].

[41] P. openMP, "Parallel programming in C with MPI and openMP : Quinn, Michael J. (Michael Jay) : Free Download, Borrow, and Streaming : Internet Archive", Internet Archive, 2004. [Online]. Available: https://archive.org/details/parallelprogramm0000quin. [Accessed: 22- Oct- 2019]

[42] Faster r-cnn: Towards real-time object detection with region proposal networks
S Ren, K He, R Girshick, J Sun - Advances in neural information processing systems, 2015

[43] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR, abs/1409.1556*.

[44] J. Puigcerver, "Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?," 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, 2017, pp. 67-72, doi: 10.1109/ICDAR.2017.20.

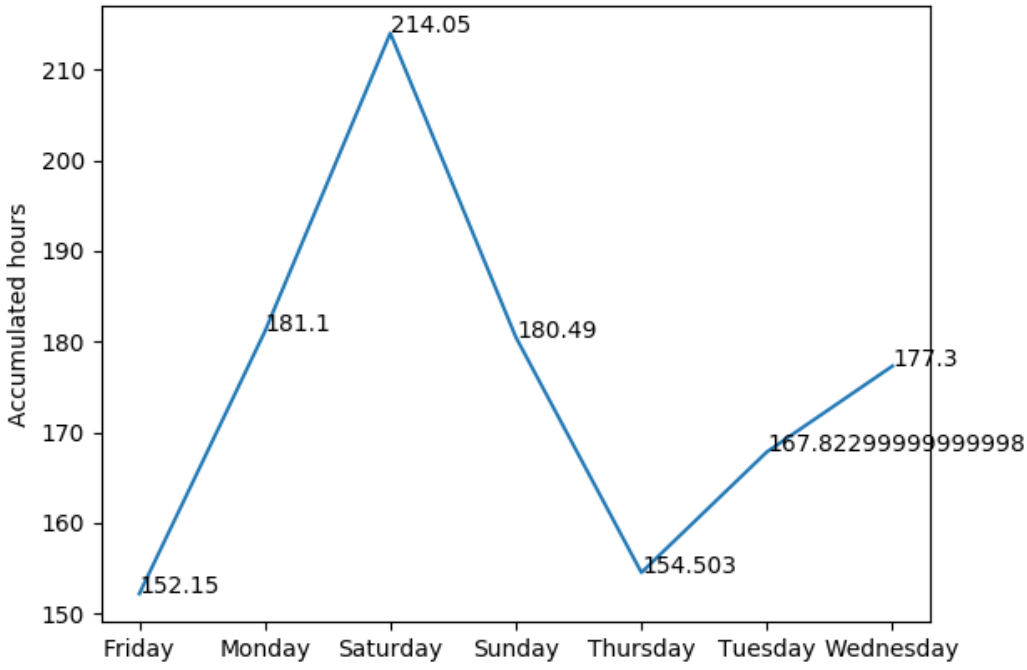
[45] U. Marti and H. Bunke. The IAM-database: An English Sentence Database for Off-line Handwriting Recognition. Int. Journal on Document Analysis and Recognition, Volume 5, pages 39 - 46, 2002.

Apéndices

Apéndice A

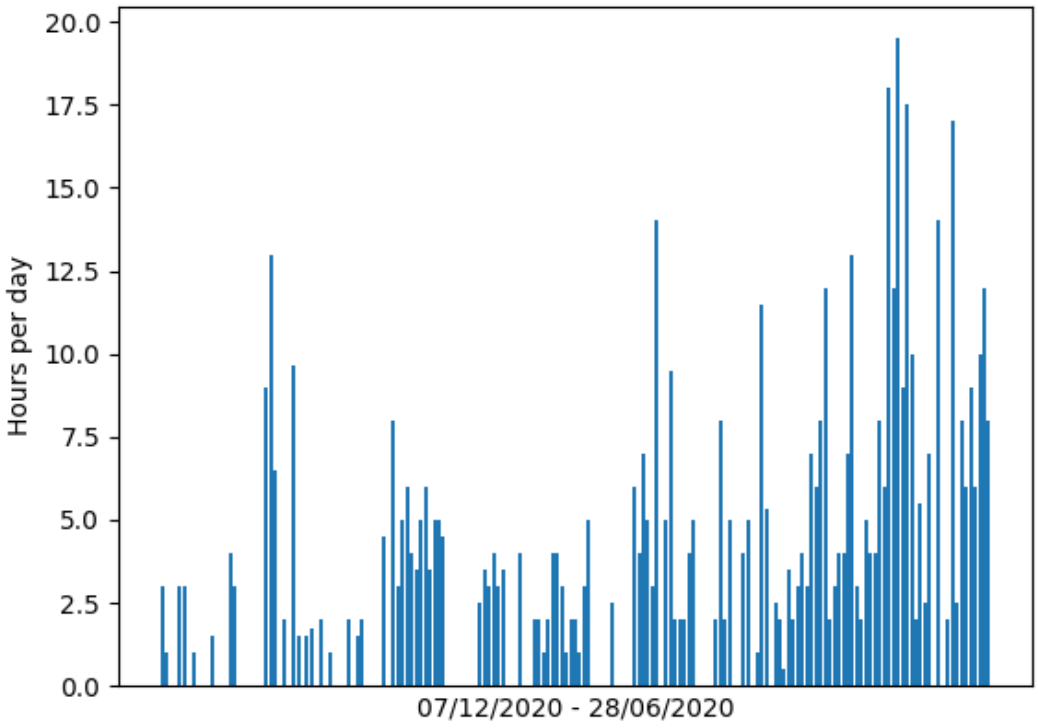
Estas gráficas fueron generadas por los registros de una bitácora de actividades que se usó para llevar el control al seguimiento del plan de proyecto.

A.1 Esfuerzo acumulado por días de la semana



Notar que los sábados fue en general el día más productivo, el jueves y viernes los menos productivos. Total de horas desde el 07 de diciembre de 2019: 1227.763.

A.2 Evolución del esfuerzo



Es posible apreciar que al final del desarrollo hubo más esfuerzo respecto al inicio del desarrollo del proyecto. Por día se promedia 6.707 horas acumuladas por los dos desarrolladores, es decir, que aproximadamente cada uno trabajó diario 3.35 horas, menor a las 4 horas que se acordaron.

Apéndice B

B.1 SRS

Especificación de Requerimientos de Software

Reconocimiento de diagramas de flujo trazados a mano usando redes neuronales convolucionales

David Betancourt Montellano
Onder Francisco Campos Garcia

Contenido

Introducción

En este documento se hace la especificación de requerimientos para el programa de software necesario para la fase de experimentación del proyecto de investigación.

Propósito

El propósito de este documento es dar a conocer los requerimientos del software para resolver el problema de reconocer diagramas de flujo trazados a mano. Asimismo, se escribe con el fin de que resulte entendible a quienes convenga el proyecto, además, el hecho en sí mismo de plasmarlo en un escrito para tenerlo presente para todos los momentos en que se requiera revisar los requerimientos especificados.

Alcance

El nombre del proyecto es “Reconocimiento de diagramas de flujo trazados a mano usando redes neuronales convolucionales”, lo que el software podrá realizar es reconocer cada una de las partes de una imagen digital de un diagrama de flujo hecho a mano y así lograr un reconocimiento global, y con ello se va a generar una versión digital del diagrama de flujo y el código fuente en lenguaje C equivalente al diagrama de flujo.

No se espera que sea una aplicación web ni móvil, sino software de inteligencia artificial en la computadora personal, el diagrama en versión digital hace referencia a una imagen digital del diagrama reconstruido que no es una versión editable para algún software de aplicación.

Los beneficios esperados es ampliar el conocimiento en el área de investigación al problema de recuperar información no textual, específicamente, de diagramas de flujo trazados a mano bajo el enfoque off-line [1] y como se ha mencionado antes se aplicará una técnica de aprendizaje profundo (área de inteligencia artificial), en este caso el uso de CNNs. La meta planteada es obtener una eficacia lo suficientemente alta para después poder generar un diagrama digital y el código fuente equivalente al algoritmo.

Definiciones, acrónimos y abreviaturas

Biblioteca: Es un conjunto de rutinas y funciones compiladas en un archivo. Otros programas pueden enlazarlas y usar esas funciones y rutinas [2].
C: Un lenguaje de programación de propósito general, nació en los laboratorios Bell de AT&T. C trabaja bajo el paradigma de programación estructurada [3].
Clase: Equivale a la generalización de un tipo específico de objetos, pero cada objeto que construimos de esa clase tendrás sus propiedades [4].
CNN: Convolutional Neural Network (Red neuronal convolucional) [5].
Colab: Colaboratory de Google es un entorno gratuito de Jupyter Notebook que no requiere configuración y que se ejecuta completamente en la nube [6].
Eficacia: Capacidad de lograr el efecto que se desea o se espera [7].
Git: Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños hasta muy grandes, con rapidez y eficiencia [8].
GitHub: Es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de computadora [9].
GNU: Es un sistema operativo de software libre, es decir, respeta la libertad de los usuarios. El sistema operativo GNU consiste en paquetes de GNU (programas publicados específicamente por el proyecto GNU) además de software libre publicado por terceras partes [10].
GPU: Graphics Processing Unit (Unidad de Procesamiento Gráfico).
GUI: Graphical User Interface (interfaz gráfica de usuario).
Linux: Un sistema operativo el cual es un conjunto de programas que le permiten interactuar con su ordenador y ejecutar otros programas [11].
No Break: También llamado fuente de alimentación ininterrumpida (UPS) es un dispositivo que permite que una computadora siga funcionando durante al menos un corto tiempo cuando se pierde la fuente de alimentación primaria. Los dispositivos UPS también brindan protección contra sobretensiones [12].
Nube: Modelo de prestación de servicios tecnológicos que permite el acceso bajo demanda y a través de internet a un conjunto de recursos compartidos configurables de modo escalable que pueden ser rápidamente asignados y liberados con una mínima gestión por parte del proveedor de internet [13].
Pipeline: Arquitectura en pipeline consiste en ir transformando un flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior [14].
Programación Orientada a Objetos (POO): La programación orientada a objetos es un modelo de la programación que utiliza objetos, ligados mediante mensajes,

para la solución de problemas,La idea central es organizar los programas a imagen y semejanza de la organización de los objetos en el mundo real [4].
Python3: Es un lenguaje de programación orientado a objetos interactivo e interpretado [15].
Sistema monolítico: Son aquellos en los que su centro es un grupo de estructuras fijas, las cuales funcionan entre sí [16].
Terminal: Máquina con teclado y pantalla mediante la cual se proporcionan datos a una computadora central o se obtiene información de ella [17].
UPIIZ: Unidad Profesional Interdisciplinaria de Ingeniería Campus Zacatecas.

Referencias

Estándares

- IEEE-STD-830-1998: Especificación de Requerimientos de software.
- En el anexo 1, se encuentra el conjunto de figuras y conectores permitidos en el diagrama de flujo a reconocer.
- En el anexo 2, se encuentra el conjunto de caracteres permitidos en el diagrama de flujo a reconocer.
- PEP-8: Guía de estilos para la codificación en Python.

Referencias

[1] W. Szwoch and M. Mucha, “Recognition of Hand Drawn Flowcharts” Advances in Intelligent Systems and Computing Image Processing and Communications Challenges 4, vol. 184, pp. 65–72, 2013.

[2] "Biblioteca (informática)", Enciclopedia Libre Universal en Español, 2005. [Online]. Available: http://enciclopedia.us.es/index.php/Biblioteca_(inform%C3%A1tica). [Accessed: 22- Oct- 2019].

[3] F. Ceballos Sierra, C/C++. Paracuellos del Jarama, Madrid: RA-MA, pp. 4-5, 2015.

[4] F. Ceballos Sierra, Java 2. México: Alfaomega, 2011.

[5] "CS 230 - Convolutional Neural Networks Cheatsheet", *Stanford.edu*, 2018. [Online].

[6] Ó. de la Fuente Sanz, "Google Colab: Python y Machine Learning en la nube - Adictos al trabajo", Adictos al trabajo, 2019. [Online]. Available: https://www.adictosaltrabajo.com/2019/06/04/google-colab-python-y-machine-learning-en-la-nube/. [Accessed: 22- Oct- 2019].

[7] "Eficacia", Dle.rae.es. [Online]. Available: https://dle.rae.es/srv/search?m=30&w=eficacia. [Accessed: 22- Oct- 2019].

[8] "Git", Git-scm.com. [Online]. Available: https://git-scm.com. [Accessed: 22- Oct- 2019].

[9] "Build software better, together", GitHub. [Online]. Available: https://github.com. [Accessed: 22- Oct- 2019].

[10] "El sistema operativo GNU y el movimiento del software libre", Gnu.org, 2019. [Online]. Available: https://www.gnu.org/home.es.html. [Accessed: 22- Oct- 2019].

[11] "1.2. ¿Qué es GNU/Linux?", Debian.org. [Online]. Available: https://www.debian.org/releases/stretch/mips/ch01s02.html.es. [Accessed: 22- Oct- 2019].

[12] M. Rouse, "What is an Uninterruptible Power Supply? - Definition from WhatIs.com", SearchDataCenter, 2019. [Online]. Available: https://searchdatacenter.techtarget.com/definition/uninterruptible-power-supply. [Accessed: 22- Oct- 2019].

[13] "Definición de computación en la nube - Diccionario del español jurídico - RAE", Diccionario del español jurídico - Real Academia Española. [Online]. Available: https://dej.rae.es/lema/computación-en-la-nube. [Accessed: 22- Oct- 2019].

[14] P. openMP, "Parallel programming in C with MPI and openMP : Quinn, Michael J. (Michael Jay) : Free Download, Borrow, and Streaming : Internet Archive", Internet Archive, 2004. [Online]. Available: https://archive.org/details/parallelprogramm0000quin. [Accessed: 22- Oct- 2019].

[15] B. Venners, "The Making of Python", Artima.com, 2003. [Online]. Available: https://www.artima.com/intv/python.html. [Accessed: 22- Oct- 2019].

[16] "Sistemas monolíticos", Los diccionarios y las enciclopedias sobre el Académico, 2012. [Online]. Available: https://enciclopedia_universal.esacademic.com/68875/Sistemas_monolíticos. [Accessed: 22- Oct- 2019].

[17] "Terminal", Dle.rae.es. [Online]. Available: https://dle.rae.es/srv/search?m=30&w=terminal. [Accessed: 22- Oct- 2019].

Vista general

Este documento está dividido en 3 secciones, la primera es la parte introductoria del documento que se presentó en los párrafos anteriores. La sección 2 consiste en una descripción general del proyecto a realizar, por último en la sección 3 se muestran los requerimientos y la forma en la que los mismos se organizan.

Descripción general

En esta sección se describen de forma general los factores a tener en cuenta para el desarrollo del software del proyecto.

Perspectiva del producto

El software a desarrollarse es independiente y será desarrollado a partir de un diseño propuesto, ese decir, no se modificará algún otro existente. Así pues, funcionará bajo restricciones propias que se verán más adelante.

El siguiente diagrama muestra de forma general los módulos principales del software a desarrollar.

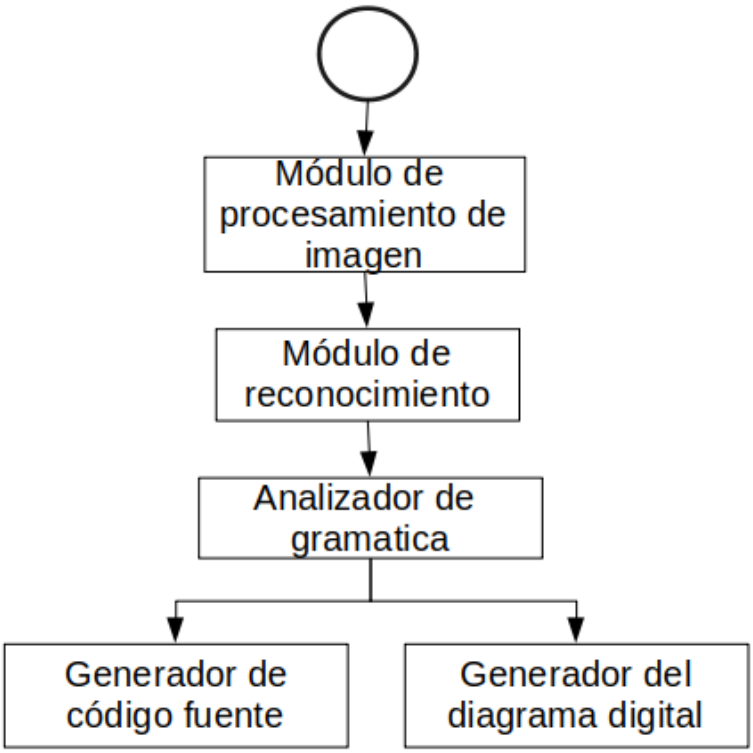


Figura 1. Vista general de los componentes de software a desarrollar.

Nota: No representa el pipeline a emplear, este último será definido en la etapa de modelado.

Interfaces de usuario: El único tipo de usuario son los mismos desarrolladores del proyecto de investigación, para el entrenamiento de la red neuronal se tendrá una GUI en donde podrá iniciar el entrenamiento; por otra parte, para probar con diagramas de flujo existirá otra vista de la GUI para hacer pruebas; así mismo donde se podrá ver el resultado del mismo, el código fuente en lenguaje C y diagrama digital reconstruido.

Interfaces de hardware: El programa se ejecuta en una computadora, por lo cual para ejecutarlo y poder ver los resultados, se hace uso del mouse, teclado y monitor.

Interfaces de software: Se usa la interfaz por software que lee la información en forma de imagen y de igual forma la del sistema operativo para mostrar una ventana gráfica y escribir archivos.

Restricciones de memoria: Es recomendable que la computadora donde se ejecute el programa, para reconocer diagramas de flujo, tenga al menos 8 GB de RAM, y al menos 3.9 GB de espacio libre en memoria secundaria.

Funcionamiento:

- Según los recursos de hardware disponibles y la entrada puede variar el tiempo de procesamiento, tanto del entrenamiento de la CNN así como el de probar con determinada imagen de un diagrama de flujo trazado a mano.
- Se podrá ejecutar el programa a través de la Terminal usando el lenguaje de programación *Python3* y las bibliotecas, necesarias para este último.
- El modelo con una red neuronal convolucional se podrá entrenar antes de poder hacer cualquier tarea de detección.
- El usuario podrá iniciar el entrenamiento de un nuevo modelo de CNN, determinar algunos parámetros del entrenamiento y visualizar detalles tanto del entrenamiento, así como del reporte de su rendimiento una vez se termine el proceso de entrenamiento.
- El usuario para probar podrá elegir la imagen del diagrama de flujo a procesar, una vez que se haga la tarea de reconocimiento y análisis de gramática, el usuario podrá ver el diagrama digital como imagen y se guardará la misma en la computadora, así mismo podrá ver el código fuente, guardar el archivo con extensión .c y se hará una compilación para probar que funciona tal código.

Funcionalidad del producto

- Entrenar modelo de CNN y visualizar datos del entrenamiento.
- Recuperar imagen del diagrama de flujo.
- Mejorar la imagen usando técnicas de procesamiento de imágenes.
- Reconocer y clasificar figuras del diagrama de flujo.
- Reconocer y clasificar los caracteres (texto).
- Hacer análisis gramatical para verificar la estructura del diagrama de flujo.
- Convertir una imagen de un diagrama de flujo trazada a mano en una versión digital del mismo.
- Convertir una imagen de un diagrama de flujo trazada a mano en el código equivalente del mismo, el código en lenguaje C.

Características del usuario

Como ya se mencionó antes, el único tipo de usuario son los mismos desarrolladores del proyecto de investigación, entonces se especifican las siguientes características:

A. Nivel de conocimientos:
    a. Básico en lenguaje de programación C.
    b. Intermedio en diagramas de flujo.
    c. Intermedio en reconocimiento de patrones.
    d. Avanzado en programación.
B. Experiencia.
    a. Uso de computadora.
    b. Diseño de algoritmos usando diagramas de flujo.
C. Experiencia técnica.
    a. Ejecutando programas de consola hechos en Python.
    b. Trabajando con CNNs.
    c. Programando en Python3 y C.

Restricciones generales

- Para el entrenamiento es necesaria una computadora con al menos 8 GB de RAM y 3.9 de espacio libre en memoria secundaria, si se cuenta con GPU es deseable configurar el entorno de pruebas usando dicha GPU; la alternativa es usar el servicio de nube llamado Colab, que permite usar una GPU de manera gratuita.
- Se debe guardar el modelo de CNN cada vez que se haya mejorado el rendimiento en el proceso de entrenamiento.

Presunciones y dependencias

+ Se asume que el software correrá en una distribución GNU/Linux con Python3 instalado y sus correspondientes bibliotecas o bien usando un servicio en la nube como Colab.
+ El tiempo de entrenamiento de la red depende de la implementación de la misma y el tamaño del conjunto de entrenamiento y validación, por otra parte, de los recursos de hardware de la computadora en la que se haga el entrenamiento.

+ La eficacia del reconocimiento del texto dependerá de la legibilidad del mismo. Se propone utilizar un modelo de clasificación existente para los caracteres del texto.
+ Se determina un conjunto finito de clases de figuras, conectores (ver anexo 1) y caracteres (ver anexo 2) que pueden usarse en los diagramas de flujo.

Especificación de requerimientos

Requisitos de la interfaz externa

Interfaz con el software

Implementar una arquitectura existente de red neuronal para la localización del texto y clasificación de los caracteres del texto.

Requerimientos funcionales

Identificador	Nombre	Descripción	Requerimientos relacionados
RF1	Cargar imagen	El sistema debe de cargar la imagen del diagrama de flujo del sistema de archivos de la computadora.	
RF2	Preprocesamiento de la imagen	El sistema debe de hacer un preprocesamiento de la imagen usando técnicas de procesamiento de imágenes.	RF1
RF3	Segmentación figura / texto	El sistema debe de ser capaz de segmentar entre lo que es una figura o conector y texto ya que cada uno se procesa de forma diferente.	RF2
RF4	Entrenar modelo de figuras	El sistema debe tener un modelo capaz de aprender de manera supervisada para diferenciar las símbolos a partir de un conjunto de imágenes proporcionadas.	

RF5	Clasificación de figuras y conectores	El sistema debe clasificar que tipo de figura o conector usando redes neuronales convolucionales.	RF3
RF6	Clasificación de texto	El sistema debe clasificar los caracteres del texto segmentados usando redes neuronales convolucionales.	RF3
RF7	Análisis de gramática	El sistema debe usar análisis de gramática para verificar la estructura global del diagrama de flujo.	RF5
RF8	Generar versión digital del diagrama de flujo	El sistema debe de generar una versión digital del diagrama de flujo a partir de la clasificación de figuras, conectores y texto de la imagen.	RF1, RF2, RF3, RF5, RF6, RF7
RF9	Generar código fuente en C equivalente	El sistema debe de generar el código fuente en C equivalente al diagrama de flujo a partir de la clasificación de de figuras, conectores y texto de la imagen.	RF1, RF2, RF3, RF5, RF6, RF7
RF10	Compilar código fuente	Para probar que el código generado funciona, se hace una llamada al sistema operativo para compilar	RF9
RF11	GUI de control	Poder reconocer diagramas de flujo trazados a mano así como hacer el entrenamiento del modelo, mediante ventanas, botones y cajas de texto.	RF3, RF4

Flujo de la información

Requerimiento	Entrada	Proceso	Salida
RF1	Imagen	1. Seleccionar archivo. 2. Leer archivo como objeto. 3. Seleccionar espacio de colores.	Objeto

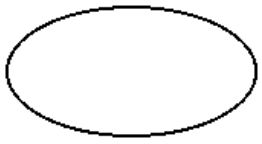
RF2	Imagen	1. Seleccionar algoritmo(s) de preprocesamiento. 2. Aplicar algoritmo(s) de preprocesamiento a la imagen.	Imagen preprocesada
RF3	Imagen	1. Aplicar algoritmo(s) de segmentación a la imagen.	Imagen segmentada.
RF4	Conjunto de imágenes	1. Cargar el conjunto de imágenes. 2. Empezar proceso iterativo de disminución de la error en la red.	Red parametrizada (entrenada), gráficas de rendimiento y matriz de confusión.
RF5	Imagen segmentada	1. Probar en la red para etiquetar la imagen ya sea alguna figura o conector con modelo pre-entrenado.	Objeto nodo con referencia en la imagen y la etiqueta.
RF6	Imagen segmentada	1. Hacer el reconocimiento de texto con modelo prenentrenado.	Objeto nodo con referencia en la imagen y el texto.
RF7	Conjunto de nodos	1. Construir grafo dirigido a. Definir adyacencia.	Grafo dirigido o mensaje de error.
RF8	Grafo dirigido	1. Crear archivo. 2. Recorrer grafo. a. Traducir cada nodo a representación gráfica digital. 3. Mostrar resultado.	Archivo con diagrama digital en .png.
RF9	Grafo dirigido	1. Crear archivo. 2. Recorrer grafo. a. Traducir nodo a código.	Archivo de código fuente .c.
RF10	Archivo de código fuente .c.	1. Llamar comando de compilación.	Mostrar resultado

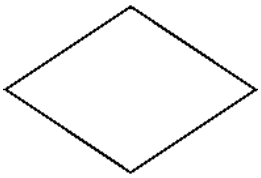
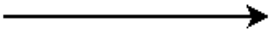
		2. Mostrar salida del proceso de compilación	dependiendo de la salida
RF11	Clicks	1. Si se elige entrenar, mandar a la ventana para entrenar modelo. 2. Si se elige reconocer diagrama de flujo, mandar a la ventana para hacer el reconocimiento.	Uso de las otras funcionalidades.

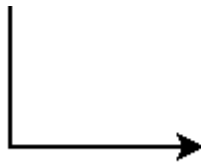
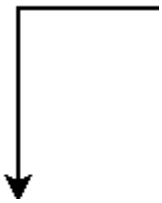
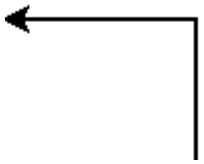
Requerimientos de desempeño

A. Dada la naturaleza monolítica del sistema de software, el número de terminales en el sistema, por computadora, es uno.
B. Dado que el sistema no está pensado para usarse de forma múltiple por varios usuarios, solo un usuario a la vez, por computadora, podrá usar el software para hacer las pruebas de la fase de experimentación.
C. Se espera entrenar a la red con aproximadamente 1,600 imágenes de figuras y conectores del diagrama de flujo, este entrenamiento debe tener una complejidad en tiempo polinomial como cota superior.
D. El reconocimiento de un diagrama de flujo, generación de diagrama digital y código fuente, se espera que tome un tiempo de procesamiento de menos de 10 segundos.

Requerimientos de la base de datos lógica

Identificador	Nombre (símbolo)	Descripción	Ejemplo
1	Terminal / Inicio	El sistema debe tener un dataset de figuras de terminal / inicio.	
2	Entrada de datos	El sistema debe tener un dataset de figuras de entrada de datos.	

3	Proceso	El sistema debe tener un dataset de figuras de proceso.	
4	Decisión	El sistema debe tener un dataset de figuras de decisión.	
5	Imprimir resultados	El sistema debe tener un dataset de figuras de Imprimir resultados.	
6	Flecha recta hacia arriba	El sistema debe tener un dataset de figuras de flujo de datos.	
7	Flecha recta derecha	El sistema debe tener un dataset de figuras de flujo de datos.	
8	Flecha recta abajo	El sistema debe tener un dataset de figuras de flujo de datos.	
9	Flecha recta izquierda	El sistema debe tener un dataset de figuras de flujo de datos.	
10	Flecha cuadrada arriba	El sistema debe tener un dataset de figuras de flujo de datos.	

11	Flecha cuadrada derecha	El sistema debe tener un dataset de figuras de flujo de datos.	
12	Flecha cuadrada abajo	El sistema debe tener un dataset de figuras de flujo de datos.	
13	Flecha cuadrada izquierda	El sistema debe tener un dataset de figuras de flujo de datos.	

Nota: Todas las representaciones de los símbolos es un dato de tipo imagen.

Restricciones de diseño

A. Uso de una GPU para el entrenamiento en una computadora o en Colab.
B. Conjunto limitado de símbolos a usar en los diagramas a reconocer, ver anexo 1 y 2.

Atributos

Confiabilidad

- Que reconozca los diagramas de flujo, genere su versión digital y código fuente de forma eficaz, probar con los siguientes algoritmos:
    - Imprimir un mensaje simple: “Hola mundo”.
    - Cálculo del n-ésimo término de la sucesión de Fibonacci.
    - Cálculo del factorial de un número entero.

Disponibilidad

- Durante el proceso de entrenamiento de la red neuronal, es deseable, tener un regulador de voltaje para evitar daños a la computadora ya que estará

durante un significativo período de tiempo procesando el entrenamiento de la red neuronal.
- Durante el proceso de entrenamiento, se guardarán, los parámetros del modelo de la red neuronal (cada vez que haya una mejora en el rendimiento del modelo), por si en dado caso ocurre un fallo eléctrico, los parámetros encontrados no se tengan que volver a calcular y no resulte en vano el procesamiento hecho antes.

Mantenibilidad

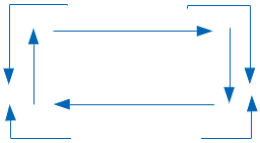
- Emplear el paradigma de programación orientada a objetos.
- Para la codificación se usará el estándar de codificación para Python llamado PEP-8, además el nombre de las variables, comentarios y clases serán en inglés.
- Uso del controlador de versiones **Git** y la plataforma **GitHub** para alojar el proyecto en la nube y publicarlo bajo una licencia con el fin de que otras personas interesadas puedan probarlo.

Portabilidad

- El equipo debe tener instalado Python 3.7.
- Se deben tener instaladas los siguientes paquetes para Python:
    - Numpy.
    - Keras.
    - TensorFlow.
    - Matplotlib.
    - OpenCV.
    - Pandas.
    - Scikit-learn.

Anexo 1. Conjunto de figuras y conectores (flechas) permitidos en el diagrama de flujo a reconocer

Nombre	Símbolo
Inicio/Terminal	

Entrada de datos	
Proceso	
Decisión	
Imprimir resultados	
Flujo de datos (conectores)	

Anexo 2. Conjunto de caracteres permitidos en el diagrama de flujo a reconocer

Tipo	Conjunto
Letras	[A-Z], [a-z]
Números	[0-9]
Especiales	+,*,-,=,_, \, /, (,), [,], ., “,”,^, ;,¿,?,¡,!, &, |, °, #, $, %, {,}, :,-, >, <

Apéndice C

Aquí se encuentra las versiones de la matriz de riesgos, a partir de C.2 sólo se incluyen la tabla control de versiones y la matriz de riesgos.

C.1 Matriz de riesgos versión inicial

CONTROL DE VERSIONES					
Autor(es)	Fecha de modificación	Versión	Descripción del cambio	Revisó	Estado
DBM	19/10/2019	1.0	Creación del documento	OFCG, ROCL, KRM	APROBADO

Propósito

Definir un marco metodológico para la correcta evaluación de los riesgos que se pueden encontrar dentro del proyecto, en el contexto de Trabajo Terminal I y II.

Niveles de probabilidad

Los niveles de probabilidad expresan el nivel que se define para la ocurrencia de un suceso, tales niveles se muestran en la siguiente tabla:

Nivel	Probabilidad	Descripción
1	Raro	Solo ocurrirá en casos excepcionales
2	Improbable	Puede ocurrir en algún momento pero las condiciones del proyecto no dan pie a que suceda
3	Posible	Podría ocurrir en algún momento del proyecto
4	Probable	Es probable que ocurra en la mayoría de las circunstancias del proyecto
5	Casi Seguro	Se espera que ocurra para todas las posibles circunstancias

Niveles de impacto

El nivel de impacto, como su nombre lo indica permite identificar qué tanto impactaría en el proyecto, la ocurrencia de algún suceso riesgoso para el proyecto, tales niveles se muestran en la siguiente tabla:

Nivel	Impacto	Descripción
1	Insignificante	Si el hecho se llega a presentar no afecta la realización del proyecto
2	Menor	Si el hecho se llega a presentar el impacto no es significativo para la realización del proyecto no, genera una desviación significativa
3	Moderado	Si el hecho se llega a presentar el impacto es aún controlable y no afecta de manera grave la realización del proyecto.

4	Mayor	Si el hecho se llega a presentar el impacto es mucho mayor e implica cambios significativos en la realización del proyecto.
5	Catastrófico	Si el hecho se llega a presentar el impacto es grave y compromete la realización del proyecto.

Nivel de riesgo

Una vez definidos los niveles de probabilidad, y los niveles de impacto se debe calcular el nivel del riesgo, para ello se debe realizar una multiplicación simple de los niveles anteriores, con ello se evalúan los riesgos que se detectan dentro del proyecto, siempre hay que considerar que a menor probabilidad e impacto, menor será el nivel del riesgo, a mayor probabilidad e impacto, mayor será el nivel de riesgo.

Probabilidad	Impacto				
	Insignificante (1)	Menor (2)	Moderado (3)	Mayor (4)	Catastrófico (5)
Raro (1)	1	2	3	4	5
Improbable (2)	2	4	6	8	10
Posible (3)	3	6	9	12	15
Probable (4)	4	8	12	16	20
Casi Seguro (5)	5	10	15	20	25

De esta manera se obtiene la siguiente matriz de nivel de riesgo

Nivel de riesgo	Probabilidad X Impacto
Muy Alto	> = 20
Alto	De 15 a 19
Medio	De 9 a 14
Bajo	De 6 a 8
Muy bajo	< = 5

Matriz de riesgos

Una vez definidos los niveles anteriores se procede a la identificación, registro, y rastreo de los riesgos detectados, para tal efecto existe la siguiente tabla que será utilizada para el proyecto de trabajo terminal “Reconocimiento de diagramas de flujo trazados a mano usando redes neuronales convolucionales”.

ID	Descripción	Fase afectada	Causa	Probabilidad	Impacto	Nivel del riesgo	Estrategia de prevención	Estrategia de mitigación
1	Un equipo de cómputo falla.	Cualquier fase puede ser afectada.	Uso excesivo, mojarse, corto eléctrico	3	3	9	Cuando sea posible, usar las computadoras de la escuela, evitar salir con la	Usar otra computadora (de la escuela), pidiendo permiso para poder instalar software

			caída.				computadora cuando llueve, no usar cuando hay tormenta eléctrica, no ponerla en lugares altos o poco estables. Para el equipo de la escuela, revisar que haya buena ventilación y usar solo lo necesario.	de desarrollo o bien, juntarse cada tarde a trabajar en una máquina en equipo.
2	Indisposición por enfermedad de un integrante.	Cualquier fase puede ser afectada.	Malos hábitos, presión, estrés.	3	4	12	Buscar dormir al menos 7 horas diarias, mejorar hábitos alimenticios, hacer ejercicio regularmente, tomar más agua y hacer actividad como hobby.	Atenderse de forma oportuna en un servicio de salud, reposar un par de días, trabajar algunos días desde casa.
3	Agotar el tiempo.	Ciclos.	Imprevistos, distracciones, carga de otras materias u ocurrencia de otros riesgos.	3	4	12	Llevar el control respecto al cronograma de actividades, identificar actividades clave, trabajar los domingos, incrementar las horas de trabajo por día, administrar el tiempo y esfuerzo personal de cada integrante.	Ya no llevar a cabo el segundo ciclo en la experimentación, acercarse más a los asesores, reducir el alcance del proyecto.
4	Problemas personales con los revisores.	Juntas con los revisores y creación de reportes y presentaciones.	Malentendidos.	2	3	6	Hablar regularmente con los asesores, tomar en cuenta la palabra de los asesores y cumplir con acuerdos.	Tratar de aclarar los malententidos.
5	Huelga en la escuela.	Cualquier fase puede ser afectada.	Política que vulnera los derechos u	3	5	15	Estar al pendiente de los avisos que involucran a la comunidad estudiantil.	Trabajar desde casa.

			oportunidades de los politécnicos.					
6	Paro en la escuela.	Cualquier fase puede ser afectada.	Docentes, administrativos y/o estudiantes exigen algo.	3	4	12	No apoyar movimientos que carecen de argumentos, si se participa incitar a formas diplomáticas para llegar a acuerdos.	Trabajar desde casa y no apoyar acciones inadecuadas.
7	Se incendia un equipo de cómputo cuando se hace procesamiento ..	Incremento de red neuronal	Ventilación insuficiente o corto circuito.	1	5	5	No dejar líquidos cerca de los equipos, permitir una buena ventilación, checar el estado del cableado y checar que no haya se caliente el procesador.	Si se está cerca apagar con extintor de clase C, tratar de aislar el fuego para evitar propagación.
8	Asesor renuncia al proyecto o ya no puede participar.	Reuniones con el cliente, y fase transversal de orientación de las fases.	Conflicto de intereses, problemas laborales o salud	3	3	9	Si el asesor tiene algún problema tratar de apoyar de alguna manera.	Mantener comunicación, brindar apoyo al asesor, buscar ayuda con otros docentes del área de proyecto.
9	Conflictos entre DBM y OFCG.	Cualquier fase puede ser afectada.	Conflicto de intereses o malentendidos.	2	4	8	Resolver las diferencias hablando, no ocultar información y comentar inquietudes.	Tratar de reconciliar intereses hablando, y si es necesario buscar ayuda con los asesores.
10	No haber considerado un riesgo con nivel alto de probabilidad e impacto.	Cualquier fase puede ser afectada.	Poco análisis al elaborar y revisar la matriz de riesgos	2	5	10	Revisar el documento de matriz de riesgos con asesor y director.	Identificar alternativas para mitigar las consecuencias de la ocurrencia del riesgo.
11	El nivel de conocimientos de los involucrados no son	Ciclos.	Poco tiempo para la fase de estudio o	2	4	8	Además de la fase de estudio, seguir aprendiendo de las áreas involucradas. Acercarse con los	Buscar asesoría con un experto en el área en internet u otra escuela.

	suficientes para desarrollar el proyecto.		temas muy complejos de abordar.				asesores para ayuda técnica.	
12	Se trunca el entrenamiento de la red.	Incremento de red neuronal convolucional.	Fallo eléctrico.	3	3	9	Usar un no break o tener un programa que guarde periódicamente los parámetros de la red neuronal en entrenamiento.	Trabajar más tiempo, ya sean más horas por día o en domingo.
13	Baja eficacia	Ciclos	Algoritmos propuestos fallan	3	4	12	Mejorar habilidades en diseño de algoritmos, usar técnicas de búsqueda de hiperparámetros, buscar y usar trucos empleados en diseño de redes neuronales.	Hacer los incrementos de diagrama digital y generación de código, con entradas controladas. E informar por qué el pipeline de reconocimiento propuesto no funcionó.

C.2 Matriz de riesgos versión actualizada

CONTROL DE VERSIONES					
Autor(es)	Fecha de modificación	Versión	Descripción del cambio	Revisó	Estado
DBM	19/10/2019	1.0	Creación del documento	OFCG, ROCL, KRM	APROBADO
DBM	21/01/2020	1.1	Actualización de riesgos	OFCG	APROBADO

ID	Descripción	Fase afectada	Causa	Probabilidad	Impacto	Nivel del riesgo	Estrategia de prevención	Estrategia de mitigación
1	Un equipo de cómputo falla.	Cualquier fase puede ser afectada.	Uso excesivo, mojarse, corto	3	3	9	Cuando sea posible, usar las computadoras de la escuela, evitar salir con la	Usar otra computadora (de la escuela), pidiendo permiso para poder

			eléctrico caída.				computadora cuando llueve, no usar cuando hay tormenta eléctrica, no ponerla en lugares altos o poco estables. Para el equipo de la escuela, revisar que haya buena ventilación y usar solo lo necesario.	instalar software de desarrollo o bien, juntarse cada tarde a trabajar en una máquina en equipo.
2	Indisposición por enfermedad de un integrante.	Cualquier fase puede ser afectada.	Malos hábitos, presión, estrés.	3	4	12	Buscar dormir al menos 7 horas diarias, mejorar hábitos alimenticios, hacer ejercicio regularmente, tomar más agua y hacer actividad como hobby.	Atenderse de forma oportuna en un servicio de salud, reposar un par de días, trabajar algunos días desde casa.
3	Agotar el tiempo.	Construcción	Imprevistos, distracciones, carga de otras materias u ocurrencia de otros riesgos.	3	4	12	Llevar el control respecto al cronograma de actividades, identificar actividades clave, trabajar los domingos, incrementar las horas de trabajo por día, administrar el tiempo y esfuerzo personal de cada integrante.	Reducir el alcance del proyecto, analizar que ha salido mal.
4	Huelga en la escuela o paro.	Cualquier fase puede ser afectada.	Política que vulnera los derechos u oportunidades de los politécnicos.	3	5	15	No ignorar los avisos de la comunidad estudiantil.	Trabajar desde casa.
5	Se incendia un equipo de cómputo cuando se hace procesamiento.	Entrenamiento de red neuronal	Ventilación insuficiente o corto circuito.	1	5	5	No dejar líquidos cerca de los equipos, permitir una buena ventilación, checar el estado del cableado y checar que no se caliente demasiado el procesador de la computadora.	Si se está cerca apagar con extintor de clase C, tratar de aislar el fuego para evitar propagación.
6	Asesor renuncia al proyecto o ya no puede participar.	Reuniones con el cliente, y fase transversal de orientación de las fases.	Conflicto de intereses, problemas laborales o salud	3	3	9	Si el asesor tiene algún problema tratar de apoyar de alguna manera.	Mantener comunicación, brindar apoyo al asesor, buscar ayuda con otros docentes del área de proyecto.
7	El nivel de	Construcc	Poco	2	4	8	Además de la fase de	Buscar asesoría con

	conocimientos de los involucrados no son suficientes para desarrollar el proyecto.	ión.	tiempo para la fase de estudio o temas muy complejos de abordar.				estudio, seguir aprendiendo de las áreas involucradas. Acercarse con los asesores para ayuda técnica.	un experto en el área en internet u otra institución.
8	Se trunca el entrenamiento de la red.	Entrenamiento de red neuronal convolucional	Fallo eléctrico	3	3	9	Usar un no break, regulador de voltaje o tener un programa que guarde periódicamente los parámetros de la red neuronal durante el entrenamiento.	Trabajar más tiempo, ya sean más horas por día o en domingo.
9	Dataset no adecuado	Obtención del dataset	Mala calidad de las imágenes, imágenes incorrectas o cantidad insuficiente de imágenes.	4	4	16	Analizar las condiciones en que deben estar las imágenes del dataset antes de obtenerlo; pedir la opinión a un experto antes de obtenerlo; cuidar y especificar el proceso de captura de las imágenes, así como la segmentación de los mismos.	Si son insuficientes, hacer aumento de datos al generar variaciones automáticas de las imágenes ya existentes. Volver a crear el dataset en condiciones más controladas.
10	Sobreajuste en el modelo	Construcción	Cantidad insuficiente de datos, sobre entrenamiento con un conjunto en particular	3	4	12	Dividir el conjunto de datos en dos partes: entrenamiento y validación, usar una técnica de regularización.	Reducir el número de capas o aumentar el número de instancias del dataset o reducir la cantidad de épocas del proceso iterativo de entrenamiento.
11	Baja eficacia para generar código fuente y diagrama digital.	Construcción	La red no puede discriminar los tipos de clases	3	5	15	Estandarizar el tipo de símbolos que se permiten reconocer, además cuidar la obtención de datos y si es necesario hacer limpieza de los mismos; probar con diferentes parámetros.	Reducir el alcance del proyecto.

Apéndice D

Aquí encuentra las versiones del diseño de la arquitectura del sistema.

D.1 Arquitectura del sistema versión inicial

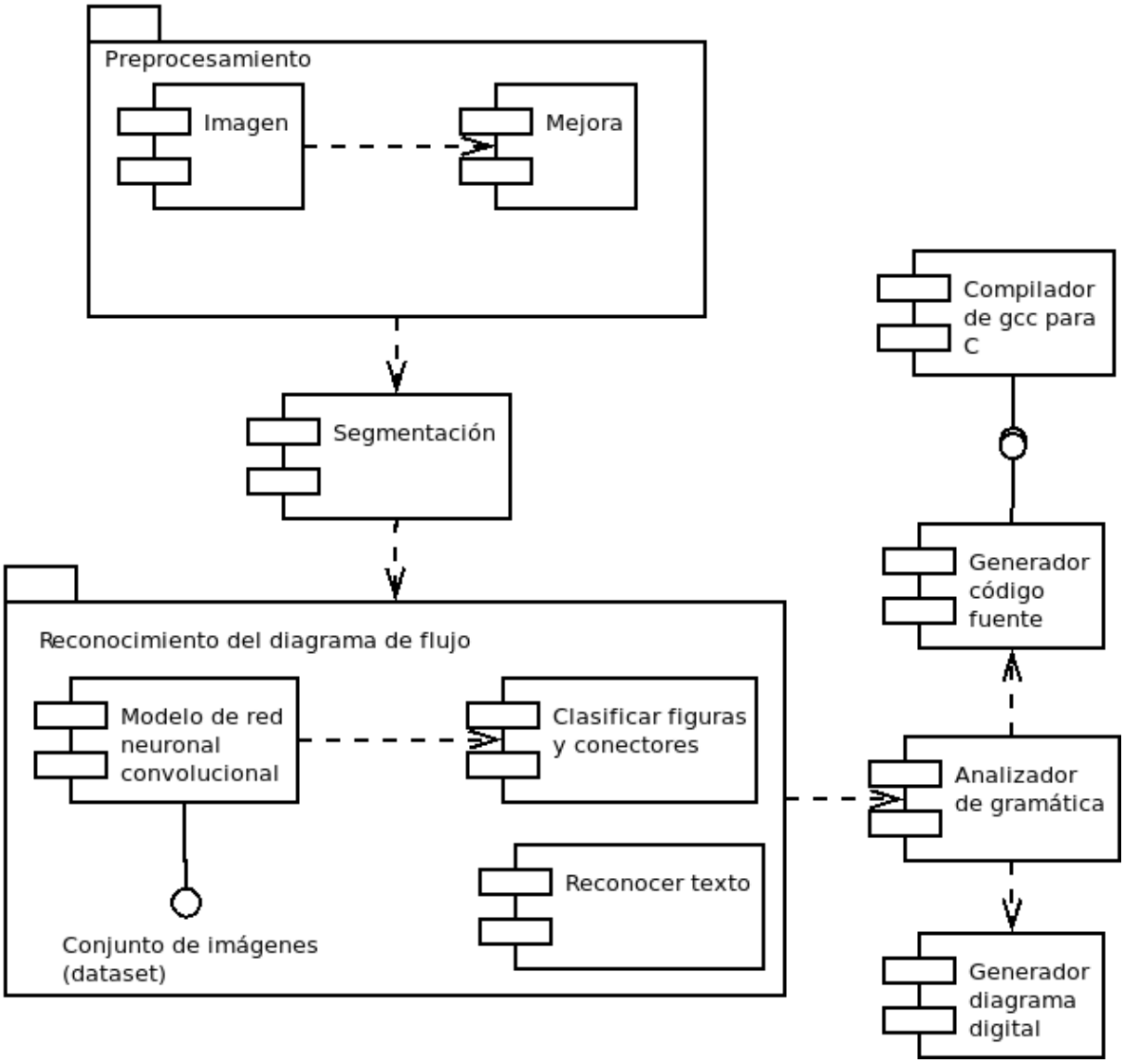


Diagrama UML de componentes

D.2 Arquitectura del sistema versión final

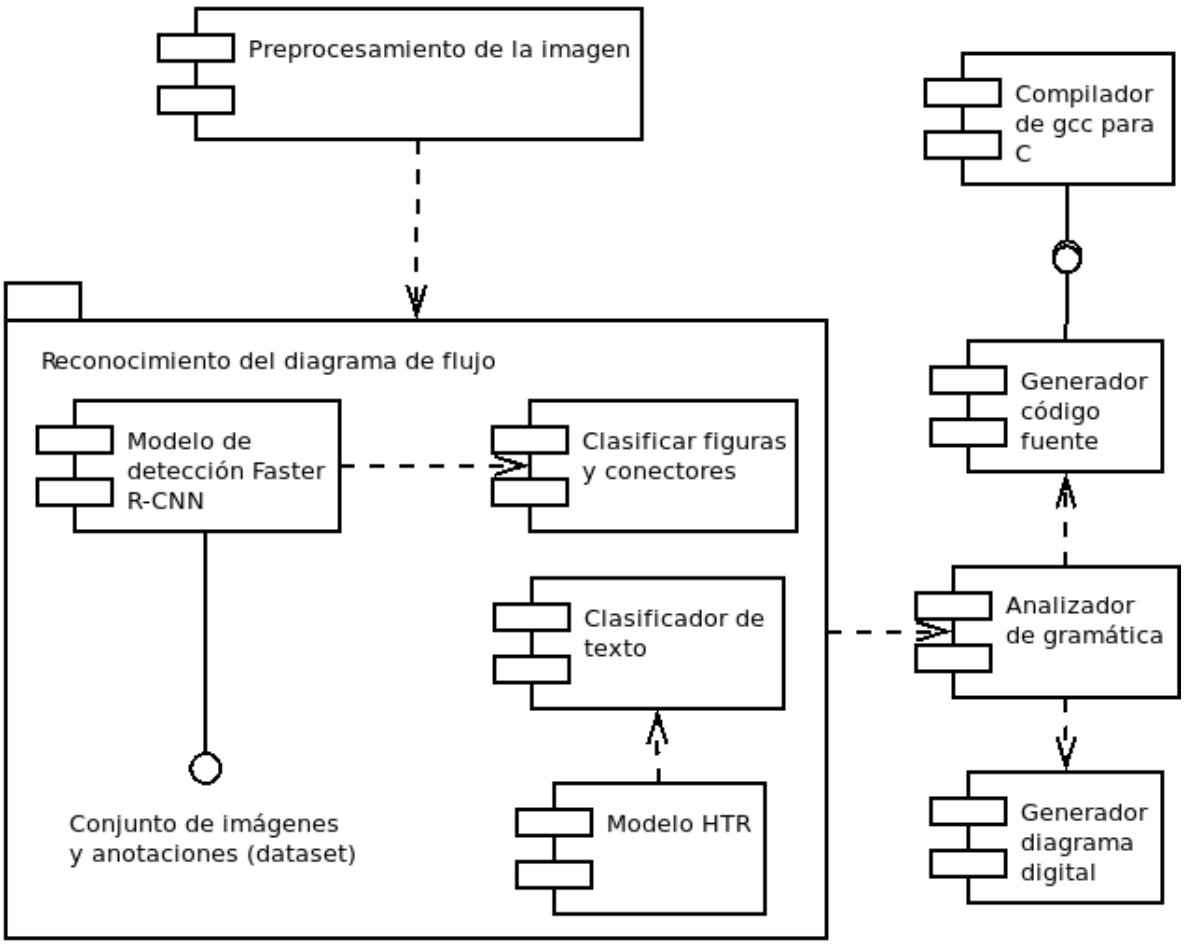
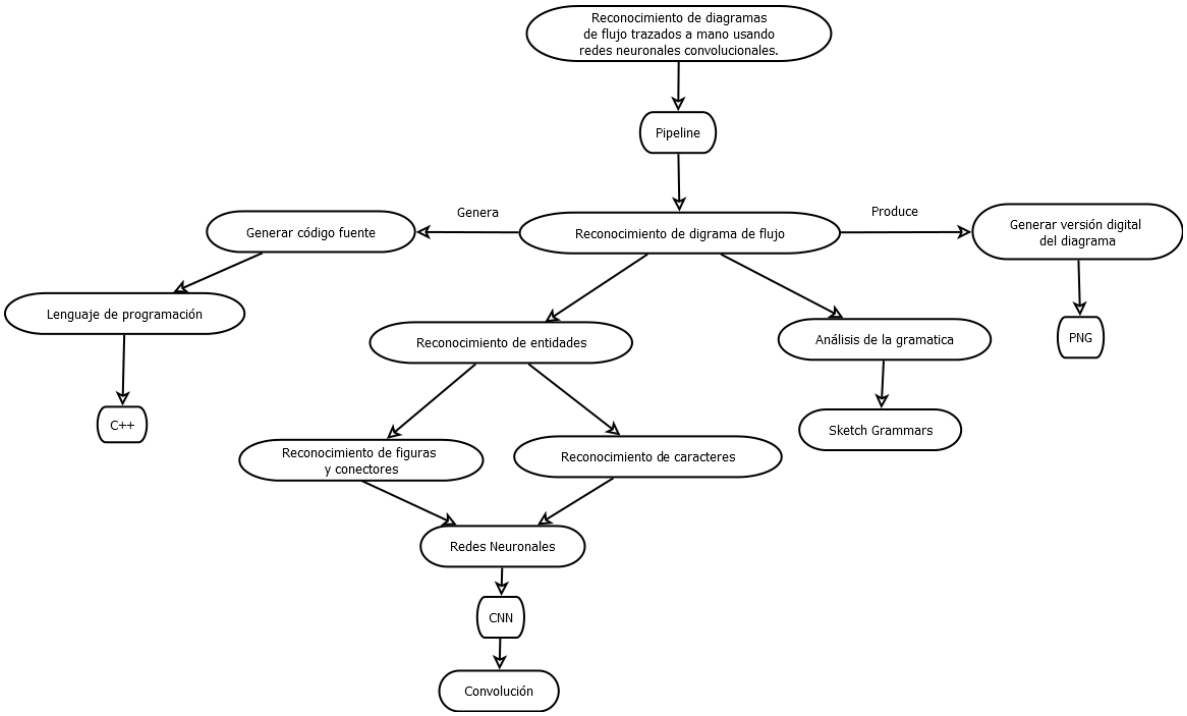


Diagrama UML de componentes

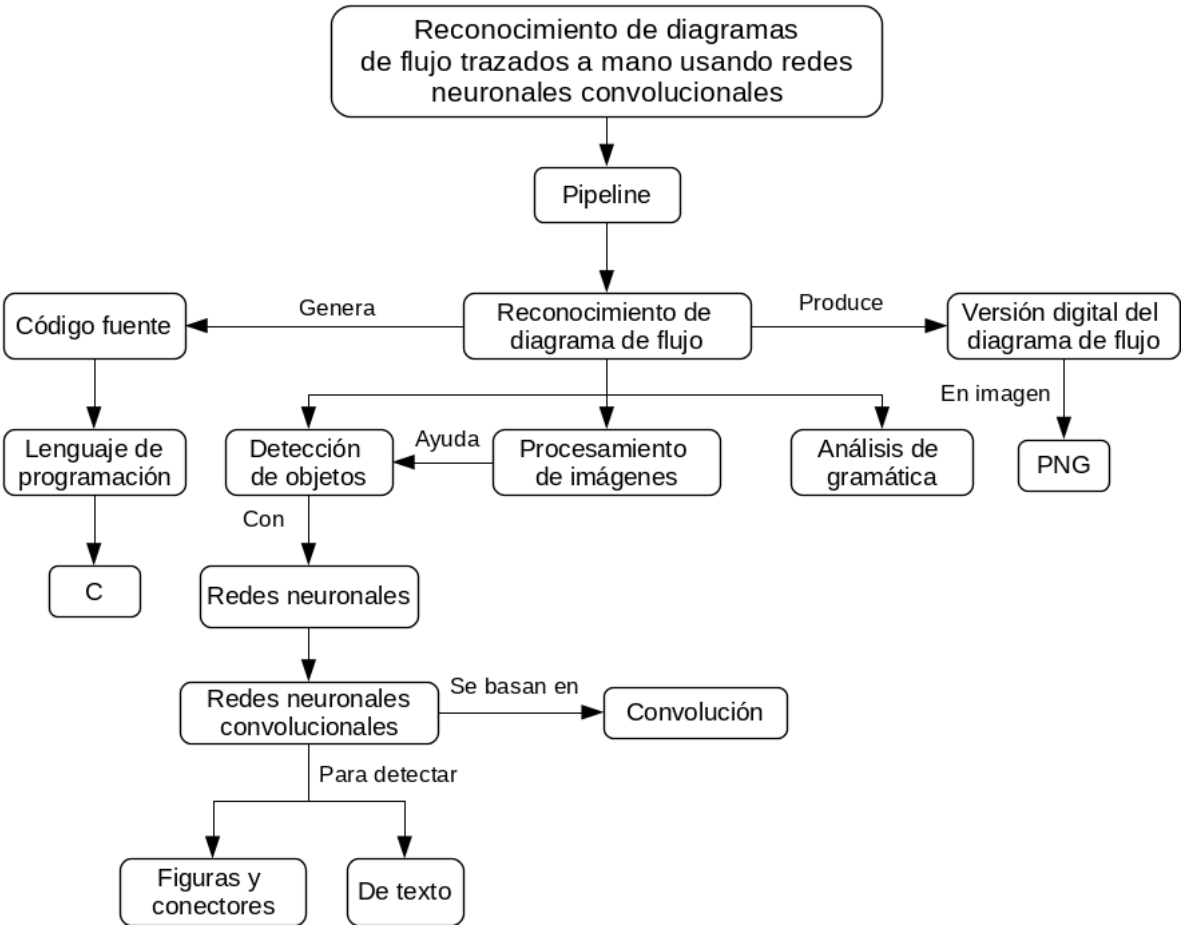
Apéndice E

En este apéndice encuentra los diagramas conceptuales para las versiones del sistema.

E.1 Diagrama conceptual versión inicial



E.2 Diagrama conceptual versión final



Apéndice F

Minutas de reunión con director y asesor, cada una de ellas muestra el propósito de la junta de trabajo, los participantes, acuerdos, acciones a realizar (y su seguimiento) y resumen de la reunión.

F.1 Reunión 1

Instituto Politécnico Nacional
Unidad Interdisciplinaria de Ingeniería campus Zacatecas
Minuta de junta de trabajo

DATOS GENERALES			
Lugar	Lab. docencia redes, UPIIZ	Fecha	25/10/2019
Equipo de trabajo		Hora inicio	12:15 P.M.
Propósito	Validar cronograma y SRS	Hora fin	12:30 P.M.

ASISTENTES Y ROLES DE LA JUNTA	
Nombre	Rol
M. en C. Roberto Oswaldo Cruz Leija, ROCL.	Director
Onder Francisco Campos García, OFCG.	Desarrollador
David Betancourt Montellano, DBM.	Desarrollador

REQUISITOS DE ENTRADA	
Descripción	Responsable
Llevar el cronograma (plan de actividades) terminado.	DBM, OFCG
Llevar el SRS, terminado.	DBM, OFCG

ACCIONES				
Acción	Responsable	Fecha probable	Listo	Fecha real
Cambiar "Mejora de la imagen" a preprocesamiento de la imagen. (SRS).	OFCG	~~25/10/2019~~	✓	29/10/2019

ACUERDOS	
Acuerdo	Involucrados
Reunirse la próxima semana, de preferencia el jueves 31/10/2019 a las 14:30.	OFCG, ROCL, DBM

RESUMEN
Revisión del SRS completo y confirmar que ya se había revisado el cronograma.

David Betancourt Montellano

Onder Francisco

1/1

F.2 Reunión 2

Instituto Politécnico Nacional
Unidad Interdisciplinaria de Ingeniería campus Zacatecas
Minuta de junta de trabajo

DATOS GENERALES			
Lugar	Lab. cómputo 1, edif. ligeros, UPIIZ.	Fecha	25/10/2019
Equipo de trabajo		Hora inicio	12:53 P.M.
Propósito	Validar cronograma y SRS.	Hora fin	01:10 P.M.

ASISTENTES Y ROLES DE LA JUNTA	
Nombre	Rol
M. en Ed. Karina Rodríguez Mejía (KRM)	Asesor
Onder Francisco Campos García, (OFCG)	Desarrollador
David Betancourt Montellano (DBM)	Desarrollador

REQUISITOS DE ENTRADA	
Descripción	Responsable
Llevar el cronograma terminado.	DBM, OFCG
Llevar el SRS terminado.	DBM, OFCG

ACCIONES				
Acción	Responsable	Fecha probable	Listo	Fecha real
SRS: Agregar acento en "análisis de gramatica"	DBM	28/10/2019	✓	29/10/2019
SRS: En símbolos permitidos, agregar el Ø	DBM	28/10/2019	✓	29/10/2019

ACUERDOS	
Acuerdo	Involucrados
Reunirse de nuevo, la siguiente semana, cualquier día entre el 28/10/2019 y el 30/10/2019, para validar diseños.	KRM, OFCG, DBM

RESUMEN

Karina Rodríguez Mejía Onder Francisco Campos David Betancourt Montellano

1/1

F.3 Reunión 3

Instituto Politécnico Nacional
Unidad Interdisciplinaria de Ingeniería campus Zacatecas
Minuta de junta de trabajo

DATOS GENERALES			
Lugar	Lab. Cómputo 1, Edif Ingenios UPIIZ	Fecha	4/11/19
Equipo de trabajo		Hora inicio	12:45 P.M.
Propósito		Hora fin	12:57 P.M.

ASISTENTES Y ROLES DE LA JUNTA	
Nombre	Rol
M. en Ed. Karina Rodríguez Mejía (KRM)	Asesor
Onder Francisco Campos García (OFCG)	Desarrollador
David Betancourt Montellano (DBM)	Desarrollador

REQUISITOS DE ENTRADA	
Descripción	Responsable
llevar Matriz de riesgos, Mostrar Arquitectura de sis	DBM, OFCG
llevar Diagrama conceptual, llevar matriz de trazabil	DBM, OFCG

ACCIONES				
Acción	Responsable	Fecha probable	Listo	Fecha real

ACUERDOS	
Acuerdo	Involucrados

RESUMEN

Revisión de matriz de riesgos, arquitectura de sistema, diagrama conceptual, matriz de trazabilidad

Karina Rodríguez Mejía — Onder Francisco Campos — David Betancourt Montellano

1/1

F.4 Reunión 4

Instituto Politécnico Nacional
Unidad Interdisciplinaria de Ingeniería campus Zacatecas
Minuta de junta de trabajo

DATOS GENERALES			
Lugar	CDS, edificio ligeros, OP117	Fecha	4/11/19
Equipo de trabajo		Hora inicio	1:05 P.M.
Propósito	Validar algunos artefactos de Diseño y calidad	Hora fin	1:15 P.M.

ASISTENTES Y ROLES DE LA JUNTA	
Nombre	Rol
M. en C. Roberto Oswaldo Cruz Leya, ROCL	Director
Onder Francisco Campos Garcia, OFCG	Desarrollador
David Betancourt Montellano, DBM	Desarrollador

REQUISITOS DE ENTRADA	
Descripción	Responsable
llevar matriz de Riesgos, llevar arquitectura de S/S,	DBM, OFCG
llevar diagrama conceptual, llevar matriz de trazabilidad	DBM, OFCG

ACCIONES				
Acción	Responsable	Fecha probable	Listo	Fecha real
Cambiar conectores en arq de sistema y ver otras opciones en arquitectura distribuida	DBM	05/11/2019	✓	05/11/2019

ACUERDOS	
Acuerdo	Involucrados

RESUMEN
Revisión de matriz de riesgos, arquitectura del sistema, diagrama conceptual, matriz de trazabilidad

Roberto O. Cruz Leya — Onder Francisco Campos — David Betancourt Montellano

1/1

F.5 Reunión 5

Instituto Politécnico Nacional
Unidad Interdisciplinaria de Ingeniería campus Zacatecas
Minuta de junta de trabajo

DATOS GENERALES			
Lugar	CDS edificio ligeros UPIIZ	Fecha	15/11/2019
Equipo de trabajo		Hora inicio	10:00 A.M.
Propósito	Revisión del reporte final	Hora fin	11:00 A.M.

ASISTENTES Y ROLES DE LA JUNTA	
Nombre	Rol
M. en C Roberto Oswaldo Cruz Leyva (ROCL)	Director
Onder Francisco Campos García (OFCG)	Desarrollador
David Betancourt Montellano (DBM)	Desarrollador

REQUISITOS DE ENTRADA	
Descripción	Responsable
llevar el reporte final	DBM, OFCG

ACCIONES				
Acción	Responsable	Fecha probable	Listo	Fecha real
• Homogeneizar formato	DBM	15/11/19	✓	16/11/2019
• Corregir redacción en cuanto a del plan de trabajo y persistencia de	DBM, OFCG	15/11/19		
	DBM, OFCG	15/11/19	✓	16/11/2019
• Añadir descripción a tablas y cuadros	DBM	15/11/19	✓	17/11/2019
◦ Agregar que significa SRS	DBM	15/11/19	✓	16/11/2019

ACUERDOS	
Acuerdo	Involucrados
Agregar conclusiones	DBM, OFCG
Agregar resumen	DBM, OFCG
Verse el 19/11/2019 en la mañana para últimos detalles y firmar	DBM, OFCG, ROCL

RESUMEN
Se revisó el reporte final. de inicio a fin

David Betancourt Montellano

Onder Francisco

1/1

F.6 Reunión 6

Instituto Politécnico Nacional
Unidad Interdisciplinaria de Ingeniería campus Zacatecas
Minuta de junta de trabajo

DATOS GENERALES			
Lugar	Lab. cómputo 1, Ed. ligeros / UPIIZ	Fecha	15/11/2019
Equipo de trabajo		Hora inicio	01:02 P.M.
Propósito	Retroalimentación sobre revisión de reporte final TTI	Hora fin	01:15 P.M.

ASISTENTES Y ROLES DE LA JUNTA	
Nombre	Rol
M. en Ed. Karina Rodríguez Mejía (KRM)	Asesor
David Betancourt Montellano (DBM)	Desarrollador
Onder Francisco Campos García (OFCG)	Desarrollador

REQUISITOS DE ENTRADA	
Descripción	Responsable
Llevar el reporte final en una computadora	DBM, OFCG

ACCIONES				
Acción	Responsable	Fecha probable	Listo	Fecha real
Considerar los comentarios de KRM que agregó al documento del reporte final de TTI	DBM, OFCG	16/11/2019	✓	18/11/19

ACUERDOS	
Acuerdo	Involucrados
Agregar conclusiones	DBM, OFCG
Agregar resumein	DBM, OFCG
Verse el 19/11/2019 en la mañana para últimos detalles y firmar	DBM, OFCG, KRM

RESUMEN
KRM nos comentó sobre la revisión y los comentarios que agregó al documento.

Karina Rodríguez Mejía — David Betancourt Montellano — Onder Francisco

1/1

Apéndice G

Anexos asociados a la fase de estudio planeada para repasar y/o aprender sobre técnicas de procesamiento de imágenes y redes neuronales convolucionales.

G.1 Certificado de finalización (curso de Udemy)

Nota: El certificado sólo se genera a nombre de uno de los desarrolladores, debido a que solo se cursó en una cuenta de la plataforma Udemy.

G.2 Concentración de recursos para estudiar

Libros

- Applied Deep Learning A Case-Based Approach to Understanding Deep Neural Networks Umberto Michelucci
- Fundamentals of Deep Learning - Nikhil Buduma (Cap. 5).
- Deep Learning with Keras - Antonio Gulli.

Cursos

- Deep Learning Computer Vision™ CNN, OpenCV, YOLO, SSD & GANs
    - https://www.udemy.com/course/master-deep-learning-computer-visiontm-cnn-ssd-yolo-gans/learn/lecture/11973884?components=buy_button,discount_expiration,gift_this_course,introduction_asset,purchase,deal_badge,redeem_coupon#overview
- MIT 6.S191: Introduction to Deep Learning - https://www.youtube.com/playlist?list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI
- Introduction to Deep Learning & Neural Networks with Keras by IBM - https://www.coursera.org/learn/introduction-to-deep-learning-with-keras/home/welcome
- Deep Learning Crash Course - https://www.youtube.com/playlist?list=PLWKotBjTDoLj3rXBL-nEIPRN9V3a9Cx07
- Curso fundamental de ML - https://www.udemy.com/course/curso-machine-learning/

Papers

- A Survey of the Recent Architectures of Deep Convolutional Neural Networks - https://arxiv.org/abs/1901.06032.
- Benchmarking State-of-the-Art Deep Learning Software Tools - https://ieeexplore.ieee.org/abstract/document/7979887
- ImageNet classification with deep convolutional neural networks - http://dl.acm.org.conricyt.remotexs.co/citation.cfm?id=3065386
- Recent advances in convolutional neural networks - http://www.sciencedirect.com.conricyt.remotexs.co/science/article/pii/S0031320317304120

Lecturas

- A Guide to Building Convolutional Neural Networks from Scratch - https://towardsdatascience.com/a-guide-for-building-convolutional-neural-networks-e4eefd17f4fd
- A Guide for Building Convolutional Neural Networks - https://towardsdatascience.com/a-guide-for-building-convolutional-neural-networks-e4eefd17f4fd

- Build your first Convolutional Neural Network to recognize images - https://medium.com/intuitive-deep-learning/build-your-first-convolutional-neural-network-to-recognize-images-84b9c78fe0ce
- “Bag of Tricks for Image Classification with Convolutional Neural Networks”: Paper Discussion - https://hackernoon.com/bag-of-tricks-for-image-classification-with-convolutional-neural-networks-paper-discussion-693c9e17d1cc

Videos

- A friendly introduction to Convolutional Neural Networks and Image Recognition - https://www.youtube.com/watch?v=2-Ol7ZB0MmU
- Lecture 5 | Convolutional Neural Networks - https://www.youtube.com/watch?v=bNb2fEVKeEo
- Redes Neuronales Convolucionales (CNN) - https://www.youtube.com/watch?v=DwBJx76KNBc
- Convolutional Neural Networks - The Math of Intelligence (Week 4) - https://www.youtube.com/watch?v=FTr3n7uBIuE
- 32. ImageNet is a Convolutional Neural Network (CNN), The Convolution Rule - https://www.youtube.com/watch?v=hwDRfkPSXng&t=2s

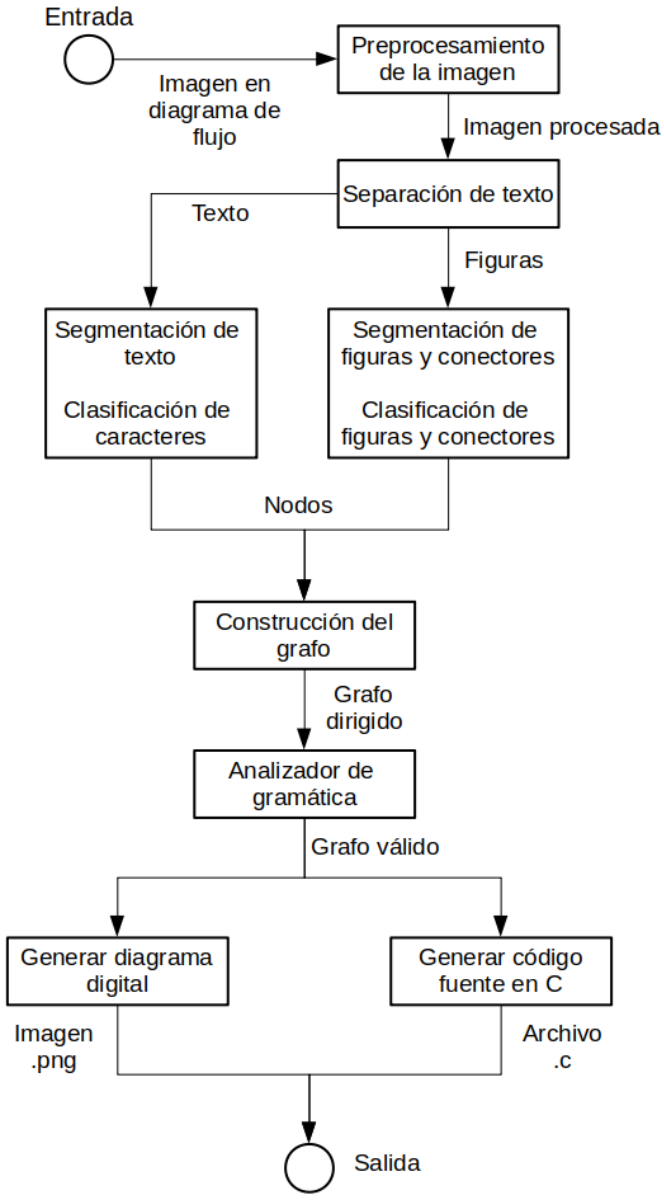
Slides

- Introduction to Convolutional Neural Networks
- R- CNN for Object Detection
- Slides Deep learning in computer vision - Caner Hazirbar

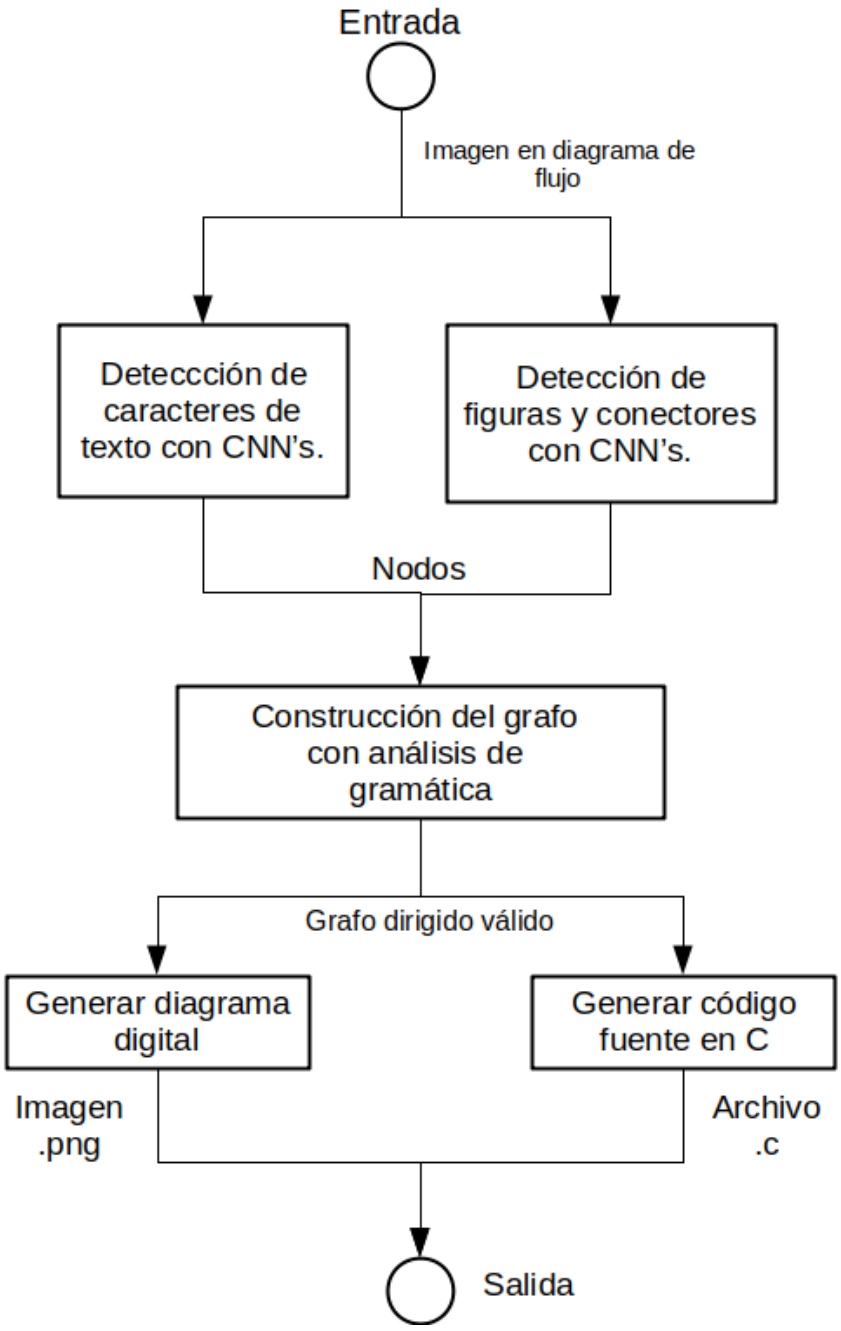
Apéndice H

Aquí se encuentran diferentes propuestas del pipeline para resolver el problema de reconocimiento de diagramas de flujo hechos a mano.

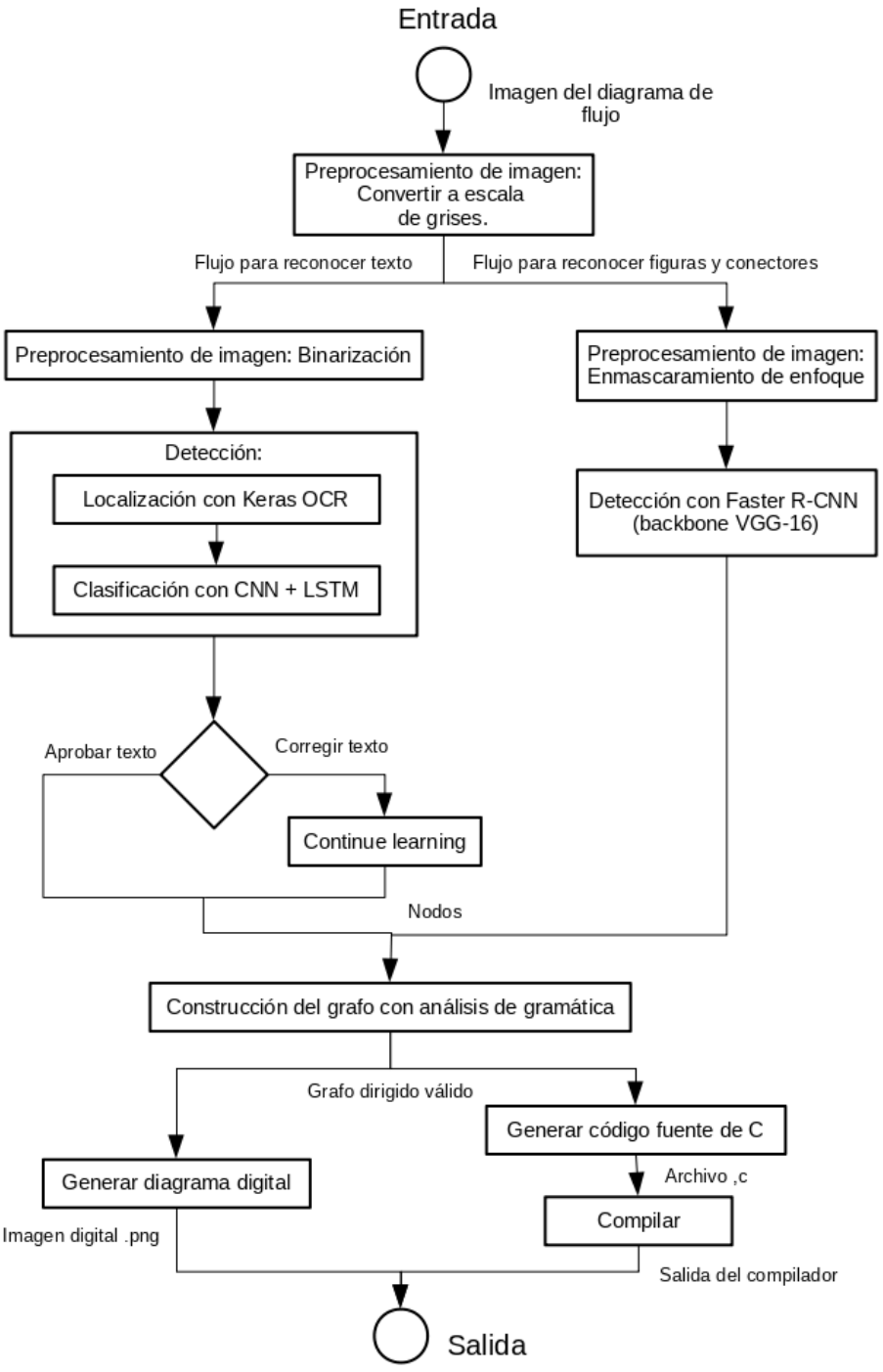
H.1 Pipeline versión planeada



H.2 Pipeline de la versión 1 del sistema



H.3 Pipeline de la versión 2 (final) del sistema

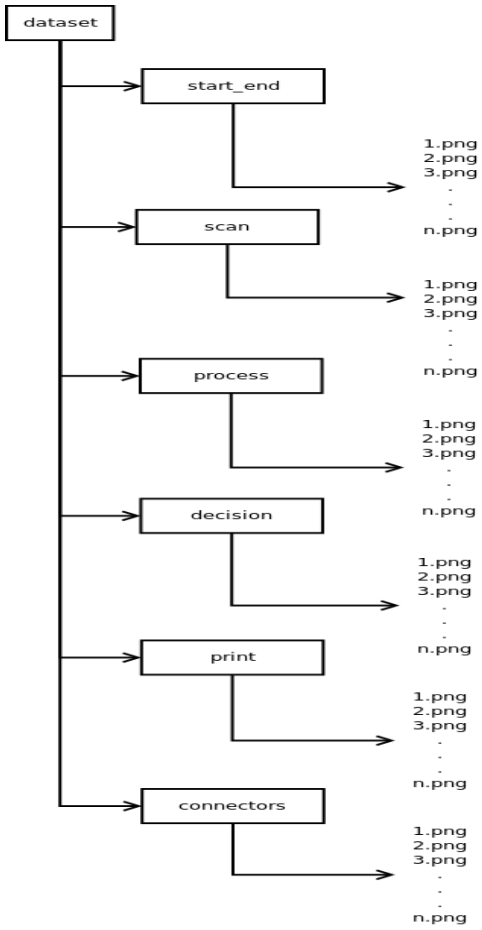


Apéndice I

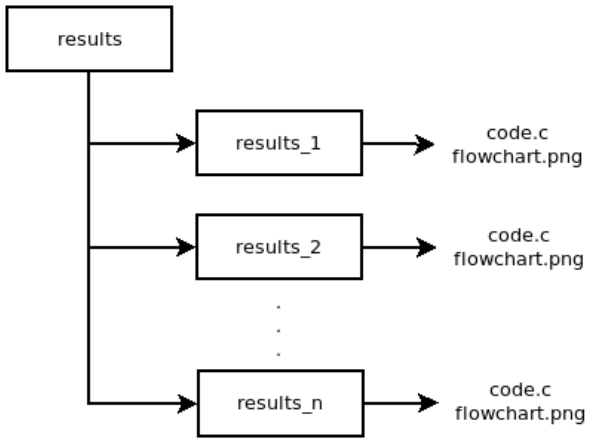
Aquí se encuentran los diagramas de persistencia de datos propuestos para el uso de archivos e interacción con del conjunto de datos.

I.1 Diagramas de persistencia de datos versión planeada

Dataset para entrenar modelo de figuras y conectores:

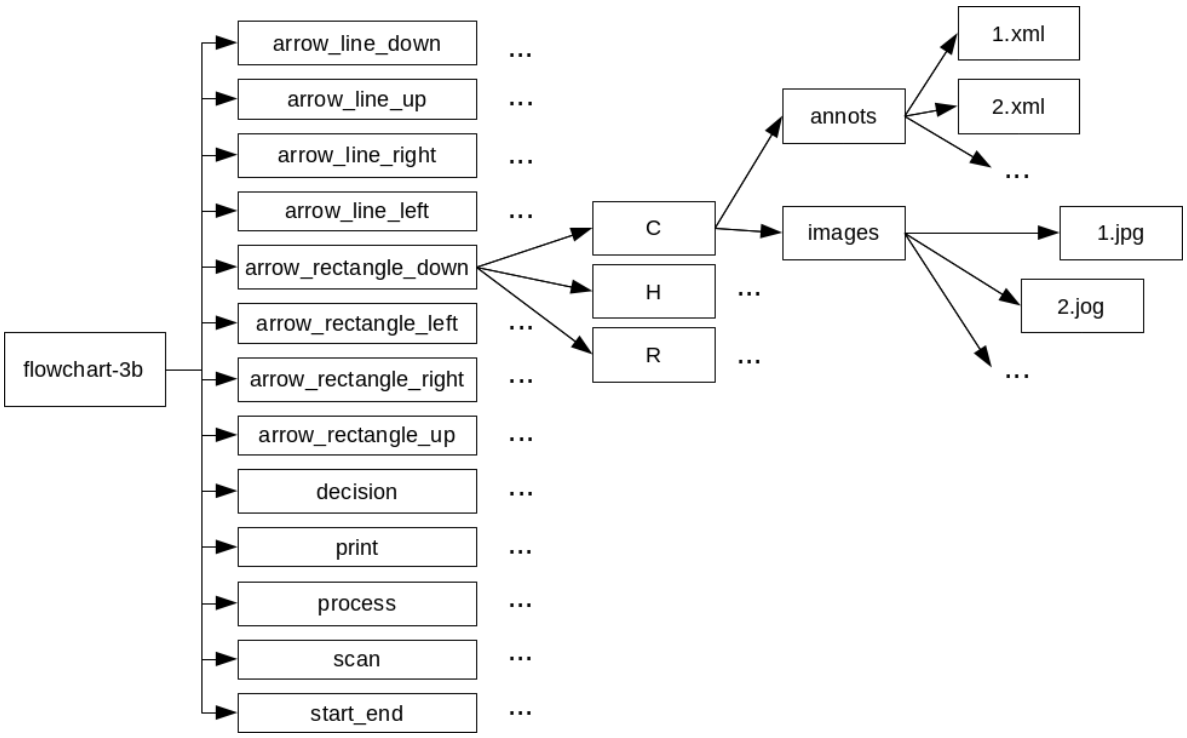


Estructura del directorio de archivos de resultados:

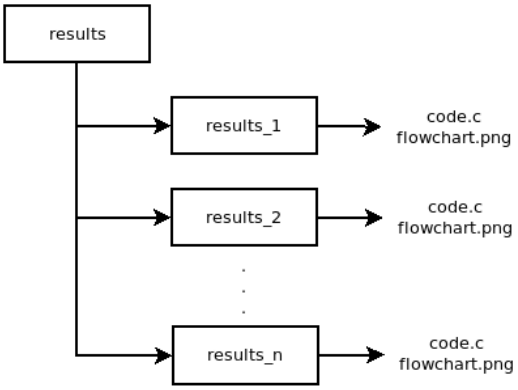


I.2 Diagramas de persistencia de datos de la versión 1 del sistema

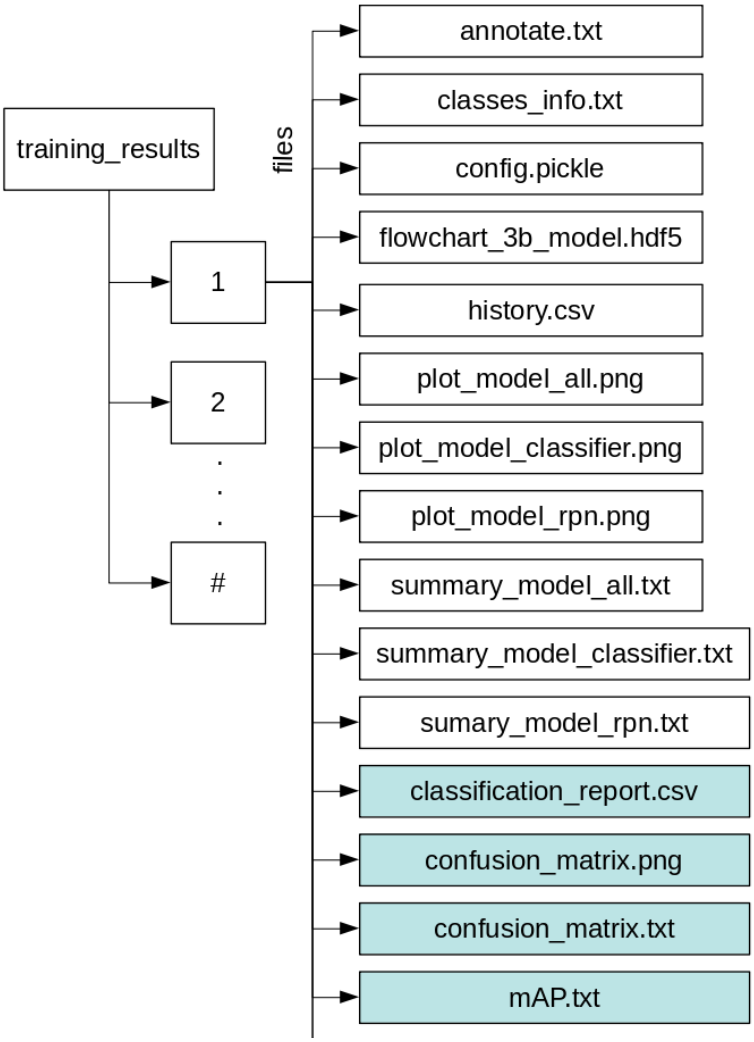
Dataset para entrenar modelo de figuras y conectores, en la primer versión del sistema:



Estructura del directorio de archivos de resultados:

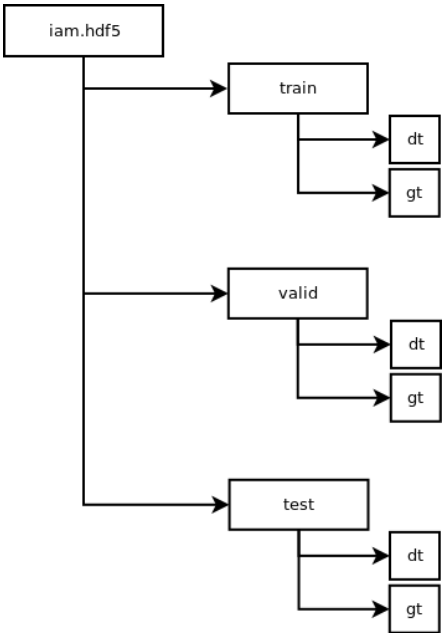


Estructura del directorio donde se almacenan los archivos para los modelos de detección de figuras y conectores:

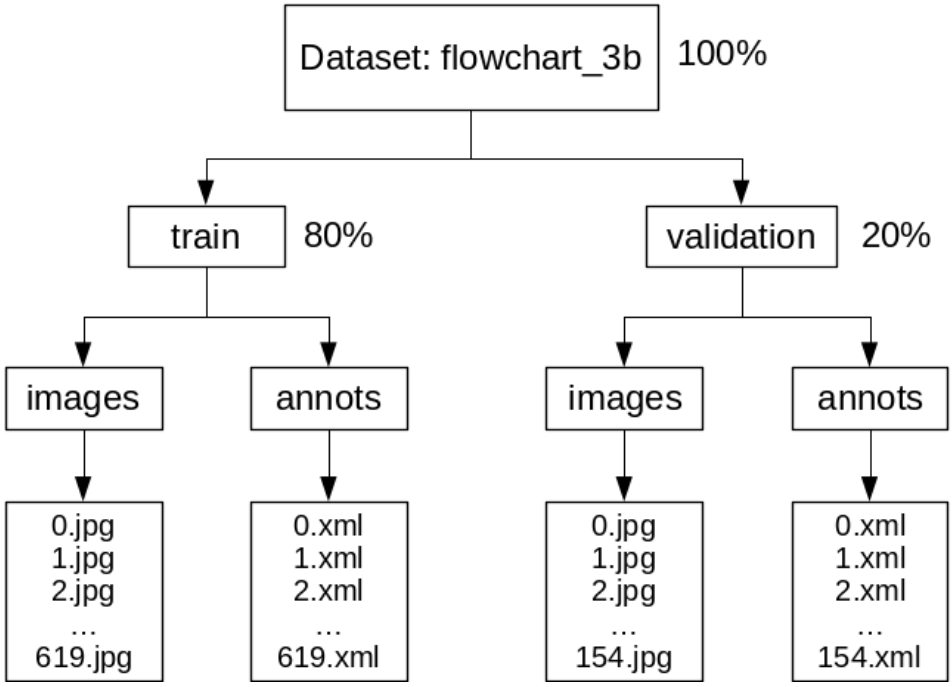


I.3 Diagramas de persistencia de datos de la versión 2 (final) del sistema

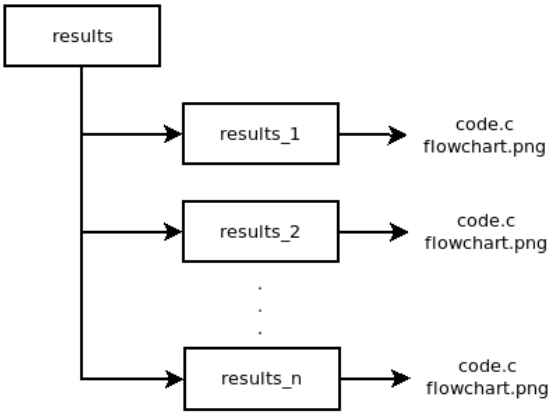
Estructura del archivo dataset que emplea el modelo de texto:



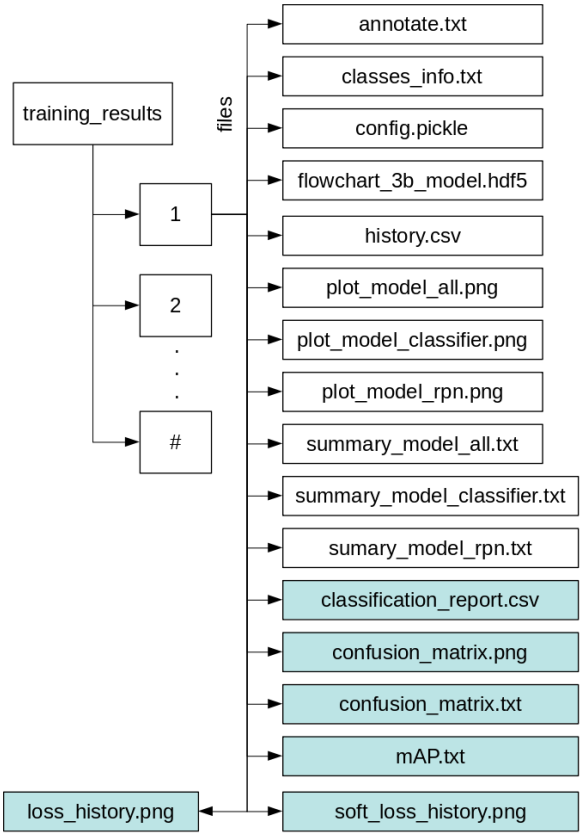
Estructura del dataset empleado para el entrenamiento del modelo de figuras y conectores:



Estructura del directorio de archivos de resultados:



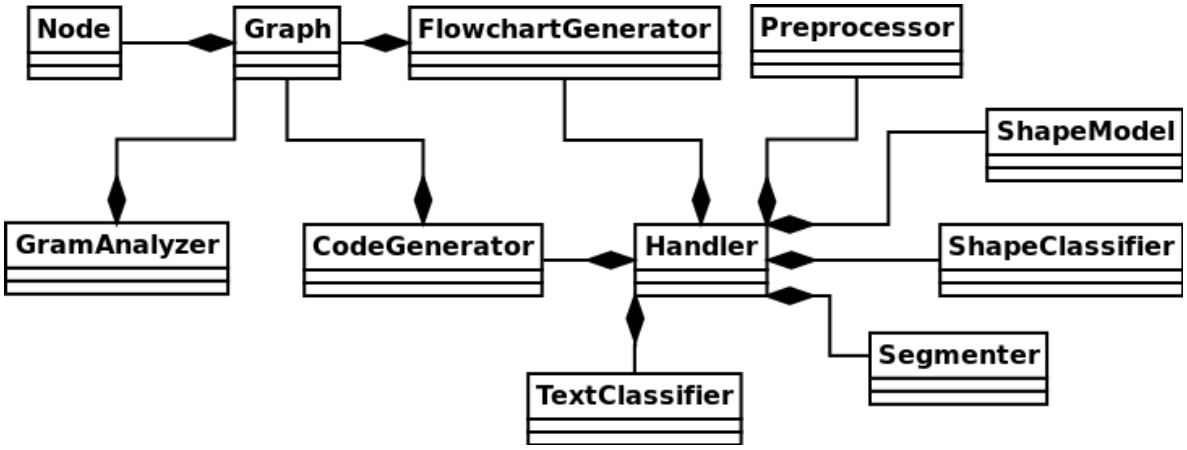
Estructura del directorio donde se almacenan los archivos para los modelos de detección de figuras y conectores.

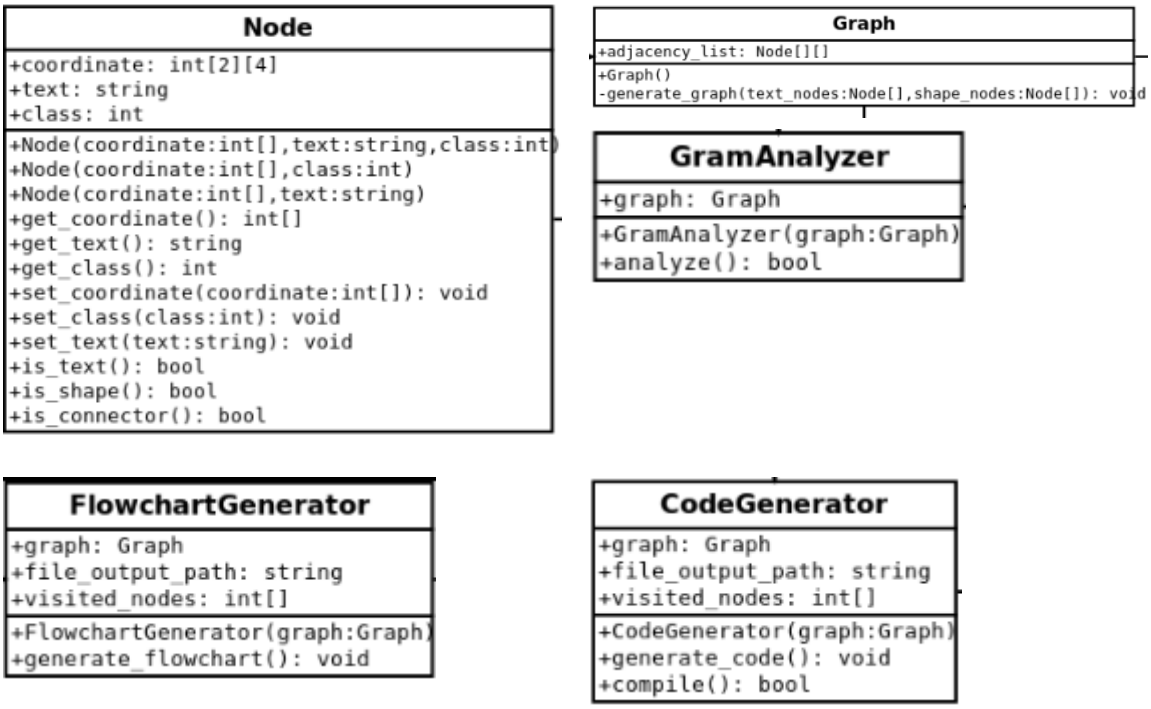


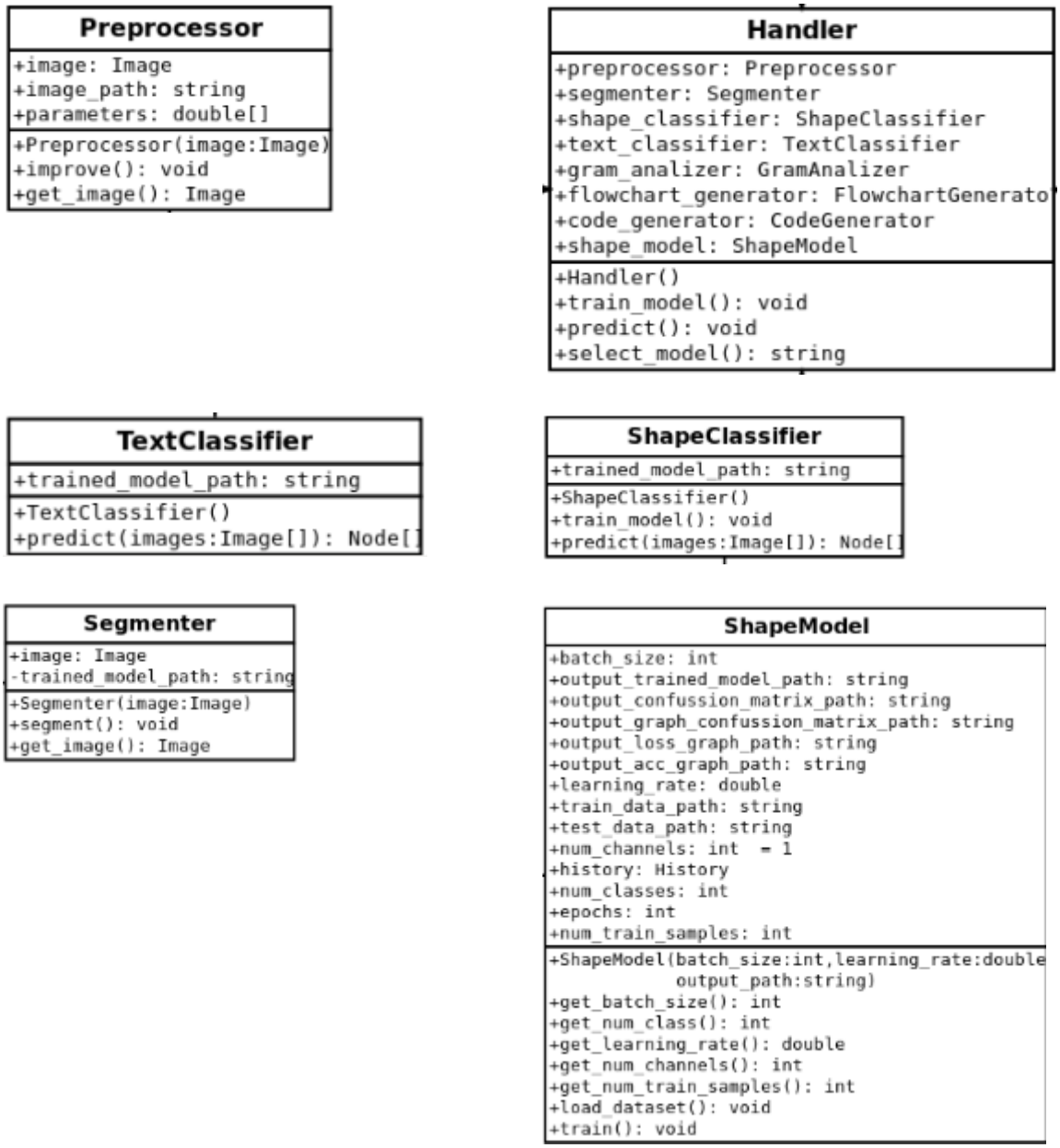
Apéndice J

Se muestra en dos partes, una donde se ven solo las relaciones y otra donde se muestran por separado las clases para una mejor visualización.

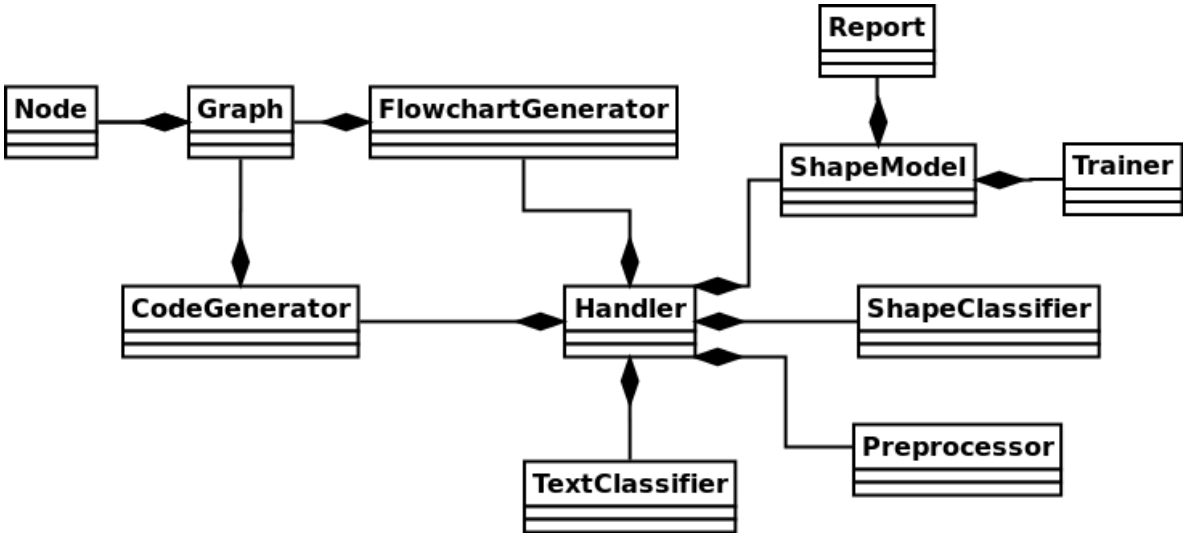
J.1 Diagrama de clases versión planeada

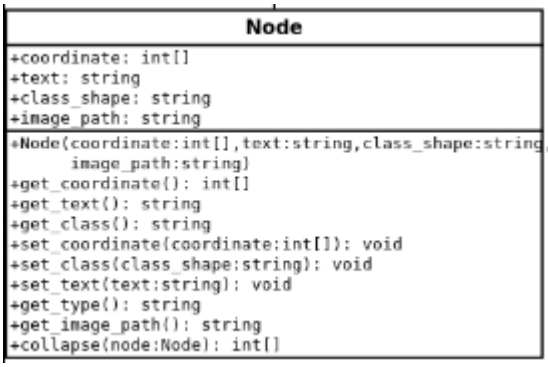


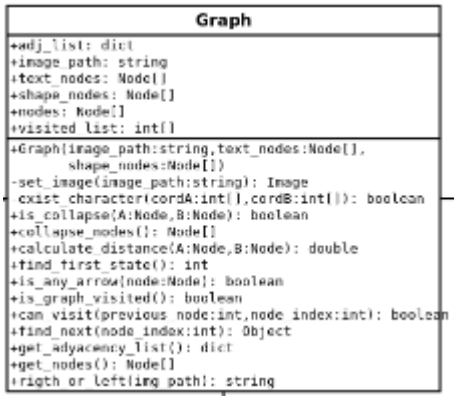


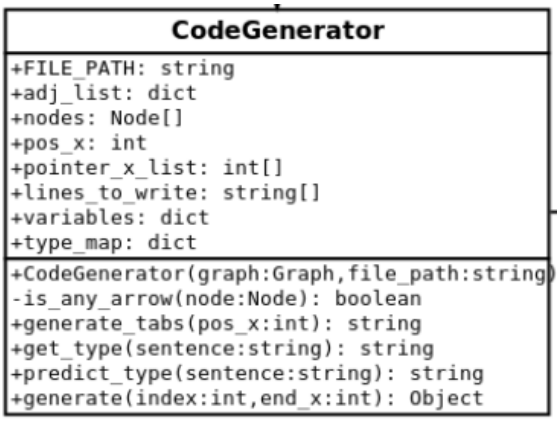

Preprocessor
+image: Image
+image_path: string
+parameters: double[]
+Preprocessor(image:Image)
+improve(): void
+get_image(): Image
Handler
+preprocessor: Preprocessor
+segmenter: Segmenter
+shape_classifier: ShapeClassifier
+text_classifier: TextClassifier
+gram_analizer: GramAnalizer
+flowchart_generator: FlowchartGenerato
+code_generator: CodeGenerator
+shape_model: ShapeModel
+Handler()
+train_model(): void
+predict(): void
+select_model(): string
TextClassifier
+trained_model_path: string
+TextClassifier()
+predict(images:Image[]): Node[]
ShapeClassifier
+trained_model_path: string
+ShapeClassifier()
+train_model(): void
+predict(images:Image[]): Node[]
Segmenter
+image: Image
-trained_model_path: string
+Segmenter(image:Image)
+segment(): void
+get_image(): Image
ShapeModel
+batch_size: int
+output_trained_model_path: string
+output_confussion_matrix_path: string
+output_graph_confussion_matrix_path: string
+output_loss_graph_path: string
+output_acc_graph_path: string
+learning_rate: double
+train_data_path: string
+test_data_path: string
+num_channels: int = 1
+history: History
+num_classes: int
+epochs: int
+num_train_samples: int
+ShapeModel(batch_size:int,learning_rate:double
output_path:string)
+get_batch_size(): int
+get_num_class(): int
+get_learning_rate(): double
+get_num_channels(): int
+get_num_train_samples(): int
+load_dataset(): void
+train(): void

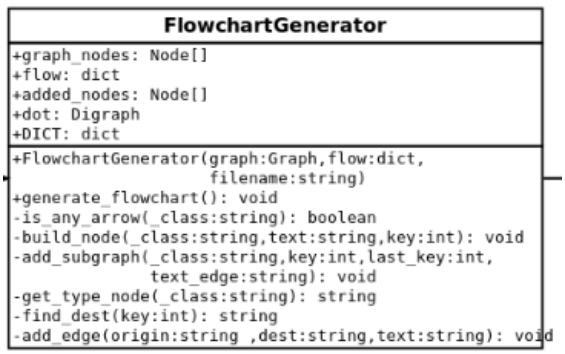
J.2 Diagrama de clases de la versión 1 del sistema

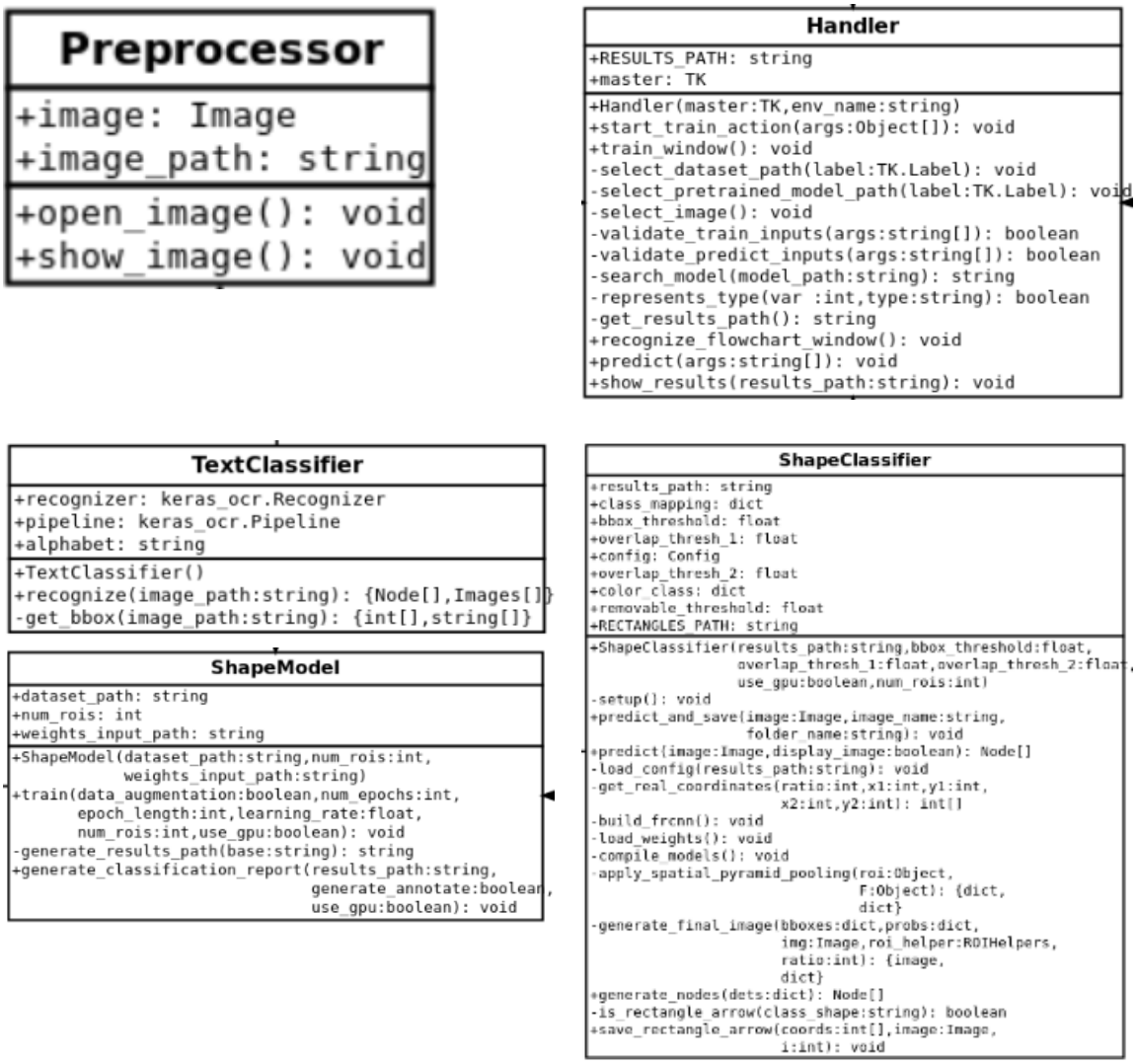




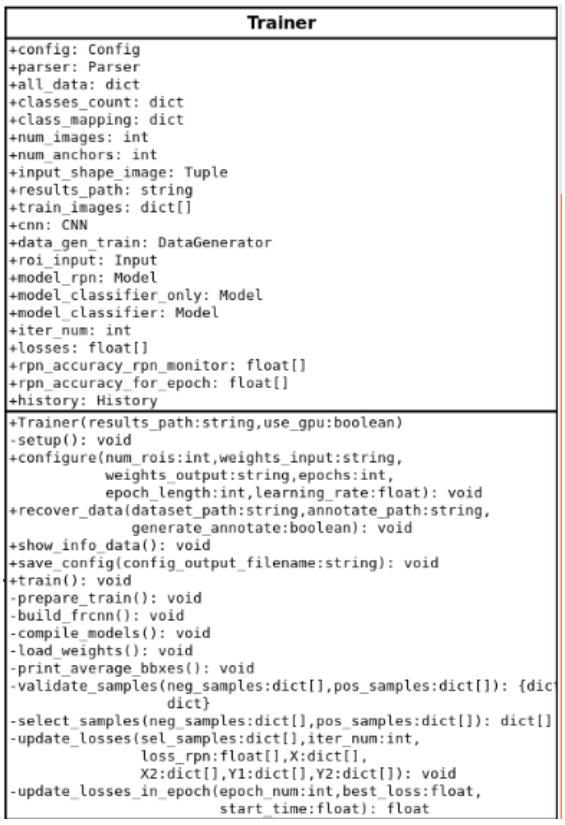




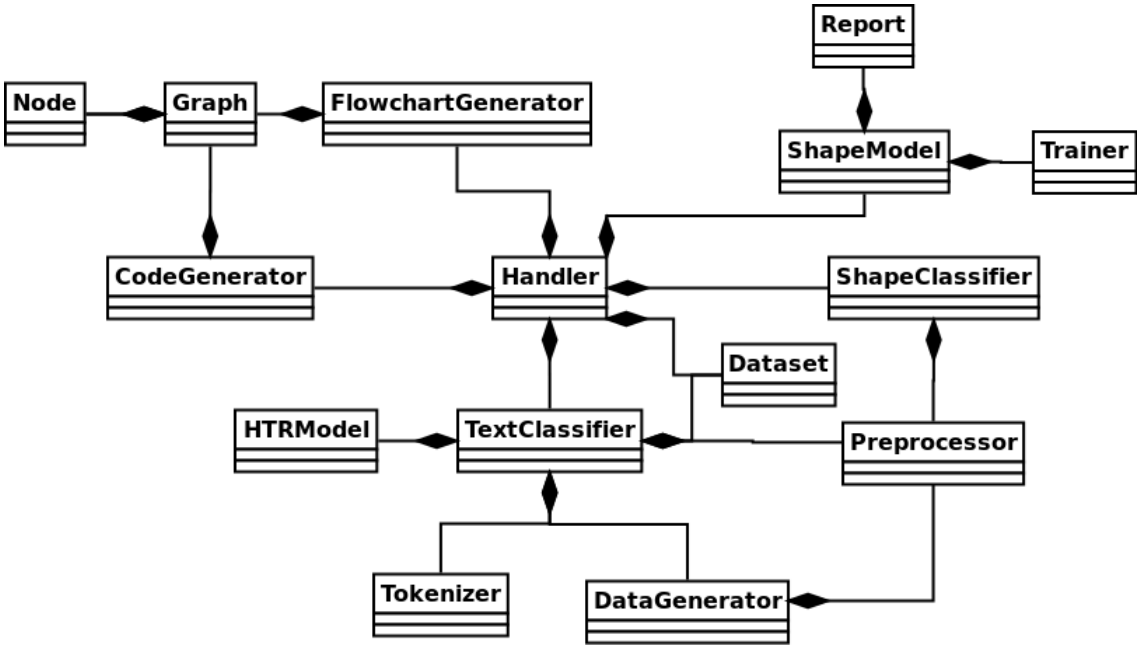


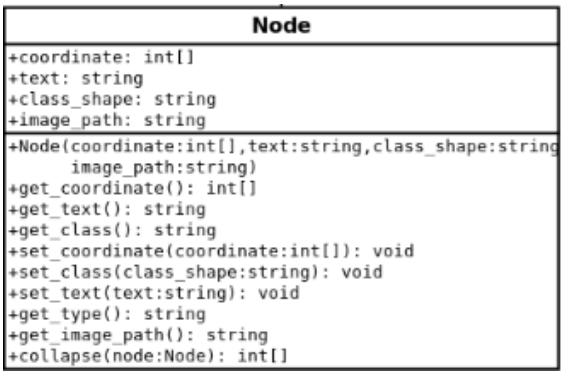

Preprocessor
+image: Image
+image_path: string
+open_image(): void
+show_image(): void
Handler
+RESULTS_PATH: string
+master: TK
+Handler(master:TK,env_name:string)
+start_train_action(args:Object[]): void
+train_window(): void
-select_dataset_path(label:TK.Label): void
-select_pretrained_model_path(label:TK.Label): void
-select_image(): void
-validate_train_inputs(args:string[]): boolean
-validate_predict_inputs(args:string[]): boolean
-search_model(model_path:string): string
-represents_type(var :int,type:string): boolean
-get_results_path(): string
+recognize_flowchart_window(): void
+predict(args:string[]): void
+show_results(results_path:string): void
TextClassifier
+recognizer: keras_ocr.Recognizer
+pipeline: keras_ocr.Pipeline
+alphabet: string
+TextClassifier()
+recognize(image_path:string): {Node[],Images[]}
-get_bbox(image_path:string): {int[],string[]}
ShapeModel
+dataset_path: string
+num_rois: int
+weights_input_path: string
+ShapeModel(dataset_path:string,num_rois:int,
weights_input_path:string)
+train(data_augmentation:boolean,num_epochs:int,
epoch_length:int,learning_rate:float,
num_rois:int,use_gpu:boolean): void
-generate_results_path(base:string): string
+generate_classification_report(results_path:string,
generate_annotate:boolean,
use_gpu:boolean): void
ShapeClassifier
+results_path: string
+class_mapping: dict
+bbox_threshold: float
+overlap_thresh_1: float
+config: Config
+overlap_thresh_2: float
+color_class: dict
+removable_threshold: float
+RECTANGLES_PATH: string
+ShapeClassifier(results_path:string,bbox_threshold:float,
overlap_thresh_1:float,overlap_thresh_2:float,
use_gpu:boolean,num_rois:int)
-setup(): void
+predict_and_save(image:Image,image_name:string,
folder_name:string): void
+predict(image:Image,display_image:boolean): Node[]
-load_config(results_path:string): void
-get_real_coordinates(ratio:int,x1:int,y1:int,
x2:int,y2:int): int[]
-build_frcnn(): void
-load_weights(): void
-compile_models(): void
-apply_spatial_pyramid_pooling(roi:Object,
F:Object): {dict,
dict}
-generate_final_image(bboxes:dict,probs:dict,
img:Image,roi_helper:ROIHelpers,
ratio:int): {image,
dict}
+generate_nodes(dets:dict): Node[]
-is_rectangle_arrow(class_shape:string): boolean
+save_rectangle_arrow(coords:int[],image:Image,
i:int): void

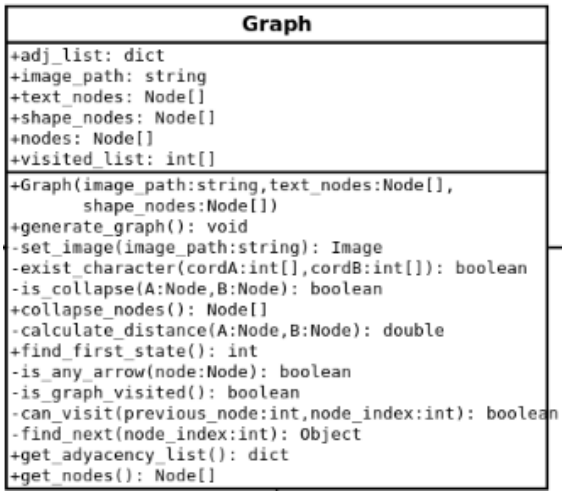

Report
+results_path: string
+annotate_path: string
+config: Object
+IOU_THRESHOLD: float
+class_mapping: dict
+cnf_matrix: int[][]
+model_rpn: Model
+model_classifier_only: Model
+model_classifier: Model
+test_images: dict
+Report(results_path:string,dataset_path:string,
generate_annotate:boolean,use_gpu:boolean)
-setup(): void
-load_config(): void
-build_frcnn(): void
-load_weights(): void
-compile_models(): void
-load_data(dataset_path:string,generate_annotate:boolean): void
+generate(): void
-apply_spatial_pyramid_pooling(roi:ROIHelpers,
F:Object): {dict,
dict}
-apply_non_max_sup_for_each_class(bboxes:dict,
probs:dict,
roi_helper:ROIHelpers): dict
-get_map(pred:dict,gt:dict,factors:int[]): {dict,
dict}
-update_confusion_matrix(pred:dict,gt:dict,
factors:int[]): void
-save_classification_report(): void

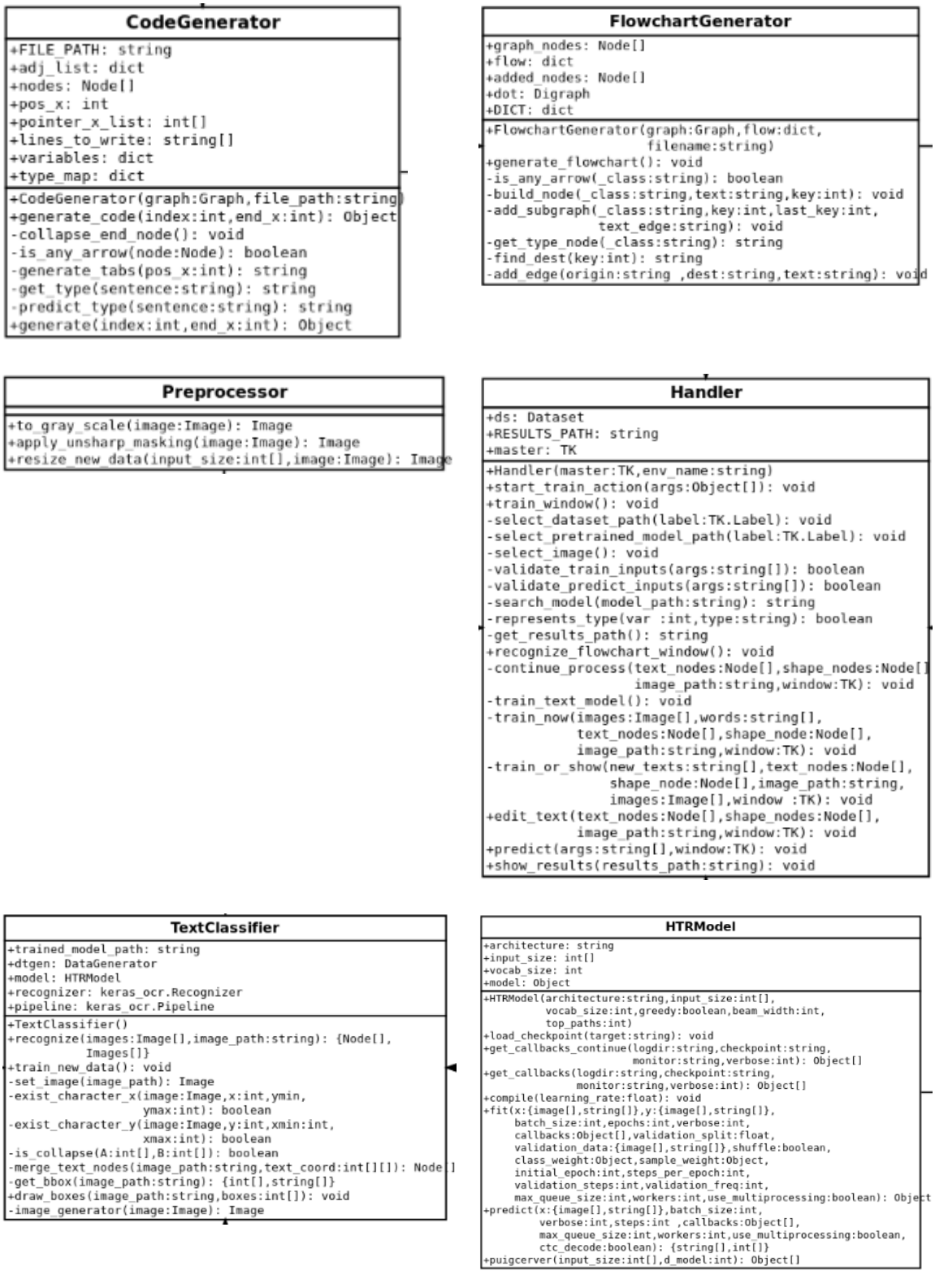

Trainer
+config: Config
+parser: Parser
+all_data: dict
+classes_count: dict
+class_mapping: dict
+num_images: int
+num_anchors: int
+input_shape_image: Tuple
+results_path: string
+train_images: dict[]
+cnn: CNN
+data_gen_train: DataGenerator
+roi_input: Input
+model_rpn: Model
+model_classifier_only: Model
+model_classifier: Model
+iter_num: int
+losses: float[]
+rpn_accuracy_rpn_monitor: float[]
+rpn_accuracy_for_epoch: float[]
+history: History
+Trainer(results_path:string,use_gpu:boolean)
-setup(): void
+configure(num_rois:int,weights_input:string,
weights_output:string,epochs:int,
epoch_length:int,learning_rate:float): void
+recover_data(dataset_path:string,annotate_path:string,
generate_annotate:boolean): void
+show_info_data(): void
+save_config(config_output_filename:string): void
+train(): void
-prepare_train(): void
-build_frcnn(): void
-compile_models(): void
-load_weights(): void
-print_average_bbxes(): void
-validate_samples(neg_samples:dict[],pos_samples:dict[]): {dict
dict}
-select_samples(neg_samples:dict[],pos_samples:dict[]): dict[]
-update_losses(sel_samples:dict[],iter_num:int,
loss_rpn:float[],X:dict[],
X2:dict[],Y1:dict[],Y2:dict[]): void
-update_losses_in_epoch(epoch_num:int,best_loss:float,
start_time:float): float

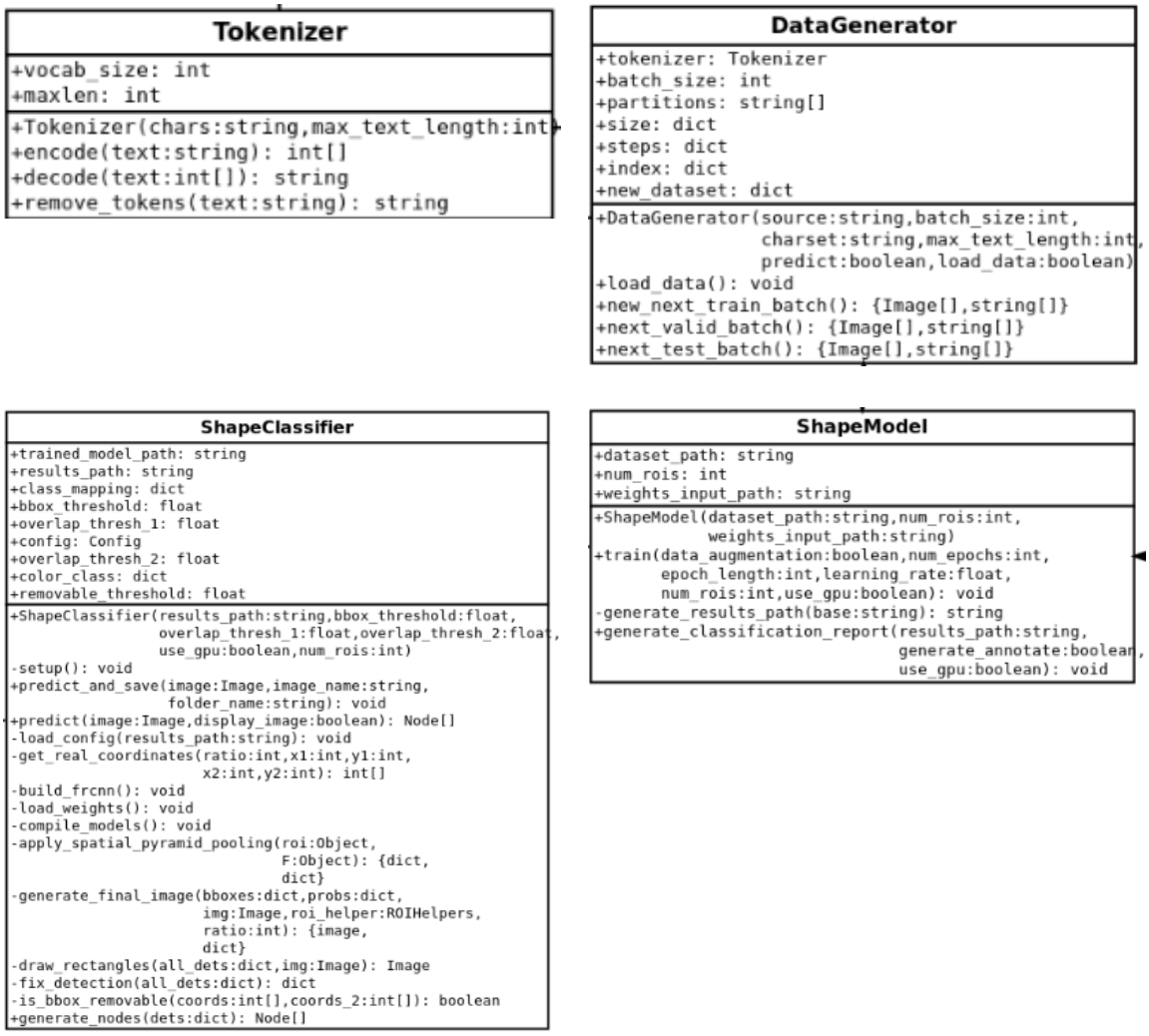
J.3 Diagrama de clases de la versión 2 (final) del sistema

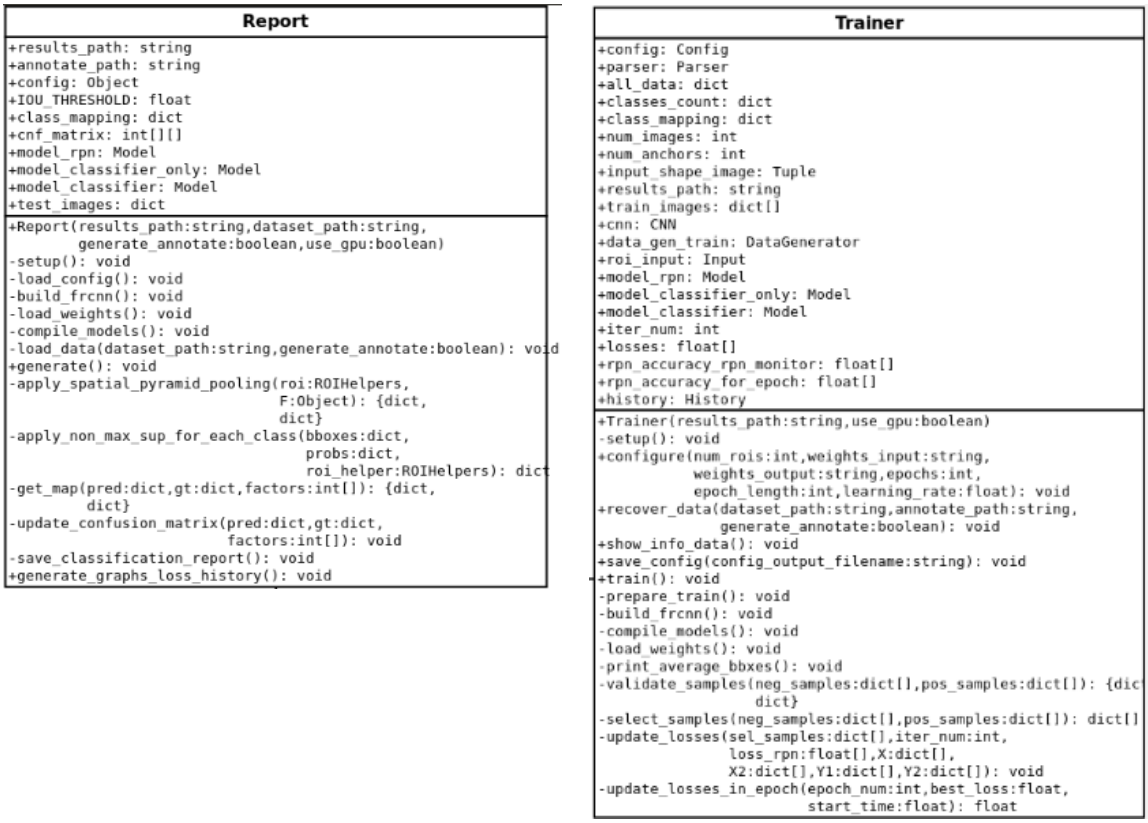







CodeGenerator
+FILE_PATH: string
+adj_list: dict
+nodes: Node[]
+pos_x: int
+pointer_x_list: int[]
+lines_to_write: string[]
+variables: dict
+type_map: dict
+CodeGenerator(graph:Graph,file_path:string)
+generate_code(index:int,end_x:int): Object
-collapse_end_node(): void
-is_any_arrow(node:Node): boolean
-generate_tabs(pos_x:int): string
-get_type(sentence:string): string
-predict_type(sentence:string): string
+generate(index:int,end_x:int): Object
FlowchartGenerator
+graph_nodes: Node[]
+flow: dict
+added_nodes: Node[]
+dot: Digraph
+DICT: dict
+FlowchartGenerator(graph:Graph,flow:dict,
filename:string)
+generate_flowchart(): void
-is_any_arrow(_class:string): boolean
-build_node(_class:string,text:string,key:int): void
-add_subgraph(_class:string,key:int,last_key:int,
text_edge:string): void
-get_type_node(_class:string): string
-find_dest(key:int): string
-add_edge(origin:string ,dest:string,text:string): void
Preprocessor
+to_gray_scale(image:Image): Image
+apply_unsharp_masking(image:Image): Image
+resize_new_data(input_size:int[],image:Image): Image
Handler
+ds: Dataset
+RESULTS_PATH: string
+master: TK
+Handler(master:TK,env_name:string)
+start_train_action(args:Object[]): void
+train_window(): void
-select_dataset_path(label:TK.Label): void
-select_pretrained_model_path(label:TK.Label): void
-select_image(): void
-validate_train_inputs(args:string[]): boolean
-validate_predict_inputs(args:string[]): boolean
-search_model(model_path:string): string
-represents_type(var :int,type:string): boolean
-get_results_path(): string
+recognize_flowchart_window(): void
-continue_process(text_nodes:Node[],shape_nodes:Node[]
image_path:string,window:TK): void
-train_text_model(): void
-train_now(images:Image[],words:string[],
text_nodes:Node[],shape_node:Node[],
image_path:string,window:TK): void
-train_or_show(new_texts:string[],text_nodes:Node[],
shape_node:Node[],image_path:string,
images:Image[],window :TK): void
+edit_text(text_nodes:Node[],shape_nodes:Node[],
image_path:string,window:TK): void
+predict(args:string[],window:TK): void
+show_results(results_path:string): void
TextClassifier
+trained_model_path: string
+dtgen: DataGenerator
+model: HTRModel
+recognizer: keras_ocr.Recognizer
+pipeline: keras_ocr.Pipeline
+TextClassifier()
+recognize(images:Image[],image_path:string): {Node[],
Images[]}
+train_new_data(): void
-set_image(image_path): Image
-exist_character_x(image:Image,x:int,ymin,
ymax:int): boolean
-exist_character_y(image:Image,y:int,xmin:int,
xmax:int): boolean
-is_collapse(A:int[],B:int[]): boolean
-merge_text_nodes(image_path:string,text_coord:int[][]): Node[]
-get_bbox(image_path:string): {int[],string[]}
+draw_boxes(image_path:string,boxes:int[]): void
-image_generator(image:Image): Image
HTRModel
+architecture: string
+input_size: int[]
+vocab_size: int
+model: Object
+HTRModel(architecture:string,input_size:int[],
vocab_size:int,greedy:boolean,beam_width:int,
top_paths:int)
+load_checkpoint(target:string): void
+get_callbacks_continue(logdir:string,checkpoint:string,
monitor:string,verbose:int): Object[]
+get_callbacks(logdir:string,checkpoint:string,
monitor:string,verbose:int): Object[]
+compile(learning_rate:float): void
+fit(x:{image[],string[]},y:{image[],string[]},
batch_size:int,epochs:int,verbose:int,
callbacks:Object[],validation_split:float,
validation_data:{image[],string[]},shuffle:boolean,
class_weight:Object,sample_weight:Object,
initial_epoch:int,steps_per_epoch:int,
validation_steps:int,validation_freq:int,
max_queue_size:int,workers:int,use_multiprocessing:boolean): Object
+predict(x:{image[],string[]},batch_size:int,
verbose:int,steps:int ,callbacks:Object[],
max_queue_size:int,workers:int,use_multiprocessing:boolean,
ctc_decode:boolean): {string[],int[]}
+puigcerver(input_size:int[],d_model:int): Object[]


Tokenizer
+vocab_size: int
+maxlen: int
+Tokenizer(chars:string,max_text_length:int)
+encode(text:string): int[]
+decode(text:int[]): string
+remove_tokens(text:string): string
DataGenerator
+tokenizer: Tokenizer
+batch_size: int
+partitions: string[]
+size: dict
+steps: dict
+index: dict
+new_dataset: dict
+DataGenerator(source:string,batch_size:int,
charset:string,max_text_length:int,
predict:boolean,load_data:boolean)
+load_data(): void
+new_next_train_batch(): {Image[],string[]}
+next_valid_batch(): {Image[],string[]}
+next_test_batch(): {Image[],string[]}
ShapeClassifier
+trained_model_path: string
+results_path: string
+class_mapping: dict
+bbox_threshold: float
+overlap_thresh_1: float
+config: Config
+overlap_thresh_2: float
+color_class: dict
+removable_threshold: float
+ShapeClassifier(results_path:string,bbox_threshold:float,
overlap_thresh_1:float,overlap_thresh_2:float,
use_gpu:boolean,num_rois:int)
-setup(): void
+predict_and_save(image:Image,image_name:string,
folder_name:string): void
+predict(image:Image,display_image:boolean): Node[]
-load_config(results_path:string): void
-get_real_coordinates(ratio:int,x1:int,y1:int,
x2:int,y2:int): int[]
-build_frcnn(): void
-load_weights(): void
-compile_models(): void
-apply_spatial_pyramid_pooling(roi:Object,
F:Object): {dict,
dict}
-generate_final_image(bboxes:dict,probs:dict,
img:Image,roi_helper:ROIHelpers,
ratio:int): {image,
dict}
-draw_rectangles(all_dets:dict,img:Image): Image
-fix_detection(all_dets:dict): dict
-is_bbox_removable(coords:int[],coords_2:int[]): boolean
+generate_nodes(dets:dict): Node[]
ShapeModel
+dataset_path: string
+num_rois: int
+weights_input_path: string
+ShapeModel(dataset_path:string,num_rois:int,
weights_input_path:string)
+train(data_augmentation:boolean,num_epochs:int,
epoch_length:int,learning_rate:float,
num_rois:int,use_gpu:boolean): void
-generate_results_path(base:string): string
+generate_classification_report(results_path:string,
generate_annotate:boolean,
use_gpu:boolean): void


Report
+results_path: string
+annotate_path: string
+config: Object
+IOU_THRESHOLD: float
+class_mapping: dict
+cnf_matrix: int[][]
+model_rpn: Model
+model_classifier_only: Model
+model_classifier: Model
+test_images: dict
+Report(results_path:string,dataset_path:string,
generate_annotate:boolean,use_gpu:boolean)
-setup(): void
-load_config(): void
-build_frcnn(): void
-load_weights(): void
-compile_models(): void
-load_data(dataset_path:string,generate_annotate:boolean): void
+generate(): void
-apply_spatial_pyramid_pooling(roi:ROIHelpers,
F:Object): {dict,
dict}
-apply_non_max_sup_for_each_class(bboxes:dict,
probs:dict,
roi_helper:ROIHelpers): dict
-get_map(pred:dict,gt:dict,factors:int[]): {dict,
dict}
-update_confusion_matrix(pred:dict,gt:dict,
factors:int[]): void
-save_classification_report(): void
+generate_graphs_loss_history(): void
Trainer
+config: Config
+parser: Parser
+all_data: dict
+classes_count: dict
+class_mapping: dict
+num_images: int
+num_anchors: int
+input_shape_image: Tuple
+results_path: string
+train_images: dict[]
+cnn: CNN
+data_gen_train: DataGenerator
+roi_input: Input
+model_rpn: Model
+model_classifier_only: Model
+model_classifier: Model
+iter_num: int
+losses: float[]
+rpn_accuracy_rpn_monitor: float[]
+rpn_accuracy_for_epoch: float[]
+history: History
+Trainer(results_path:string,use_gpu:boolean)
-setup(): void
+configure(num_rois:int,weights_input:string,
weights_output:string,epochs:int,
epoch_length:int,learning_rate:float): void
+recover_data(dataset_path:string,annotate_path:string,
generate_annotate:boolean): void
+show_info_data(): void
+save_config(config_output_filename:string): void
+train(): void
-prepare_train(): void
-build_frcnn(): void
-compile_models(): void
-load_weights(): void
-print_average_bbxes(): void
-validate_samples(neg_samples:dict[],pos_samples:dict[]): {dict
dict}
-select_samples(neg_samples:dict[],pos_samples:dict[]): dict[]
-update_losses(sel_samples:dict[],iter_num:int,
loss_rpn:float[],X:dict[],
X2:dict[],Y1:dict[],Y2:dict[]): void
-update_losses_in_epoch(epoch_num:int,best_loss:float,
start_time:float): float

Apéndice K

Aquí se encuentran los 3 tipos de pruebas efectuadas para los dos ciclos de construcción llevados a cabo, es relevante mencionar que no todas las pruebas unitarias se hacen para la segunda versión dado que no hubo modificaciones en algunos módulos. Además, que algunas otras pruebas solo se hicieron para la segunda versión.

Sistema: Reconocimiento de diagramas de flujo trazados a mano usando redes neuronales convolucionales.

Técnica de las pruebas: Caja negra.

Ambientes de pruebas:

ID	**Descripción del ambiente**
AP1	Software: Conda 4.7.12, Tensorflow 2.1.0, Keras 2.3.1 SO: Ubuntu 18.04 LTS, Equipo: 16 GB RAM, 123 GB SSD, Intel CORE i7-9700, GPU GeForce GTX 1660.
AP2	Software: Conda 4.8.0, Tensorflow 2.1.0, Keras 2.3.1 SO: Ubuntu 18.04 LTS, Equipo: 8 GB RAM, 1 TB HDD, Intel CORE i3-5005U CPU @ 2.00 GHz..
AP3	Software: Colab de Google. Equipo: GPU Tesla P100, RAM 12.72 GB y disco 68.4 GB.

Versión del sistema:

- VS1: Versión 1.
- VS2: Versión 2.

K.1 Pruebas unitarias

Preprocessor

EPU 001	**Módulo:** Preprocessor	**Autor:** OFCG
Fecha: 29/01/2020	**Requerimiento(s):** RF1, RF2	**ID ambiente:** AP2
Objetivo: Verificar el funcionamiento de las funciones para preparar la imagen.		

VS2	**Tester:** DBM	**Fecha:** 25/06/2020	
Escenario: 1. Leer imagen. 2. Aplicar un algoritmo de la clase de preprocesamiento.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Imagen: Ruta correcta, formato .png - Algoritmo: Convertir a escala de grises.	Imagen en escala de grises	Correcto
2	- Imagen: Ruta correcta, formato .png - Algoritmo: Aplicar enmascaramiento de enfoque.	Imagen en escala de grises con mayor nitidez	Correcto
3	- Imagen: Ruta correcta, formato .png - Algoritmo: Preparar para clasificación de texto.	Imagen umbralizada de forma transpuesta	Correcto

Shape Classifier

EPU 004	**Módulo:** ShapeClassifier	**Autor:** OFCG
Fecha: 29/01/2020	**Requerimiento(s):** RF5	**ID ambiente:** AP2
Objetivo: Verificar el funcionamiento de las función de predecir.		

VS1	**Tester:** DBM	**Fecha:** 20/04/2020	
Escenario: 1. Se instancia un objeto de tipo ShapeClassifier, pasandole la ruta de la carpeta de resultados. 2. Se llama a la función predict, pasándole la imagen a reconocer.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**

1	- Imagen: De diagrama de flujo - Ruta de carpeta de resultados: Ruta válida.	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3]	Correcto
2	- Imagen: De diagrama de flujo - Ruta de carpeta de resultados: Ruta no válida.	Muestra error: No fue posible encontrar el archivo de configuración	Correcto
3	- Imagen: De diagrama de flujo - Ruta de carpeta de resultados: Ruta válida.	Retorna lista de nodos y además genera guarda imágenes de flechas recortadas en la carpeta images/tmp	Correcto

VS2	**Tester:** DBM	**Fecha:** 13/06/2020	
Escenario: 1. Se instancia un objeto de tipo ShapeClassifier, pasandole la ruta de la carpeta de resultados. 2. Se llama a la función predict, pasándole la imagen a reconocer.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Imagen: Diagrama “Hola mundo” con fondo de máquina. - Ruta de carpeta de resultados: Ruta no válida.	Muestra error: No fue posible encontrar el archivo de configuración.	Correcto
2	- Imagen: Diagrama de flujo: "Hola mundo" con fondo de máquina. - Ruta de carpeta de resultados: Ruta válida, donde se guarda mejor modelo entrenado	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3,...] de los conectores y figuras del diagrama.	Correcto
3	- Imagen: Diagrama de flujo: "Hola mundo" con fondo de raya. - Ruta de carpeta de resultados: Ruta válida, donde se guarda mejor modelo entrenado	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3,...] de los conectores y figuras del diagrama.	Correcto, se reconocen todos las figuras y conectores, pero sobran 3 nodos que son detecciones falsas en el fondo.
4	- Imagen: Diagrama de flujo: "Hola mundo" con fondo de cuadrícula. - Ruta de carpeta de resultados: Ruta válida, donde se guarda mejor modelo entrenado	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3,...] de los conectores y figuras del diagrama.	Correcto

5	- Imagen: Diagrama de flujo: "Factorial" con fondo de hoja de máquina. - Ruta de carpeta de resultados: Ruta válida, donde se guarda mejor modelo entrenado	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3,...] de los conectores y figuras del diagrama.	Correcto
6	- Imagen: Diagrama de flujo: "Factorial" con fondo de raya. - Ruta de carpeta de resultados: Ruta válida, donde se guarda mejor modelo entrenado	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3,...] de los conectores y figuras del diagrama.	Correcto
7	- Imagen: Diagrama de flujo: "Factorial" con fondo de cuadrícula. - Ruta de carpeta de resultados: Ruta válida, donde se guarda mejor modelo entrenado	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3,...] de los conectores y figuras del diagrama.	Correcto
8	- Imagen: Diagrama de flujo: "Cálculo de raíz cuadrada" con fondo de hoja de máquina. - Ruta de carpeta de resultados: Ruta válida, donde se guarda mejor modelo entrenado	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3,...] de los conectores y figuras del diagrama.	Correcto
9	- Imagen: Diagrama de flujo: "Cálculo de raíz cuadrada" con fondo de raya. - Ruta de carpeta de resultados: Ruta válida, donde se guarda mejor modelo entrenado	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3,...] de los conectores y figuras del diagrama.	Correcto
10	- Imagen: Diagrama de flujo: "Cálculo de raíz cuadrada" con fondo de cuadrícula. - Ruta de carpeta de resultados: Ruta válida, donde se guarda mejor modelo entrenado	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3,...] de los conectores y figuras del diagrama.	Correcto
11	- Imagen: Diagrama de flujo: "Cálculo del n-ésimo elemento de la sucesión de Fibonacci" con fondo de hoja de máquina. - Ruta de carpeta de resultados: Ruta válida, donde se guarda mejor modelo entrenado	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3,...] de los conectores y figuras del diagrama.	Correcto, aunque una flecha hacia arriba no fue reconocida, esto es de esperarse ya que no es un modelo determinístico.
12	- Imagen: Diagrama de flujo: "Cálculo del n-ésimo elemento de la sucesión de Fibonacci" con fondo de hoja de raya. - Ruta de carpeta de resultados: Ruta válida, donde se guarda mejor modelo entrenado	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3,...] de los conectores y figuras del diagrama.	Correcto, aunque una flecha hacia arriba no fue reconocida,

			esto es de esperarse ya que no es un modelo determinístico.
13	- Imagen: Diagrama de flujo: "Cálculo del n-ésimo elemento de la sucesión de Fibonacci" con fondo de hoja de cuadrícula. - Ruta de carpeta de resultados: Ruta válida, donde se guarda mejor modelo entrenado	retorna una lista de nodos, nodos = [nodo1,nodo2,nodo3,...] de los conectores y figuras del diagrama.	Correcto

Shape Model

EPU 005	**Módulo:** ShapeModel	**Autor:** DBM
Fecha: 28/01/2020	**Requerimiento(s):** RF4	**ID ambiente:** AP2, AP3
Objetivo: Verificar que se obtiene la referencia al dataset y de éste se indexan los datos de entrenamiento y validación.		

VS1	**Tester:** DBM	**Fecha:** 20/04/2020	
Escenario: 1. Instanciar un objeto de la clase "ShapeModel" pasándole los parámetros: ruta del dataset y número de regiones de interés a considerar. 2. Llamar el método para entrenar y pasarle: banderas de flip horizontal y/o vertical, número de épocas, tasa de aprendizaje y bandera de uso de la GPU. 3. Abortar la ejecución			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- dataset_path: "/home/david/Escritorio/flowchart-3b(splitter)" - num_rois: 32 - horizontal_flips: ‘False’ - vertical_flips: ‘False’ - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘False’	Imprimir: Número de instancias de entrenamiento y validación, y mapeo por clases.	Correcto
2	- dataset_path: "" - num_rois: 32 - horizontal_flips: ‘False’ - vertical_flips: ‘False’	Error: Falta la ruta de dataset y salir.	Correcto

	- num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘False’		
3	- dataset_path: "/home/david/Escritorio/flowchart-3b(splitter)" - num_rois: 32 - horizontal_flips: ‘False’ - vertical_flips: ‘False’ - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘True’	Mostrar errores: Al tratar de configurar la GPU.	Correcto

VS2	**Tester:** DBM	**Fecha:** 13/06/2020	
Escenario: 1. Instanciar un objeto de la clase "ShapeModel" pasándole los parámetros: ruta del dataset, número de regiones de interés a considerar y ruta/URL del archivo de pesos del backbone del modelo. 2. Llamar el método para entrenar y pasarle: bandera de aumento de datos, número de épocas, tasa de aprendizaje y bandera de uso de la GPU.. 3. Abortar la ejecución			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- dataset_path: "/home/david/Escritorio/flowchart_3b_v3.1" - num_rois: 32 - data_augmentation: ‘False’ - weights_input_path: "vgg19_weights_tf_dim_ordering_tf_kernels.h5" - num_epochs: 1 - learning_rate: 0.00001 - use_gpu: ‘False’	Imprimir: Número de instancias de entrenamiento y validación, y mapeo por clases.	Correcto
2	- dataset_path: "" - num_rois: 32 - data_augmentation: ‘False’ - weights_input_path: "vgg19_weights_tf_dim_ordering_tf_kernels.h5" - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘False’	Error: Falta la ruta de dataset y salir. * Se aborta el entrenamiento.	Correcto
3	- dataset_path: "/home/david/Escritorio/flowchart_3b_v3.1" - num_rois: 32 - data_augmentation: ‘True’ - weights_input_path: “” - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘True’	Mostrar errores: Al tratar de configurar la GPU, y sobre que no se cargaron los pesos. * Se aborta el entrenamiento.	Correcto

4	- dataset_path: “/content/drive/My Drive/0_TT/TT2/flowchart_3b_v3” - num_rois: 32 - data_augmentation: ‘True’ - weights_input_path: “"https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels.h5"” - num_epochs: 200 - learning_rate: 0.00001 - use_gpu: ‘True’	Imprimir: Número de instancias de entrenamiento y validación, y mapeo por clases.	Correcto

EPU 006	**Módulo:** ShapeModel	**Autor:** DBM
Fecha: 28/01/2020	**Requerimiento(s):** RF4	**ID ambiente:** AP2, AP3
Objetivo: Verificar que el error de la primer época a la última disminuye y la eficacia aumenta.		

VS1	**Tester:** DBM	**Fecha:** 20/04/2020	
Escenario: 1. Instanciar un objeto de la clase "ShapeModel" pasándole los parámetros: ruta del dataset y número de regiones de interés a considerar. 2. Llamar el método para entrenar y pasarle: banderas de flips horizontal y vertical, número de épocas, tasa de aprendizaje y bandera de uso de la GPU. 3. Generar reporte de resultados con modelo pre-entrenado, pasandole la ruta de la carpeta de resultados.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- dataset_path: ""/home/david/Escritorio/flowchart-3b(splitter)"" - num_rois: 32 - horizontal_flips: ‘False’ - vertical_flips: ‘False’ - num_epochs: 2 - learning_rate: 0.00001 - use_gpu: ‘False’ - results_path: "training_results/1/"	La pérdida del entrenamiento disminuye, se genera carpeta de resultados con pesos, configuración, info. de las clases, el archivo de anotaciones, matriz de confusión, evolución de las métricas del entrenamiento, resumen e imagen de los modelos de CNNs empleados y métricas de clasificación mAP por clases y general.	Correcto
2	- dataset_path: "" - num_rois: 32 - horizontal_flips: ‘False’	La pérdida del entrenamiento se comporta de forma	Correcto

	- vertical_flips: ‘False’ - num_epochs: 5 - learning_rate: 1 - use_gpu: ‘False’ - results_path: "training_results/1/"	inestable	
3	- dataset_path: "/home/david/Escritorio/flowchart-3b(splitter)" - num_rois: 32 - horizontal_flips: ‘False’ - vertical_flips: ‘False’ - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘False’ - results_path: "training_results/x/"	Mostrar error de que la ruta de resultados no es correcta y salir	Correcto
4	- dataset_path: "/home/david/Escritorio/flowchart-3b(splitter)" - num_rois: 64 - horizontal_flips: ‘False’ - vertical_flips: ‘False’ - num_epochs: 2 - learning_rate: 0.00001 - use_gpu: ‘False’ - results_path: "training_results/2/"	La pérdida del entrenamiento disminuye, se genera carpeta de resultados con pesos, configuración, info. de las clases, el archivo de anotaciones, matriz de confusión, evolución de las métricas del entrenamiento, resumen e imagen de los modelos de CNNs empleados y métricas de clasificación mAP por clases y general.	Correcto
5	- dataset_path: "/home/david/Escritorio/flowchart-3b(splitter)" - num_rois: 32 - horizontal_flips: ‘True’ - vertical_flips: ‘True’ - num_epochs: 2 - learning_rate: 0.00001 - use_gpu: ‘False’ - results_path: "training_results/1/"	La pérdida del entrenamiento disminuye, se genera carpeta de resultados con pesos, configuración, info. de las clases, el archivo de anotaciones, matriz de confusión, evolución de las métricas del entrenamiento, resumen e imagen de los modelos de CNNs empleados y métricas de clasificación mAP por clases y general.	Correcto

VS2	**Tester:** DBM	**Fecha:** 13/06/2020
Escenario: 1. Instanciar un objeto de la clase "ShapeModel" pasándole los parámetros: ruta del dataset, número de		

regiones de interés a considerar y ruta/url de pesos.
2. Llamar el método para entrenar y pasarle: bandera de aumento de datos, número de épocas, tasa de aprendizaje y bandera de uso de la GPU.
3. Generar o no reporte de resultados con modelo pre-entrenado, pasandole la ruta de la carpeta de resultados.

Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- dataset_path: "/home/david/Escritorio/flowchart_3b_v3.1" - num_rois: 32 - data_augmentation: 'False' - weights_input_path: "vgg19_weights_tf_dim_ordering_tf_kernels.h5" - num_epochs: 2 - learning_rate: 0.00001 - use_gpu: 'False' - results_path: "training_results/1/"	La pérdida del entrenamiento disminuye, se genera carpeta de resultados con pesos, configuración, info. de las classes, el archivo de anotaciones, matriz de confusión, evolución de las métricas del entrenamiento con 2 gráficas de la pérdida, resumen e imagen de los modelos de CNNs empleados y métricas de clasificación mAP por clases y general.	Correcto
2	- dataset_path: "" - num_rois: 32 - data_augmentation: 'False' - weights_input_path: "vgg19_weights_tf_dim_ordering_tf_kernels.h5" - num_epochs: 5 - learning_rate: 1 - use_gpu: 'False' - results_path: "training_results/1/"	La pérdida del entrenamiento se comporta de forma inestable	Correcto
3	- dataset_path: "/home/david/Escritorio/flowchart_3b_v3.1" - num_rois: 32 - data_augmentation: 'False' - weights_input_path: "" - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: 'False - results_path: "training_results/x/"	Mostrar error de que la ruta de resultados no es correcta y salir	Correcto
4	- dataset_path: "vgg19_weights_tf_dim_ordering_tf_kernels.h5" - num_rois: 32 - data_augmentation: 'True' - weights_input_path: ""https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels.h5""	La pérdida del entrenamiento disminuye, se genera carpeta de resultados con pesos, configuración, info. de las clases, el archivo de anotaciones, matriz de confusión,	Correcto

	- num_epochs: 2 - learning_rate: 0.00001 - use_gpu: ‘False’ - results_path: "training_results/2/"	evolución de las métricas del entrenamiento con 2 gráficas de la pérdida, resumen e imagen de los modelos de CNNs empleados y métricas de clasificación mAP por clases y general.	
5	- dataset_path: “/content/drive/My Drive/0_TT/TT2/flowchart_3b_v3” - num_rois: 32 - data_augmentation: ‘True’ - weights_input_path: “"https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels.h5"” - num_epochs: 2 - learning_rate: 0.00001 - use_gpu: ‘False’ - results_path: "training_results/1/"	La pérdida del entrenamiento disminuye, se genera carpeta de resultados con pesos, configuración, info. de las clases, el archivo de anotaciones, matriz de confusión, evolución de las métricas del entrenamiento con 2 gráficas de la pérdida, resumen e imagen de los modelos de CNNs empleados y métricas de clasificación mAP por clases y general.	Correcto
6	- dataset_path: “/content/drive/My Drive/0_TT/TT2/flowchart_3b_v3” - num_rois: 32 - data_augmentation: ‘True’ - weights_input_path: “"https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels.h5"” - num_epochs: 500 - learning_rate: 0.00001 - use_gpu: ‘True’ - results_path: [no se genera]	La pérdida del entrenamiento disminuye, se genera carpeta de resultados con pesos, configuración, info. de las clases y el archivo de anotaciones.	Correcto
7	- dataset_path: "/content/drive/My Drive/0_TT/TT2/" - num_rois: 32 - data_augmentation: ‘True’ - weights_input_path: “"https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels.h5"” - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘True’ - results_path: [no se genera]	Mostrar error de que los pesos no pudieron ser cargados.	Correcto

Text Classifier

EPU 007	**Módulo:** TextClassifier	**Autor:** OFCG
Fecha: 29/01/2020	**Requerimiento(s):** RF6	**ID ambiente:** AP1
Objetivo: Verificar el funcionamiento de la función "predict(image)".		

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020	
Escenario: 1. Instanciar un objeto de tipo TextClassifier 2. Llamar a la función predict, pasándole como argumento las imagen de los grupos de texto.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1		"("	Correcto
2		"20"	Correcto
3		"w"	Incorrecto, ","
4		"y"	Incorrecto, "r"
5		")"	Correcto
6		"H"	Incorrecto, "I"
7		"g."	Incorrecto, "M."

8		“,”	Correcto
9		“23.1”	Correcto
10		“_”	Correcto
11		“u”	Incorrecto, ““‘
12		“a”	Correcto
13		“.”	Correcto
14		“o”	Correcto
15		“;”	Correcto
16		“?”	Correcto
17		‘ “ ‘	Correcto
18		“G”	Correcto
19		“e”	Incorrecto, “a”
20		“!”	Correcto

21		“t”	Incorrecto, “, t”
22		“:”	“:”

Node

EPU 008	**Módulo:** Node	**Autor:** DBM
Fecha: 28/01/2020	**Requerimiento(s):** RF5, RF6 y RF7	**ID ambiente:** AP2
Objetivo: Verificar si el tipo de nodo que se construye es texto, figura o conector.		

VS1	**Tester:** DBM	**Fecha:** 21/04/2020	
Escenario: 1. Instanciar un objeto de la clase "Node" pasándole los parámetros: Las coordenadas (x1,x2,y1,y2), la clase y el texto. 2. Llamar el método "get_type()".			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Coordenadas: (34, 90, 56, 98) - Texto: “Sí” - Clase:	Es texto	Correcto
2	- Coordenadas: (90, 120, 100, 120) - Texto: “No” - Clase:	Es texto	Correcto
3	- Coordenadas: (34, 90, 56, 98) - Texto: “Inicio” - Clase: start_end	Es figura	Correcto
4	- Coordenadas: (34, 90, 56, 98) - Texto: - Clase: arrow_line_right	Es conector	Correcto
5	- Coordenadas: (34, 90, 56, 98) - Texto: - Clase:	None	Correcto

Graph

EPU 011	**Módulo:** Graph	**Autor:** OFCG
Fecha: 29/01/2020	**Requerimiento(s):** RF7, RF8 y RF9	**ID ambiente:** AP2
Objetivo: Verificar el correcto funcionamiento de la construcción del grafo.		

VS1	**Tester:** OFCG	**Fecha:** 06/04/2020	
Escenario: 1. Instanciar un objeto de la clase "Graph". 2. Llamar a la función "generate_graph(...)" pasando como parámetro 2 listas de objetos tipo "Node".			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Nodos de texto: Node(coordinate = [384,600,123,189],text = "inicio") t2 = Node(coordinate = [375,597,378,438],text = "x=6") t3 = Node(coordinate = [429,570,678,750],text = "x>5") t4 = Node(coordinate = [402,579,960,1008],text = "verdad") t5 = Node(coordinate = [783,930,948,1008],text = "falso") t6 = Node(coordinate = [426,546,1341,1410],text = "fin") t7 = Node(coordinate = [705,825,606,666],text = "si") t8 = Node(coordinate = [363,423,858,906],text = "no") - Nodos de figura: s1 = Node(coordinate = [318,675,81,231],class_shape = "start_end") s2 = Node(coordinate = [456,513,237,354],class_shape = "arrow_line_down") s3 = Node(coordinate = [306,684,354,489],class_shape = "process") s4 = Node(coordinate = [462,522,483,594],class_shape = "arrow_line_down") s5 = Node(coordinate = [345,615,588,858],class_shape = "decision") s6 = Node(coordinate = [612,921,702,939],class_shape = "arrow_rectangle_down",image_path="graph_images/rect_down.png") s7 = Node(coordinate = [438,504,849,954],class_shape = "arrow_line_down") s8 = Node(coordinate = [372,645,948,1125],class_shape = "print") s9 = Node(coordinate = [741,1059,936,1095],class_shape = "print") s10 = Node(coordinate = [420,471,1119,1320],class_shape = "arrow_line_down") s11 = Node(coordinate = [669,849,1092,1410],class_shape = "arrow_rectangle_right",image_path="graph_images/rect_right.png")	adjacency_list = {0: [1], 1: [2], 2: [3], 3: [4], 4: [6, 5], 5: [8], 6: [7], 7: [9], 8: [10], 9: [11], 10: [11], 11: []}	Correcto

	s12 = Node(coordinate = [339,669,1317,1455],class_shape = "start_end")		
2	- Nodos de texto: t1 = Node(coordinate = [339,435,70,102],text = "inicio") t2 = Node(coordinate = [336,463,199,232],text = "x=0") t3 = Node(coordinate = [370,404,339,369],text = "x") t4 = Node(coordinate = [350,477,494,536],text = "x=x/2") t5 = Node(coordinate = [367,477,650,692],text = "x+5") t6 = Node(coordinate = [374,452,912,959],text = "fin") - Nodos de figura: s1 = Node(coordinate = [302,484,45,124],class_shape =self.visited_list "start_end") s2 = Node(coordinate = [380,412,127,182],class_shape = "arrow_line_down") s3 = Node(coordinate = [282,497,177,263],class_shape = "process") s4 = Node(coordinate = [376,412,249,302],class_shape = "arrow_line_down") s5 = Node(coordinate = [286,502,303,390],class_shape = "print") s6 = Node(coordinate = [378,421,392,482],class_shape = "arrow_line_down") s7 = Node(coordinate = [292,537,479,580],class_shape = "process") s8 = Node(coordinate = [404,427,576,634],class_shape = "arrow_line_down") s9 = Node(coordinate = [319,517,635,751],class_shape = "print") s10 = Node(coordinate = [403,447,708,881],class_shape = "arrow_line_down") s11 = Node(coordinate = [312,537,881,984],class_shape = "start_end")	adjacency_list = {0: [1], 1: [2], 2: [3], 3: [4], 4: [5], 5: [6], 6: [7], 7: [8], 8: [9], 9: [10], 10: []}	Correcto
3	- Nodos de texto: t1 = Node(coordinate = [504,1344,334,436],text = "ans=0,n=0,a=0,b=1,cont=2") t2 = Node(coordinate = [1068,1318,96,212],text = "inicio") t3 = Node(coordinate = [772,846,642,694],text = "n") t4 = Node(coordinate = [170,312,630,716],text = "fin") t5 = Node(coordinate = [660,896,1006,1090],text = "cont<n") t6 = Node(coordinate = [450,554,964,1036],text = "no") t7 = Node(coordinate = [132,270,1028,1086],text = "ans") t8 = Node(coordinate = [828,888,1250,1326],text = "si") t9 = Node(coordinate = [696,974,1452,1528],text = "ans=a+b") t10 = Node(coordinate = [734,878,1760,1824],text = "a=b") t11 = Node(coordinate = [728,948,1992,2058],text = "b=ans") t12 = Node(coordinate = [664,1106,2232,2302],text = "cont=cont+1") - Nodos de figura: s1 = Node(coordinate = [950,1413,80,148],class_shape = "start_end") s2 = Node(coordinate = [717,957,113,309],class_shape	adjacency_list = {0: [1], 1: [2], 2: [3], 3: [4], 4: [6], 5: [], 6: [10], 7: [5], 8: [7], 9: [8], 10: [12, 9], 11: [10], 12: [13], 13: [14], 14: [15], 15: [16], 16: [17], 17: [18], 18: [19], 19: [20], 20: [11]}	Correcto

	= "arrow_rectangle_down",image_path="graph_images/set9/rectdown.png") s3 = Node(coordinate = [468,1361,303,453],class_shape = "process") s4 = Node(coordinate = [798,867,441,611],class_shape = "arrow_line_down") s5 = Node(coordinate = [572,1020,609,726],class_shape = "scan") s6 = Node(coordinate = [74,434,614,742],class_shape = "start_end") s7 = Node(coordinate = [780,852,738,926],class_shape = "arrow_line_down") s8 = Node(coordinate = [176,262,786,1006],class_shape = "arrow_line_up") s9 = Node(coordinate = [80,328,998,1172],class_shape = "print") s10 = Node(coordinate = [334,630,1022,1082],class_shape = "arrow_line_left") s11 = Node(coordinate = [626,964,914,1230],class_shape = "decision") s12 = Node(coordinate = [962,1324,1008,1742],class_shape = "arrow_rectangle_left",image_path="graph_images/set9/rectleft.png") s13 = Node(coordinate = [780,830,1224,1434],class_shape = "arrow_line_down") s14 = Node(coordinate = [646,1036,1438,1562],class_shape = "process") s15 = Node(coordinate = [788,838,1570,1726],class_shape = "arrow_line_down") s16 = Node(coordinate = [646,986,1728,1852],class_shape = "process") s17 = Node(coordinate = [784,836,1852,1966],class_shape = "arrow_line_down") s18 = Node(coordinate = [668,1006,1966,2088],class_shape = "process") s19 = Node(coordinate = [780,836,2086,2216],class_shape = "arrow_line_down") s20 = Node(coordinate = [602,1140,2222,2328],class_shape = "process") s21 = Node(coordinate = [1154,1330,1740,2252],class_shape = "arrow_rectangle_up",image_path="graph_images/set9/rectup.png")		

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020	
Escenario: 1. Instanciar un objeto de la clase "Graph". 2. Llamar a la función "generate_graph(...)" pasando como parámetros 2 listas de objetos tipo "Node".			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Nodos de texto: [Node(class:None,text:inicio), Node(class:None,text:n=0), Node(class:None,text:n), Node(class:None,text:Fin), Node(class:None,text:no), Node(class:None,text:"F"), Node(class:None,text:n == 0),	adjacency_list = {0: [6], 1: [14], 2: [7, 13], 3: [], 4: [10], 5: [], 6: [3], 7: [0], 8: [12], 9: [2], 10:	Correcto

	Node(class:None,text:si), Node(class:None,text:"n"), Node(class:None,text:Fin)] - Nodos de figura: [Node(class:print,text:None), Node(class:print,text:None), Node(class:decision,text:None), Node(class:start_end,text:None), Node(class:start_end,text:None), Node(class:start_end,text:None), Node(class:arrow_line_down,text:None), Node(class:arrow_line_down,text:None), Node(class:arrow_line_down,text:None), Node(class:arrow_line_down,text:None), Node(class:arrow_line_down,text:None), Node(class:process,text:None), Node(class:scan,text:None), Node(class:arrow_line_right,text:None), Node(class:arrow_line_right,text:None)]	[11], 11: [8], 12: [9], 13: [1], 14: [5]}	
2	- Nodos de texto: [Node(class:None,text:Inicio), Node(class:None,text:cont = 2), Node(class:None,text:No), Node(class:None,text:Fin), Node(class:None,text:cont <= 100), Node(class:None,text:si), Node(class:None,text:cont), Node(class:None,text:cont = cont + 2)] - Nodos de figura: [Node(class:process,text:None), Node(class:process,text:None), Node(class:start_end,text:None), Node(class:start_end,text:None), Node(class:arrow_line_down,text:None), Node(class:arrow_line_down,text:None), Node(class:arrow_line_down,text:None), Node(class:arrow_line_down,text:None), Node(class:decision,text:None), Node(class:arrow_line_right,text:None), Node(class:arrow_line_right,text:None), Node(class:arrow_line_up,text:None), Node(class:arrow_line_left,text:None), Node(class:print,text:None)]	adj_list = {0: [12], 1: [5], 2: [6], 3: [], 4: [0], 5: [8], 6: [1], 7: [13], 8: [7, 9], 9: [3], 10: [8], 11: [10], 12: [11], 13: [4]}	Correcto
3	- Nodos de texto: [Node(class:None,text:Inicio), Node(class:None,text:"Hola mundo"), Node(class:None,text:Fin)] - Nodos de figura: [Node(class:start_end,text:None), Node(class:start_end,text:None), Node(class:arrow_line_down,text:None), Node(class:arrow_line_down,text:None), Node(class:print,text:None)]	adj_list {0: [3], 1: [], 2: [1], 3: [4], 4: [2]}	Correcto

Flowchart Generator

EPU 013	**Módulo:** FlowchartGenerator	**Autor:** OFCG
Fecha: 29/01/2020	**Requerimiento(s):** RF8	**ID ambiente:** AP2
Objetivo: Comprobar el buen funcionamiento de la función generate_flowchart.		

VS1	**Tester:** DBM	**Fecha:** 18/04/2020 - 19/04/2020	
Escenario: 3. Instanciar un objeto de la clase "FlowchartGenerator", se le pasa un grafo dirigido válido y la ruta con nombre para guardar el resultado, 4. Llamar a la función "generate_flowchart()"..			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Grafo dirigido validado: “Hola mundo”.	Imagen digital reconstruida del diagrama de flujo.	Correcto
2	- Grafo dirigido validado: “Sumar dos números”.	Imagen digital reconstruida del diagrama de flujo.	Correcto
3	- Grafo dirigido validado: “Factorial”.	Imagen digital reconstruida del diagrama de flujo.	En la decisión, al poner un acento en " Sí" para el label de la flecha lo corrompe, sin acento funciona correctamente.
4	- Grafo dirigido validado: “Imprimir números pares hasta el 100”	Imagen digital reconstruida del diagrama de flujo.	Correcto
5	- Grafo dirigido validado: “Imprimir números impares hasta el 100”	Imagen digital reconstruida del diagrama de flujo.	Correcto
6	- Grafo dirigido validado: “Calcular número de dígitos de un entero”	Imagen digital reconstruida del diagrama de flujo.	Correcto
7	- Grafo dirigido validado: “Calcular una raíz cuadrada”	Imagen digital reconstruida del diagrama de flujo.	Correcto
8	- Grafo dirigido validado: “Imprimir una palabra al revés”	Imagen digital reconstruida del diagrama de flujo.	Correcto
9	- Grafo dirigido validado: “Calcular el área de un cuadrado”.	Imagen digital reconstruida del diagrama de flujo.	Correcto

Code Generator

EPU 014	**Módulo:** CodeGenerator	**Autor:** DBM
Fecha: 29/01/2020	**Requerimiento(s):** RF9	**ID ambiente:** AP1
Objetivo: Verificar que se genera el código fuente en C.		

VS1	**Tester:** OFCG	**Fecha:** 06/04/2020	
Escenario: 1. Instanciar un objeto de la clase "CodeGenerator" pasándole como parámetro un grafo dirigido correcto. 2. Llamar el método "generate()". 3. Comprobar que el archivo con extensión .c se ha generado, y tiene código en lenguaje de programación C.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**

1	- Grafo dirigido validado: {0: [1], 1: [2], 2: [3], 3: [4], 4: [6, 5], 5: [8], 6: [7], 7: [9], 8: [10], 9: [11], 10: [11], 11: []}.	#include<stdio.h>
		int main(){
		int x=6;
		if(x>5){
		printf(""verdad"");
		}else{
		printf(""falso"");
		}
		return 0;
		}
			Incorrecto, IndexError: list index out of range

2	- Grafo dirigido validado: {0: [1], 1: [2], 2: [3], 3: [4], 4: [6], 5: [], 6: [10], 7: [5], 8: [7], 9: [8], 10: [12, 9], 11: [10], 12: [13], 13: [14], 14: [15], 15: [16], 16: [17], 17: [18], 18: [19], 19: [20], 20: [11]}.	#include<stdio.h>
		int main(){
		int ans=0,n=0,a=0,b=1,cont=2;
		scanf("%d",&n);
		while(cont<n){
		ans=a+b;
		a=b;
		b=ans;
		cont=cont+1;
		}
		printf("%d",ans);
		return 0;
		}
			Incorrecto, IndexError: list index out of range

3	- nodos: [Node(class:start_end,text:inicio),	#include<stdio.h>	Incorrecto,

	Node(class:arrow_line_down,text:None),
	Node(class:process,text:x=x+1),
	Node(class:arrow_line_down,text:None),
	Node(class:print,text:x),
	Node(class:arrow_line_down,text:None),
	Node(class:process,text:x=x/2),
	Node(class:arrow_line_down,text:None),
	Node(class:print,text:x+5),
	Node(class:arrow_line_down,text:None),
	Node(class:start_end,text:fin)]
	- Grafo dirigido validado: {0: [1], 1: [2], 2: [3], 3: [4], 4: [5], 5: [6], 6: [7], 7: [8], 8: [9], 9: [10], 10: []}
		int main(){
		int x=0;
		printf("%d",x);
		x=x/2;
		printf("%d",x+5);
		return 0;
		}
			IndexError: list index out of range

VS1	**Tester:** OFCG	**Fecha:** 06/04/2020	
Escenario: 1. Instanciar un objeto de la clase "CodeGenerator" pasándole como parámetro un grafo dirigido correcto. 2. Llamar el método "generate()". 3. Comprobar que el archivo con extensión .c se ha generado, y tiene código en lenguaje de programación C.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**

1	- Grafo dirigido validado: {0: [1], 1: [2], 2: [3], 3: [4], 4: [6, 5], 5: [8], 6: [7], 7: [9], 8: [10], 9: [11], 10: [11], 11: []}.	#include<stdio.h>
		int main(){
		int x=6;
		if(x>5){
		printf(""verdad"");
		}else{
		printf(""falso"");
		}
		return 0;
		}
			#include<stdio.h>
			int main(){
			int x=6;
			if(x>5){
			printf("verdad");
			}
			printf("falso");
			return 0;
			}

2	- Grafo dirigido validado: {0: [1], 1: [2], 2: [3], 3: [4], 4: [6], 5: [], 6: [10], 7: [5], 8: [7], 9: [8], 10: [12, 9], 11: [10], 12: [13], 13: [14], 14: [15], 15: [16], 16: [17], 17: [18], 18: [19], 19: [20], 20: [11]}.	#include<stdio.h>
		int main(){
		int
		ans=0,n=0,a=0,b=1,cont=2;
		scanf("%d",&n);
		while(cont<n){
		ans=a+b;
		a=b;
		b=ans;
		cont=cont+1;
		}
		printf("%d",ans);
		return 0;
		}
			Incorrecto, IndexError: list index out of range

3	- nodos: [Node(class:start_end,text:inicio),
	Node(class:arrow_line_down,text:None),
	Node(class:process,text:x=x+1),
	Node(class:arrow_line_down,text:None),
	Node(class:print,text:x),
	Node(class:arrow_line_down,text:None),
	Node(class:process,text:x=x/2),
	Node(class:arrow_line_down,text:None),
	Node(class:print,text:x+5),
	Node(class:arrow_line_down,text:None),
	Node(class:start_end,text:fin)]
	- Grafo dirigido validado: {0: [1], 1: [2], 2: [3], 3: [4], 4: [5], 5: [6], 6: [7], 7: [8], 8: [9], 9: [10], 10: []}
		#include<stdio.h>
		int main(){
		int x=0;
		printf("%d",x);
		x=x/2;
		printf("%d",x+5);
		return 0;
		}
			#include<stdio.
			h>
			int main(){
			int x=0;
			
			printf("%d",x);
			x=x/2;
			
			printf("%d",x);
			return 0;
			}

VS1	**Tester:** OFCG	**Fecha:** 07/04/2020	
Escenario: 1. Instanciar un objeto de la clase "CodeGenerator" pasándole como parámetro un grafo dirigido correcto. 2. Llamar el método "generate()". 3. Comprobar que el archivo con extensión .c se ha generado, y tiene código en lenguaje de programación C.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**

1	- Grafo dirigido validado: {0: [1], 1: [2], 2: [3], 3: [4], 4: [6, 5], 5: [8], 6: [7], 7: [9], 8: [10], 9: [11], 10: [11], 11: []}.	#include<stdio.h>
		int main(){
		int x=6;
		if(x>5){
		printf(""verdad"");
		}else{
		printf(""falso"");
		}
		return 0;
		}
			#include<stdio.
			h>
			int main(){
			int x=6;
			if(x>5){
			
			printf("%s","ver
			dad");
			}
			else{
			
			printf("%s","fal
			so");
			}
			return 0;
			}

2	- Grafo dirigido validado: {0: [1], 1: [2], 2: [3], 3: [4], 4: [6], 5: [], 6: [10], 7: [5], 8: [7], 9: [8], 10: [12, 9], 11: [10], 12: [13], 13: [14], 14: [15], 15: [16], 16: [17], 17: [18], 18: [19], 19: [20], 20: [11]}.	#include<stdio.h>
		int main(){
		int
		ans=0,n=0,a=0,b=1,cont=2;
		scanf("%d",&n);
		while(cont<n){
		ans=a+b;
		a=b;
		b=ans;
		cont=cont+1;
		}
			#include<stdio.
			h>
			int main(){
			int
			ans=0,n=0,a=0,
			b=1,cont=2;
			
			scanf("%d",&n)
			;
			
			while(cont<n){

		printf("%d",ans);
		return 0;
		}
			ans=a+b;
			a=b;
			b=ans;
			
			cont=cont+1;
			}
			
			printf("%d",ans)
			;
			return 0;
			}

3	- nodos: [Node(class:start_end,text:inicio),
	Node(class:arrow_line_down,text:None),
	Node(class:process,text:x=x+1),
	Node(class:arrow_line_down,text:None),
	Node(class:print,text:x),
	Node(class:arrow_line_down,text:None),
	Node(class:process,text:x=x/2),
	Node(class:arrow_line_down,text:None),
	Node(class:print,text:x+5),
	Node(class:arrow_line_down,text:None),
	Node(class:start_end,text:fin)]
	- Grafo dirigido validado: {0: [1], 1: [2], 2: [3], 3: [4], 4: [5], 5: [6], 6: [7], 7: [8], 8: [9], 9: [10], 10: []}
		#include<stdio.h>
		int main(){
		int x=0;
		printf("%d",x);
		x=x/2;
		printf("%d",x+5);
		return 0;
		}
			#include<stdio.
			h>
			int main(){
			int x=0;
			
			printf("%d",x);
			x=x/2;
			
			printf("%d",x);
			return 0;
			}

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020	
Escenario: 1. Instanciar un objeto de la clase "CodeGenerator" pasándole como parámetro un grafo dirigido correcto. 2. Llamar el método "generate()". 3. Comprobar que el archivo con extensión .c se ha generado, y tiene código en lenguaje de programación C.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**

1	- nodos: [Node(class:process,text:cont = cont + 2),
	Node(class:process,text:cont = 2),
	Node(class:start_end,text:Inicio),
	Node(class:start_end,text:Fin),
	Node(class:arrow_line_down,text:None),
	Node(class:arrow_line_down,text:None),
	Node(class:arrow_line_down,text:None),
	Node(class:arrow_line_down,text:si),
	Node(class:decision,text:cont <= 100),
	Node(class:arrow_line_right,text:No),
	Node(class:arrow_line_right,text:None),
	Node(class:arrow_line_up,text:None),
	Node(class:arrow_line_left,text:None),
	Node(class:print,text:cont)]
	- Grafo dirigido validado: {0: [12], 1: [5], 2: [6], 3: [], 4: [0], 5: [8], 6: [1], 7: [13], 8: [7, 9], 9: [3], 10: [8], 11: [10], 12: [11], 13: [4]}
		#include<stdio.h>
		int main(){
		int n=o;
		scanf("%d",&n);
		if(n == 0){
		printf("%s","n");
		}
		else{
		printf("%s","F");
		}
		return 0;
		}
			#include<stdio.
			h>
			int main(){
			int n=o;
			
			scanf("%d",&n)
			;
			if(n == 0){
			
			printf("%s","n")
			;
			}
			else{
			
			printf("%s","F")
			;
			}
			return 0;
			}

2	- nodos: [Node(class:process,text:cont = cont + 2),
	Node(class:process,text:cont = 2),
	Node(class:start_end,text:Inicio),
	Node(class:start_end,text:Fin),
	Node(class:arrow_line_down,text:None),
	Node(class:arrow_line_down,text:None),
	Node(class:arrow_line_down,text:None),
	Node(class:arrow_line_down,text:si),
	Node(class:decision,text:cont <= 100),
	Node(class:arrow_line_right,text:No),
	Node(class:arrow_line_right,text:None),
	Node(class:arrow_line_up,text:None),
	Node(class:arrow_line_left,text:None),
	Node(class:print,text:cont)]
	- Grafo dirigido validado: {0: [12], 1: [5], 2: [6], 3: [], 4: [0], 5: [8], 6: [1], 7: [13], 8: [7, 9], 9: [3], 10: [8], 11: [10], 12: [11], 13: [4]}
		#include<stdio.h>
		int main(){
		int cont=2;
		while(cont <= 100){
		printf("%d",cont);
		cont=cont+2;
		}
		return 0;
		}
			#include<stdio.h>
			int main(){
			int cont=2;
			while(cont <= 100){
			
			printf("%d",cont);
			
			cont=cont+2;
			}
			return 0;
			}

3	- nodos: [Node(class:start_end,text:Inicio),
	Node(class:start_end,text:Fin),
	Node(class:arrow_line_down,text:None),
	Node(class:arrow_line_down,text:None),
	Node(class:print,text:"Hola mundo")]
	[Node(class:start_end,text:Inicio),
	Node(class:start_end,text:Fin),
	Node(class:arrow_line_down,text:None),
	Node(class:arrow_line_down,text:None),
	Node(class:print,text:"Hola mundo")]
	- Grafo dirigido validado: {0: [3], 1: [], 2: [1], 3: [4], 4: [2]}
		#include<stdio.h>
		int main(){
		printf("%s","Hola mundo");
		return 0;
		}
			#include<stdio.h>
			int main(){
			
			printf("%s","Hola mundo");
			return 0;
			}

EPU 015	**Módulo:** CodeGenerator	**Autor:** DBM
Fecha: 29/01/2020	**Requerimiento(s):** RF9	**ID ambiente:** AP1
Objetivo: Verificar que se compila un archivo .c.		

VS1	**Tester:** OFCG	**Fecha:** 17/04/2020	
Escenario: 1. Ya una vez que se ha generado el archivo con el código en C. 2. Llamar el método "compile()".			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Archivo: hello_world.c	Warnings, errores o nada.	Correcto
2	- Archivo: add_two_numbers.c	Warnings, errores o nada.	Correcto
3	- Archivo: factorial.c	Warnings, errores o nada.	Correcto
4	- Archivo: even_to_100.c	Warnings, errores o nada.	Correcto

5	- Archivo: odd_to_100.c	Warnings, errores o nada.	Correcto
6	- Archivo: get_number_digits.c	Warnings, errores o nada.	Correcto
7	- Archivo: elevate.c	Warnings, errores o nada.	Correcto
8	- Archivo: sqrt.c	Warnings, errores o nada.	Correcto
9	- Archivo: reversed_word.c	Warnings, errores o nada.	Correcto
10	- Archivo: reversed_word.c	Warnings, errores o nada.	Correcto

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020	
Escenario: 1. Ya una vez que se ha generado el archivo con el código en C. 2. Llamar el método "compile()".			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Archivo: hello_world.c	Warnings, errores o nada.	Correcto
2	- Archivo: add_two_numbers.c	Warnings, errores o nada.	Correcto
3	- Archivo: factorial.c	Warnings, errores o nada.	Correcto
4	- Archivo: even_to_100.c	Warnings, errores o nada.	Correcto
5	- Archivo: odd_to_100.c	Warnings, errores o nada.	Correcto
6	- Archivo: get_number_digits.c	Warnings, errores o nada.	Correcto
7	- Archivo: elevate.c	Warnings, errores o nada.	Correcto
8	- Archivo: sqrt.c	Warnings, errores o nada.	Correcto
9	- Archivo: reversed_word.c	Warnings, errores o nada.	Correcto
10	- Archivo: reversed_word.c	Warnings, errores o nada.	Correcto

Handler

EPU 016	**Módulo:** Handler	**Autor:** DBM
Fecha: 29/01/2020	**Requerimiento(s):** RF11	**ID ambiente:** AP2.
Objetivo: Comprobar el buen funcionamiento de la ventana principal.		

VS1	**Tester:** DBM	**Fecha:** 21/04/2020	
Escenario: 1. Ejecutar el script de la GUI , "handler.py". 2. Oprimir el botón de entrenar. 3. Cerrar ventana de entrenar 4. Oprimir el botón de predecir.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Oprimir botón "Entrenar".	Se abre la ventana train_model	Correcto
2	- Oprimir botón "Reconocer diagrama de flujo".	Se abre la ventana recognize	Correcto

VS2	**Tester:** OFCG	**Fecha:** 25/07/2020	
Escenario: 1. Ejecutar el script de la GUI , "handler.py". 2. Oprimir el botón de entrenar modelo de figuras. 3. Cerrar ventana de entrenar 4. Oprimir el botón de predecir. 5. Cerrar ventana de predecir 6. Oprimir ventana de entrenar modelo de texto.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Oprimir botón "Entrenar modelo de figuras".	Se abre la ventana train_model	Correcto
2	- Oprimir botón "Reconocer diagrama de flujo".	Se abre la ventana recognize	Correcto
3	- Oprimir botón “Entrenar modelo de texto”	Se abre ventana train_text_model	Correcto

EPU 017	**Módulo:** Handler	**Autor:** OFCG
Fecha: 29/01/2020	**Requerimiento(s):** RF11	**ID ambiente:** AP2.
Objetivo: Comprobar el buen funcionamiento de la ventana del entrenamiento del modelo de figuras.		

VS1	**Tester:** DBM	**Fecha:** 22/04/2020

Escenario:

1. Ejecutar el script de la GUI , "handler.py".
2. Oprimir el botón de entrenar.
3. Se abre una ventana para entrenar un modelo.
4. Seleccionar carpeta del dataset
5. Añadir un número (entero) de regiones de interés a considerar.
6. Seleccionar modelo pre-entrenado para cargar los pesos.
7. Añadir un número (entero) para el número de épocas.
8. Añadir un número (real) en el input de learning rate.
9. Check o no en el uso de la GPU.
10. Oprimir botón de iniciar.

Caso	Entrada	Resultado esperado	Resultado
1	- dataset_path: “/home/david/Escritorio/flowchart-3b(splitter)” - num_rois: 32 - pre-trained_model_path: “” - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘False’	Muestra mensaje "Train"	Correcto
2	- dataset_path: “/home/david/Escritorio/flowchart-3b(splitter)” - num_rois: 42 - pre-trained_model_path: “/home/david/Escritorio/handwritten-flowchart-with-cnn/model/vgg16_weights_tf_dim_ordering_tf_kernels.h5” - num_epochs: 100 - learning_rate: 0.001 - use_gpu: ‘True’	Muestra mensaje "Train"	Correcto
3	- dataset_path: “/home/david/Escritorio/flowchart-3b(splitter)” - num_rois: -1 - pre-trained_model_path: “” - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘False’	Muestra mensaje: Num rois not valid	Correcto
4	- dataset_path: “/home/david/Escritorio/flowchart-3b(splitter)” - num_rois: 32 - pre-trained_model_path: “” - num_epochs: xd - learning_rate: 0.00001 - use_gpu: ‘False’	Mensaje de error: Num epochs must be a integer	Correcto
5	- dataset_path: “/home/david/Escritorio/flowchart-3b(splitter)” - num_rois: 32 - pre-trained_model_path: “”	Mensaje de error: Learning rate must be a real number	Correcto

	- num_epochs: xd - learning_rate: 45xsd - use_gpu: ‘False’		
6	- dataset_path: “” - num_rois: 32 - pre-trained_model_path: “” - num_epochs: xd - learning_rate: 0.00001 - use_gpu: ‘False’	Mensaje de error: Dataset path not valid	Correcto

EPU 018	**Módulo:** Handler	**Autor:** OFCG
Fecha: 29/01/2020	**Requerimiento(s):** RF11	**ID ambiente:** AP2.
Objetivo: Comprobar el buen funcionamiento de la ventana de reconocimiento.		

VS1	**Tester:** DBM	**Fecha:** 23/04/2020	
Escenario: 1. Ejecutar el script de la GUI , "handler.py". 2. Oprimir el botón de predecir. 3. Se abre una ventana para predecir modelo. 4. Presionar el combobox y seleccionar la carpeta donde se encuentra el modelo entrenado. 5. Presionar botón de seleccionar imagen. 6. Seleccionar imagen. 7. Hacer check o no en el uso de la GPU 8. Modificar el número de regiones propuestas 9. Presionar botón de predecir.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Carpeta de entrenamiento: 3/ - Seleccionar imagen: hello_world.jpg - RoIs: - Use GPU: ‘True’	Se abre la ventana de resultados	Correcto
2	- Carpeta de entrenamiento: 8/ - Seleccionar imagen: Fibo.jpg - RoIs: 32 - Use GPU: ‘False’	Se abre la ventana recognize	Correcto
3	- Carpeta de entrenamiento: best_results_miniset/ - Seleccionar imagen: hello_world.jpg - RoIs: - Use GPU: ‘False’	Mensaje de error: Selected folder doesn't contains any model	Correcto
4	- Carpeta de entrenamiento: 8/ - Seleccionar imagen:	Mensaje de error: Image not selected	Correcto

	- RoIs: - Use GPU: ‘False’		
5	- Carpeta de entrenamiento: 8/ - Seleccionar imagen: hello_world.jpg - RoIs: -3 - Use GPU: ‘False’	Mensaje de error: Image not selected	Correcto

VS2	**Tester:** DBM	**Fecha:** 25/06/2020	
Escenario: 10. Ejecutar el script de la GUI , "handler.py". 11. Oprimir el botón de predecir. 12. Se abre una ventana para predecir modelo. 13. Presionar el combobox y seleccionar la carpeta donde se encuentra el modelo entrenado. 14. Presionar botón de seleccionar imagen. 15. Seleccionar imagen. 16. Hacer check o no en el uso de la GPU 17. Modificar el número de regiones propuestas 18. Presionar botón de predecir.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Carpeta de entrenamiento: 3/ - Seleccionar imagen: hello_world.jpg - RoIs: - Use GPU: ‘True’	Se abre la ventana de resultados	Correcto
2	- Carpeta de entrenamiento: 8/ - Seleccionar imagen: Fibo.jpg - RoIs: 32 - Use GPU: ‘False’	Se abre la ventana recognize	Correcto
3	- Carpeta de entrenamiento: best_results_miniset/ - Seleccionar imagen: hello_world.jpg - RoIs: - Use GPU: ‘False’	Mensaje de error: Selected folder doesn't contains any model	Correcto
4	- Carpeta de entrenamiento: 8/ - Seleccionar imagen: - RoIs: - Use GPU: ‘False’	Mensaje de error: Image not selected	Correcto
5	- Carpeta de entrenamiento: 8/ - Seleccionar imagen: hello_world.jpg - RoIs: -3 - Use GPU: ‘False’	Mensaje de error: Image not selected	Correcto

EPU 019	**Módulo:** Handler	**Autor:** OFCG

Fecha: 25/06/2020	**Requerimiento(s):** RF11	**ID ambiente:** AP1.
Objetivo: Comprobar el buen funcionamiento de la ventana de edit_text.		

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020	
Escenario: 1. Ejecutar el script de la GUI , "handler.py". 2. Oprimir el botón de predecir. 3. Se abre una ventana para predecir modelo. 4. Presionar el combobox y seleccionar la carpeta donde se encuentra el modelo entrenado. 5. Presionar botón de seleccionar imagen. 6. Seleccionar imagen. 7. Hacer check o no en el uso de la GPU 8. Modificar el número de regiones propuestas 9. Presionar botón de predecir. 10. Se abre la ventana de editar texto 11. Se cambian los textos de las imágenes si es necesario			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Corregir texto. - Acción: Corregir texto. - Acción: Corregir texto.	Se abre la ventana de elección entre entrenar modelo ahora o más tarde	Correcto
2	- Acción: Corregir texto. - Acción: - Acción: Corregir texto.	Se abre la ventana de elección entre entrenar modelo ahora o más tarde	Correcto

EPU 020	**Módulo:** Handler	**Autor:** OFCG
Fecha: 25/06/2020	**Requerimiento(s):** RF11	**ID ambiente:** AP1.
Objetivo: Comprobar el buen funcionamiento de la ventana de train_text.		

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020
Escenario: 1. Ejecutar el script de la GUI , "handler.py". 2. Oprimir el botón de predecir. 3. Se abre una ventana para predecir modelo. 4. Presionar el combobox y seleccionar la carpeta donde se encuentra el modelo entrenado. 5. Presionar botón de seleccionar imagen. 6. Seleccionar imagen. 7. Hacer check o no en el uso de la GPU 8. Modificar el número de regiones propuestas		

9. Presionar botón de predecir. 10. Se abre la ventana de editar texto 11. Se cambian los textos de las imágenes si es necesario 12. Se oprime el botón de continuar. 13. Se abre la ventana de decidir entrenar ahora o luego.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Entrenar ahora.	Se empieza el proceso entrenamiento, después se muestra los resultados.	Correcto
2	- Acción: Entrenar después.	Se abre la ventana de resultados	Correcto

K.2 Pruebas de integración

Handler y Shape Classifier

EPI 003	**Autor:** OFCG

Fecha: 29/01/2020	**Requerimiento(s):** RF11, RF1, RF2, RF3 y RF5.	**ID ambiente:** AP2.
Objetivo: Comprobar la integridad del módulo Handler con el ShapeClassifier.		

VS1	**Tester:** DBM	**Fecha:** 21/04/2020	
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona una carpeta de los resultados de entrenamiento. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona o no la opción de usar GPU 10. Se ingresa el valor o no para regiones de interés (RoIs). 11. Se oprime el botón "Predict". 12. En la terminal se muestra la lista de nodos reconocidos en el diagrama de flujo.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de máquina. - Usar GPU: No - RoIs: 32 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto
2	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto, aunque faltaron dos nodos de reconocer
3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5	Una lista de objetos de tipo "Node".	Correcto, aunque faltaron dos nodos de

	- Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".		reconocer
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de máquina. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto, aunque faltaron 11 nodos de reconocer, y hay dos nodos mal clasificados.
5	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto, aunque: 9 no reconocidos, 4 mal clasificados.
6	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto, aunque: 14 no reconocidos, 1 mal clasificado.
7	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de máquina. - Usar GPU: No	Una lista de objetos de tipo "Node".	Correcto, aunque: 13 no reconocidos, 3 mal clasificados.

	- RoIs: 42 - Acción: Oprimir botón "Predict".		
8	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de cuadrícula.. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto, aunque: 15 no reconocidos, 4 mal clasificados.
9	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto, aunque: 19 no reconocidos.

Handler, Shape Classifier y Text Classifier

EPI 004	**Autor:** OFCG	
Fecha: 29/01/2020	**Requerimiento(s):** RF11, RF1, RF2, RF3, RF5 y RF6.	**ID ambiente:** AP2.
Objetivo: Comprobar la integridad del módulo Handler con el ShapeClassifier y TextClassifier.		

VS1	**Tester:** DBM	**Fecha:** 10/05/2020
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona una carpeta de los resultados de entrenamiento. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona o no la opción de usar GPU 10. Se ingresa el valor o no para regiones de interés (RoIs). 11. Se oprime el botón "Predict".		

12. En la terminal se muestran dos listas de nodos reconocidos en el diagrama de flujo.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de máquina. - Usar GPU: No - RoIs: 32 - Acción: Oprimir botón "Predict".	Dos listas de objetos de tipo "Node".	Correcto, aunque no 100% preciso.
2	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Dos listas de objetos de tipo "Node".	Correcto, aunque no 100% preciso.
3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Dos listas de objetos de tipo "Node".	Correcto, aunque no 100% preciso.
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de máquina. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Dos listas de objetos de tipo "Node".	Correcto, aunque no 100% preciso.
5	- Acción: Oprimir botón	Dos listas de objetos de	Correcto,

	"Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	tipo "Node".	aunque no 100% preciso.
6	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Dos listas de objetos de tipo "Node".	Correcto, aunque no 100% preciso.
7	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de máquina. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Dos listas de objetos de tipo "Node".	Correcto, aunque no 100% preciso.
8	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de cuadrícula.. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Dos listas de objetos de tipo "Node".	Correcto, aunque no 100% preciso..
9	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir".	Dos listas de objetos de tipo "Node".	Correcto, aunque no 100% preciso.

	- Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".		

Handler, Shape Classifier, Text Classifier y Graph

EPI 005	**Autor:** OFCG	
Fecha: 29/01/2020	**Requerimiento(s):** RF11, RF1, RF2, RF3, RF5, RF6 y RF7.	**ID ambiente:** AP2.
Objetivo: Comprobar la integridad del módulo Handler con el ShapeClassifier, TextClassifier y Graph.		

VS1	**Tester:** DBM	**Fecha:** 10/05/2020	
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona una carpeta de los resultados de entrenamiento. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona o no la opción de usar GPU 10. Se ingresa el valor o no para regiones de interés (RoIs). 11. Se oprime el botón "Predict". 12. En la terminal se muestran dos listas de nodos reconocidos en el diagrama de flujo.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de máquina. - Usar GPU: No - RoIs: 32 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph".	Aparece un mensaje diciendo que el inicio del grafo no es válido
2	- Acción: Oprimir botón "Recognize flowchart".	Lista de adyacencia de un objeto tipo "Graph".	Aparece un mensaje

	- Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".		diciendo que el inicio del grafo no es válido.
3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph".	Aparece un mensaje diciendo que el inicio del grafo no es válido
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de máquina. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph".	Correcto, aunque no 100% preciso.
5	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Aparece un mensaje diciendo que el inicio del grafo no es válido	Correcto, aunque no 100% preciso.
6	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de	Lista de adyacencia de un objeto tipo "Graph".	Aparece error en el generador de diagramas del tipo, TypeError, con el mensaje: list

	raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".		indices must be integers or slices, not NoneType.
7	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de máquina. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph".	Aparece error en el generador de diagramas del tipo, TypeError, con el mensaje: list indices must be integers or slices, not NoneType
8	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de cuadrícula.. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph".	Aparece error en el generador de diagramas del tipo, TypeError, con el mensaje: list indices must be integers or slices, not NoneType
9	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph".	Aparece error en el generador de diagramas del tipo, TypeError, con el mensaje: list indices must be integers or slices, not NoneType

Handler, Shape Classifier, Text Classifier, Graph y Gram Analyzer.

EPI 006	**Autor:** OFCG	
Fecha: 29/01/2020	**Requerimiento(s):** RF11, RF1, RF2, RF3, RF5, RF6 y RF7.	**ID ambiente:** AP2.

Objetivo: Comprobar la integridad del módulo Handler con el ShapeClassifier, TextClassifier, Graph y GramAnalyzer (Graph).

VS1	**Tester:** DBM	**Fecha:** 10/05/2020	
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona una carpeta de los resultados de entrenamiento. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona o no la opción de usar GPU 10. Se ingresa el valor o no para regiones de interés (RoIs). 11. Se oprime el botón "Predict". 12. En la terminal se muestran dos listas de nodos reconocidos en el diagrama de flujo.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, correcto - Usar GPU: No - RoIs: 32 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph" validado.	Incorrecto, aparece el mensaje de que inicio no válido.
2	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, incorrecto. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Mensaje que indique que algo anda mal en el diagrama	Correcto, aunque no muestra el motivo real que hace el diagrama no válido.
3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de suma de 2 números, correcto.	Mensaje que indique que algo anda mal en el diagrama	Aparece un error en la terminal de excepción en la generación del grafo, TypeError: list indices must

	- Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".		be integers or slices, not NoneType
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de suma de 2 números, incorrecto. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Mensaje que indique que algo anda mal en el diagrama	Aparece un error en la terminal de excepción en la generación del grafo, TypeError: list indices must be integers or slices, not NoneType
5	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de factorial, correcto. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph" validado.	Aparece error: Exception in Tkinter callback, error en función locateROI
6	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, incorrecto. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Mensaje que indique que algo anda mal en el diagrama	Aparece un error en la terminal de excepción en la generación del grafo, TypeError: list indices must be integers or slices, not NoneType.

Handler, Shape Classifier, Text Classifier, Graph, Gram Analyzer y Code Generator

EPI 007	**Autor:** OFCG	
Fecha: 29/01/2020	**Requerimiento(s):** RF11, RF1, RF2, RF3, RF5, RF6, RF7 y RF9.	**ID ambiente:** AP2.

Objetivo: Comprobar la integridad del módulo Handler con el ShapeClassifier, TextClassifier, Graph, GramAnalyzer (Graph) y CodeGenerator.

VS1	**Tester:** DBM	**Fecha:** 10/05/2020	
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona una carpeta de los resultados de entrenamiento. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona o no la opción de usar GPU 10. Se ingresa el valor o no para regiones de interés (RoIs). 11. Se oprime el botón "Predict". 12. Se genera un archivo .c.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de máquina. - Usar GPU: No - RoIs: 32 - Acción: Oprimir botón "Predict".	Código en C del "Hola mundo".	No se genera nada, aparece error del tipo IndexError para el generador del código.
2	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Código en C del "hola mundo".	No se genera nada, aparece error del tipo IndexError para el generador del código.
3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir".	Código en C del "hola mundo".	No se genera nada, aparece error del tipo IndexError para el generador del

	- Imagen: Diagrama del hola mundo, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".		código.
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de máquina. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Código en C del "cálculo de factorial".	No se genera nada, aparece error del tipo IndexError para el generador del código.
5	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Código en C del "cálculo de factorial".	No se genera nada, aparece error del tipo IndexError para el generador del código..
6	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Código en C del "cálculo de factorial".	No se genera nada, aparece error del tipo IndexError para el generador del código..
7	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de máquina. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Código en C de "generar sucesión de Fibonacci".	No se genera nada, aparece error del tipo IndexError para el generador del código.

8	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de cuadrícula.. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Código en C de "generar sucesión de Fibonacci".	No se genera nada, aparece error del tipo IndexError para el generador del código.
9	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Código en C de "generar sucesión de Fibonacci".	No se genera nada, aparece error del tipo IndexError para el generador del código.

Handler, Shape Classifier, Text Classifier, Graph, Gram Analyzer, Code Generator y Flowchart Generator

EPI 008	**Autor:** OFCG	
Fecha: 29/01/2020	**Requerimiento(s):** RF11, RF1, RF2, RF3, RF5, RF6, RF7 y RF9.	**ID ambiente:** AP2.
Objetivo: Comprobar la integridad del módulo Handler con el ShapeClassifier, TextClassifier, Graph, Gram Analyzer (Graph), CodeGenerator y FlowchartGenerator.		

VS1	**Tester:** DBM	**Fecha:** 10/05/2020
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona una carpeta de los resultados de entrenamiento. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen.		

9. Se selecciona o no la opción de usar GPU 10. Se ingresa el valor o no para regiones de interés (RoIs). 11. Se oprime el botón "Predict". 12. Se genera un archivo .c. e imagen del diagrama digital reconstruido			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de máquina. - Usar GPU: No - RoIs: 32 - Acción: Oprimir botón "Predict".	Imagen digital del diagrama reconstruido.	No se genera nada, aparece error del tipo IndexError para el generador del código.
2	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen digital del diagrama reconstruido.	No se genera nada, aparece error del tipo IndexError para el generador del código.
3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen digital del diagrama reconstruido	No se genera nada, aparece error del tipo IndexError para el generador del código.
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de máquina. - Usar GPU: No - RoIs: 42	Imagen digital del diagrama reconstruido	No se genera nada, aparece error del tipo IndexError para el generador del código.

	- Acción: Oprimir botón "Predict".		
5	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen digital del diagrama reconstruido	No se genera nada, aparece error del tipo IndexError para el generador del código..
6	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen digital del diagrama reconstruido	No se genera nada, aparece error del tipo IndexError para el generador del código..
7	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de máquina. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen digital del diagrama reconstruido	No se genera nada, aparece error del tipo IndexError para el generador del código.
8	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo 8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de cuadrícula.. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen digital del diagrama reconstruido	No se genera nada, aparece error del tipo IndexError para el generador del código.
9	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: Modelo	Imagen digital del diagrama reconstruido	No se genera nada, aparece error del tipo

	8, flowchart_3b_model.hdf5 - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".		IndexError para el generador del código.

Handler y Shape Model

EPI 009	**Autor:** OFCG	
Fecha: 29/01/2020	**Requerimiento(s):** RF11 y RF4.	**ID ambiente:** AP2.
Objetivo: Comprobar la integridad del módulo Handler con el ShapeModel.		

VS1	**Tester:** DBM	**Fecha:** 10/05/2020	
Escenario: 1. Ejecutar el script de la GUI, "handler.py". 2. Oprimir el botón de entrenar. 3. Se abre una ventana para entrenar un modelo. 4. Seleccionar carpeta del dataset 5. Añadir un número (entero) de regiones de interés a considerar. 6. Seleccionar modelo pre-entrenado para cargar los pesos. 7. Añadir un número (entero) para el número de épocas. 8. Añadir un número (real) en el input de learning rate. 9. Check o no en el uso de la GPU. 10. Oprimir botón de iniciar.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- dataset_path: “/home/david/Escritorio/flowchart-3b(splitter)” - num_rois: 32 - pre-trained_model_path: “” - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘False’	Abre Terminal donde se muestra el progreso del entrenamiento	Correcto
2	- dataset_path: “/home/david/Escritorio/flowchart-3b(splitter)” - num_rois: 42 - pre-trained_model_path: “/home/david/Escritorio/handwritten-flowchart-with-cnn/model/vgg16_weights_tf_dim_ordering_tf_k	Abre Terminal donde se muestra el progreso del entrenamiento	Correcto

	ernels.h5” - num_epochs: 2 - learning_rate: 0.001 - use_gpu: ‘True’		
3	- dataset_path: “/home/david/Escritorio/flowchart-3b(splitter)” - num_rois: - pre-trained_model_path: “/home/david/Escritorio/handwritten-flowchart-with-cnn/model/vgg16_weights_tf_dim_ordering_tf_kernels.h5” - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘False’	Abre Terminal donde se muestra el progreso del entrenamiento	Correcto
4	- dataset_path: “/home/david/Escritorio/flowchart-3b(splitter)” - num_rois: - pre-trained_model_path: “” - num_epochs: - learning_rate: 0.00001 - use_gpu: ‘False’	Abre Terminal donde se muestra el progreso del entrenamiento	Correcto
5	- dataset_path: “/home/david/Escritorio/flowchart-3b(splitter)” - num_rois: - pre-trained_model_path: “” - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘False’	Abre Terminal donde se muestra el progreso del entrenamiento	Correcto

Handler y Shape Model

EPI 009	**Autor:** OFCG	
Fecha: 29/01/2020	**Requerimiento(s):** RF11 y RF4.	**ID ambiente:** AP2.
Objetivo: Comprobar la integridad del módulo Handler con el ShapeModel.		

VS2	**Tester:** DBM	**Fecha:** 25/05/2020
Escenario: 1. Ejecutar el script de la GUI, "handler.py". 2. Oprimir el botón de entrenar. 3. Se abre una ventana para entrenar un modelo. 4. Seleccionar carpeta del dataset 5. Añadir un número (entero) de regiones de interés a considerar. 6. Seleccionar modelo pre-entrenado para cargar los pesos.		

7. Añadir un número (entero) para el número de épocas. 8. Añadir un número (real) en el input de learning rate. 9. Check o no en el uso de la GPU. 10. Oprimir botón de iniciar.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- dataset_path: “/home/david/Escritorio/flowchart-3b_v3” - num_rois: 32 - pre-trained_model_path: “” - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘False’	Abre Terminal donde se muestra el progreso del entrenamiento	Correcto
2	- dataset_path: “/home/david/Escritorio/flowchart-3b_v3” - num_rois: 42 - pre-trained_model_path: “/home/david/Escritorio/handwritten-flowchart-with-cnn/model/vgg16_weights_tf_dim_ordering_tf_kernels.h5” - num_epochs: 2 - learning_rate: 0.001 - use_gpu: ‘True’	Abre Terminal donde se muestra el progreso del entrenamiento	Correcto
3	- dataset_path: “/home/david/Escritorio/flowchart-3b_v3” - num_rois: - pre-trained_model_path: “/home/david/Escritorio/handwritten-flowchart-with-cnn/model/vgg16_weights_tf_dim_ordering_tf_kernels.h5” - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘False’	Abre Terminal donde se muestra el progreso del entrenamiento	Correcto
4	- dataset_path: “/home/david/Escritorio/flowchart-3b_v3” - num_rois: - pre-trained_model_path: “” - num_epochs: - learning_rate: 0.00001 - use_gpu: ‘False’	Abre Terminal donde se muestra el progreso del entrenamiento	Correcto
5	- dataset_path: “/home/david/Escritorio/flowchart-3b_v3” - num_rois: - pre-trained_model_path: “” - num_epochs: 5 - learning_rate: 0.00001 - use_gpu: ‘False’	Abre Terminal donde se muestra el progreso del entrenamiento	Correcto

Handler y Preprocessor

EPI 001	**Autor:** OFCG	
Fecha: 25/01/2020	**Requerimiento(s):** RF11, RF1 y RF2.	**ID ambiente:** AP1.
Objetivo: Comprobar la integridad del módulo Handler con el Preprocessor.		

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020	
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona una carpeta de los resultados de entrenamiento. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona o no la opción de usar GPU 10. Se ingresa el valor o no para regiones de interés (RoIs). 11. Se oprime el botón "Predict". 12. Aparece la imagen modificada.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de máquina. - Usar GPU: No - RoIs: 32 - Acción: Oprimir botón "Predict".	Imagen con modificaciones	Correcto
2	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen con modificaciones	Correcto

3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen con modificaciones	Correcto
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de máquina. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen con modificaciones	Correcto
5	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen con modificaciones	Correcto
6	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen con modificaciones	Correcto
7	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de máquina. - Usar GPU: No - RoIs: 42	Imagen con modificaciones	Correcto

	- Acción: Oprimir botón "Predict".		
8	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de cuadrícula.. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen con modificaciones	Correcto
9	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen con modificaciones	Correcto

Handler, Preprocessor y Shape Classifier

EPI 003	**Autor:** OFCG	
Fecha: 25/01/2020	**Requerimiento(s):** RF11, RF1, RF2, RF3, RF5.	**ID ambiente:** AP1.
Objetivo: Comprobar la integridad del módulo Handler con el Preprocessor y ShapeClassifier.		

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona una carpeta de los resultados de entrenamiento. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona o no la opción de usar GPU 10. Se ingresa el valor o no para regiones de interés (RoIs). 11. Se oprime el botón "Predict". 12. Aparece una lista de nodos reconocidos en el diagrama de flujo.		

Caso	Entrada	Resultado esperado	Resultado
1	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de máquina. - Usar GPU: No - RoIs: 32 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto
2	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de cuadrícula. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto
3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto, aunque aparecen dos nodos de más uno de process y una línea derecha
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de máquina. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto
5	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de cuadrícula.	Una lista de objetos de tipo "Node".	Correcto

	- Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".		
6	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del factorial, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto
7	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de máquina. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto
8	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de cuadrícula.. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto
9	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama de calcular n-ésimo término de la sucesión de Fibonacci, fondo hoja de raya. - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Correcto

Handler, Preprocessor, Shape Classifier y Text Classifier

EPI 004	**Autor:** OFCG	
Fecha: 25/01/2020	**Requerimiento(s):** RF11, RF1, RF2, RF3, RF5 y RF6.	**ID ambiente:** AP1.
Objetivo: Comprobar la integridad del módulo Handler con el Preprocessor, ShapeClassifier y TextClassifier.		

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020	
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona una carpeta de los resultados de entrenamiento. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona o no la opción de usar GPU 10. Se ingresa el valor o no para regiones de interés (RoIs). 11. Se oprime el botón "Predict". 12. En la Terminal aparecen dos listas de nodos reconocidos del diagrama de flujo.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de máquina. - Usar GPU: Sí - RoIs: 32 - Acción: Oprimir botón "Predict".	Dos listas de objetos de tipo "Node".	Correcto
2	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del flujo IF. - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".	Dos listas de objetos de tipo "Node".	Correcto
3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/	Dos listas de objetos de tipo "Node".	Correcto

	- Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama “Imprimir pares hasta el 100” - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".		
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama “Imprimir N”. - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".	Una lista de objetos de tipo "Node".	Incorrecto, no se detectó una palabra, por lo que faltó un nodo de texto.

Handler, Preprocessor, Shape Classifier, Text Classifier y Graph

EPI 005	**Autor:** OFCG	
Fecha: 25/01/2020	**Requerimiento(s):** RF11, RF1, RF2, RF3, RF5, RF6 y RF7.	**ID ambiente:** AP1.
Objetivo: Comprobar la integridad del módulo Handler con el Preprocessor, ShapeClassifier, TextClassifier y Graph.		

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020	
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona una carpeta de los resultados de entrenamiento. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona o no la opción de usar GPU 10. Se ingresa el valor o no para regiones de interés (RoIs). 11. Se oprime el botón "Predict". 12. En la Terminal aparecen dos listas de nodos reconocidos del diagrama de flujo.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Oprimir botón		Correcto

	"Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de máquina. - Usar GPU: Sí - RoIs: 32 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph".	
2	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del flujo IF. - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph".	Correcto
3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama “Imprimir pares hasta el 100” - Usar GPU: No - RoIs: 42 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph".	Correcto
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama “Imprimir N”. - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph".	Incorrecto, KeyError: None.

Handler, Preprocessor, Shape Classifier, Text Classifier, Graph y Gram Analyzer

EPI 006	**Autor:** OFCG	
Fecha: 25/01/2020	**Requerimiento(s):** RF11, RF1, RF2, RF3, RF5, RF6 y RF7.	**ID ambiente:** AP1.
Objetivo: Comprobar la integridad del módulo Handler con el Preprocessor, ShapeClassifier, TextClassifier, Graph y Gramanalyzer (Graph).		

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020	
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona una carpeta de los resultados de entrenamiento. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona o no la opción de usar GPU 10. Se ingresa el valor o no para regiones de interés (RoIs). 11. Se oprime el botón "Predict". 12. En la Terminal aparecen dos listas de nodos reconocidos del diagrama de flujo.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de máquina. - Usar GPU: Sí - RoIs: 32 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph" validado.	Correcto
2	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del flujo IF. - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph" validado.	Correcto
3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama “Imprimir pares hasta el 100” - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".	Lista de adyacencia de un objeto tipo "Graph" validado.	Correcto
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/	Lista de adyacencia de un objeto tipo "Graph" validado.	Incorrecto, no se detectó una palabra por lo

	- Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama “Imprimir N”. - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".		que faltó un nodo de texto

Handler, Preprocessor, Shape Classifier, Text Classifier, Graph, Gram Analyzer y Code Generator

EPI 007	**Autor:** OFCG	
Fecha: 25/01/2020	**Requerimiento(s):** RF11, RF1, RF2, RF3, RF5, RF6, RF7 y RF9.	**ID ambiente:** AP1.
Objetivo: Comprobar la integridad del módulo Handler con el Preprocessor, ShapeClassifier, TextClassifier, Graph, GramAnalyzer (Graph) y CodeGenerator.		

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020	
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona una carpeta de los resultados de entrenamiento. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona o no la opción de usar GPU 10. Se ingresa el valor o no para regiones de interés (RoIs). 11. Se oprime el botón "Predict". 12. Se genera archivo de código fuente .c.			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de máquina. - Usar GPU: Sí - RoIs: 32 - Acción: Oprimir botón "Predict".	Código en C del "hola mundo".	Correcto

2	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del flujo IF. - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".	Código en C del "if".	Correcto
3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama “Imprimir pares hasta el 100” - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".	Código en C de la impresión de pares.	Correcto
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama “Imprimir N”. - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".	Código en C del "imprimir n".	KeyError: None

Handler, Preprocessor, Shape Classifier, Text Classifier, Graph, Gram Analyzer, Code Generator y Flowchart Generator.

EPI 008	**Autor:** OFCG	
Fecha: 25/01/2020	**Requerimiento(s):** RF11, RF1, RF2, RF3, RF5, RF6, RF7 y RF9.	**ID ambiente:** AP1.
Objetivo: Comprobar la integridad del módulo Handler con el Preprocessor, ShapeClassifier, TextClassifier, Graph, GramAnalyzer (Graph), CodeGenerator y FlowchartGenerator.		

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020
Escenario: 1. Activar el ambiente virtual. 2. Ejecutar el script de la GUI, "handler.py". 3. Oprimir el botón "Recognize flowchart".		

4. Se abre la ventana de "Recognize flowchart".
5. Se selecciona una carpeta de los resultados de entrenamiento.
6. Oprimir el botón de "Select image".
7. Se abre una ventana selector de archivos.
8. Se selecciona la imagen.
9. Se selecciona o no la opción de usar GPU
10. Se ingresa el valor o no para regiones de interés (RoIs).
11. Se oprime el botón "Predict".
12. Se genera archivo diagrama de flujo reconstruido.

Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del hola mundo, fondo hoja de máquina. - Usar GPU: Sí - RoIs: 32 - Acción: Oprimir botón "Predict".	Imagen digital del diagrama reconstruido.	Correcto
2	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama del flujo IF. - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen digital del diagrama reconstruido.	Correcto
3	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama “Imprimir pares hasta el 100” - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen digital del diagrama reconstruido.	Correcto
4	- Acción: Oprimir botón "Recognize flowchart". - Carpeta de resultados de entrenamiento: 9/ - Acción: Oprimir botón "Select image". - Acción: Oprimir botón "Abrir". - Imagen: Diagrama “Imprimir N”. - Usar GPU: Sí - RoIs: 42 - Acción: Oprimir botón "Predict".	Imagen digital del diagrama reconstruido.	KeyError: None

K.3 Pruebas de sistema

Detección completa

EPS 001	**Autor:** DBM	
Fecha: 29/01/2020	**Requerimiento(s):** RF1, FR2, RF3, RF5, RF6, RF7, RF8, RF9 y RF10.	**ID ambiente:** AP2 y AP1 (VS2).

Objetivo: Determinar si es posible generar el código en C y diagrama digital a partir de una foto de un diagrama de flujo trazado a mano.

VS1	**Tester:** DBM	**Fecha:** 10/05/2020	
Escenario: 1. Ejecutar el script de la GUI, "handler.py". 2. Oprimir el botón de entrenar. 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona el modelo de reconocimiento de figuras que se desea utilizar. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona la opción de usar GPU o no. 10. Se ingresar el número de RoIs. 11. Se oprime el botón "Predict". 12. Aparece la ventana "Results".			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Hola mundo, fondo de máquina.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	No se genera nada, aparece error del tipo IndexError para el generador del código.
2	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Hola mundo, fondo de cuadrícula.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	No se genera nada, aparece error del tipo TypeError, en el generador del grafo.
3	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Hola mundo, fondo de raya.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	No se genera nada, aparece error del tipo IndexError para el generador del código.
4	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Factorial, fondo de máquina.	En la ventana de resultados muestra el diagrama digital y el	No se genera nada, aparece error del tipo

		código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	TypeError, en el generador del grafo.
5	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Factorial, fondo de cuadrícula.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	No se genera nada, aparece error del tipo IndexError para el generador del código.
6	- Carpeta de modelo de figuras: 8/ - Imagen diagrama:Factorial, fondo de raya.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	No se genera nada, aparece error del tipo TypeError, en el generador del grafo.
7	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Calcular n-ésimo término de la sucesión de Fibonacci, fondo de máquina.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	No se genera nada, aparece error del tipo TypeError, en el generador del grafo.
8	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Calcular n-ésimo término de la sucesión de Fibonacci,, fondo de cuadrícula.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	No se genera nada, aparece error del tipo TypeError, en el generador del grafo.
9	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Calcular n-ésimo término de la sucesión de Fibonacci, fondo de raya.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	No se genera nada, aparece error del tipo TypeError, en el generador del grafo.
10	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Suma de 2 números incorrecto,	Mostrar ventana de error de que no fue posible	Solo se muestra una

	fondo de hoja de máquina.	generar el diagrama y código, dado que el diagrama de flujo es incorrecto.	excepción en la Terminal, error del tipo TypeError en el generador de grafo.
11	- Carpeta de modelo de figuras: “” - Imagen diagrama: Hola mundo, fondo de máquina.	Mostrar mensaje de error: "Debe seleccionar un modelo entrenado".	Correcto
12	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: “”	Mostrar mensaje de error: "Debe seleccionar una imagen de un diagrama de flujo o que la imagen no fue encontrada".	Correcto

VS2	**Tester:** OFCG	**Fecha:** 25/06/2020	
Escenario: 1. Ejecutar el script de la GUI, "handler.py". 2. Oprimir el botón de entrenar. 3. Oprimir el botón "Recognize flowchart". 4. Se abre la ventana de "Recognize flowchart". 5. Se selecciona el modelo de reconocimiento de figuras que se desea utilizar. 6. Oprimir el botón de "Select image". 7. Se abre una ventana selector de archivos. 8. Se selecciona la imagen. 9. Se selecciona la opción de usar GPU o no. 10. Se ingresar el número de RoIs. 11. Se oprime el botón "Predict". 12. Aparece la ventana "Results".			
Caso	**Entrada**	**Resultado esperado**	**Resultado**
1	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Hola mundo, fondo de máquina.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	Correcto
2	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Hola mundo, fondo de cuadrícula.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde	CUDA_ERROR_OUT_OF_MEMORY

		se guardan los resultados.	
3	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Hola mundo, fondo de raya.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	CUDA_ERROR_OUT_OF_MEMORY
4	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Factorial, fondo de máquina.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	CUDA_ERROR_OUT_OF_MEMORY
5	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Factorial, fondo de cuadrícula.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	KeyError: None, No se detectó un nodo de texto
6	- Carpeta de modelo de figuras: 8/ - Imagen diagrama:Factorial, fondo de raya.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	CUDA_ERROR_OUT_OF_MEMORY (Probar con la compu de la escuela)
7	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Calcular n-ésimo término de la sucesión de Fibonacci, fondo de máquina.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	Correcto
8	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Calcular n-ésimo término de la sucesión de Fibonacci,, fondo de cuadrícula.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se	TypeError: can only concatenate str (not "NoneType") to str

		genera una carpeta donde se guardan los resultados.	
9	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Calcular n-ésimo término de la sucesión de Fibonacci, fondo de raya.	En la ventana de resultados muestra el diagrama digital y el código fuente en C; y en la consola el resultado de la compilación. Y se genera una carpeta donde se guardan los resultados.	No se detecta un nodo de texto.
10	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: Suma de 2 números incorrecto, fondo de hoja de máquina.	Mostrar ventana de error de que no fue posible generar el diagrama y código, dado que el diagrama de flujo es incorrecto.	Correcto
11	- Carpeta de modelo de figuras: “” - Imagen diagrama: Hola mundo, fondo de máquina.	Mostrar mensaje de error: "Debe seleccionar un modelo entrenado".	Correcto
12	- Carpeta de modelo de figuras: 8/ - Imagen diagrama: “”	Mostrar mensaje de error: "Debe seleccionar una imagen de un diagrama de flujo o que la imagen no fue encontrada".	Correcto

Entrenamiento

EPS 002, VS1, VS2	**Autor:** DBM	
Fecha: 29/01/2020	**Requerimiento(s):** RF4 y RF10.	**ID ambiente:** AP2 (VS1) y AP1 (VS2).
Objetivo: Determinar si se puede entrenar la CNN para figuras y textos desde la GUI.		

Esta prueba, para ambas versiones del sistema resulta equivalente a la prueba de integración EPI 009, por lo que ya no se muestran los resultados.

Apéndice L

En este apéndice se encuentran documentos relacionados a los datasets empleados en los modelos de reconocimiento.

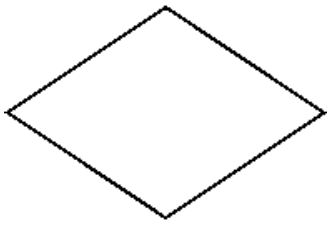
L.1 Especificación del dataset para modelo de figuras y conectores

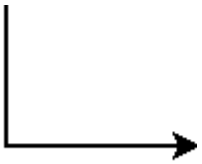
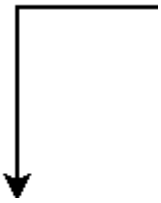
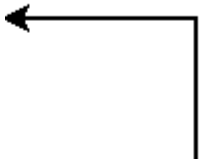
Especificación del dataset

Para la obtención del dataset se pedirá la ayuda de 40 alumnos de la escuela, los cuales serán encargados de dibujar las figuras en hojas (con los distintos fondos) y en un formato de 12 figuras por hoja, y otros 20 alumnos para hacer el etiquetado o segmentación de las figuras en las imágenes digitales capturadas de las hojas con las figuras.

Para la obtención del dataset

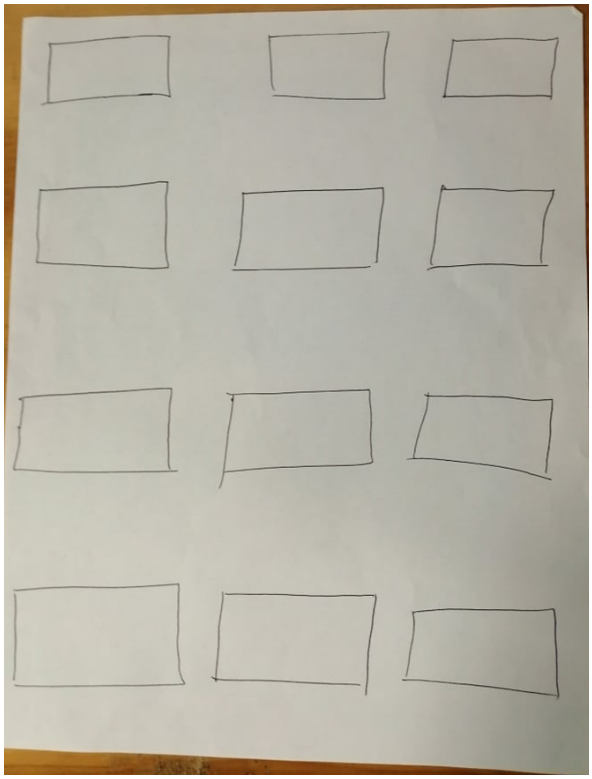
Clases

Nombre	Símbolo
start_end	
scan	
process	
decision	
print	
arrow_line_up	

arrow_line_rigth	
arrow_line_down	
arrow_line_left	
arrow_rectangle_up	
arrow_rectangle_rigth	
arrow_rectangle_down	
arrow_rectangle_left	

Especificación para el dibujo de las imágenes

Por cada figura se requiere que se dibuje en 3 plantillas, en donde la primera es un fondo de hoja de máquina, otra en hoja de raya de libreta y la tercera en hoja con fondo cuadriculado de libreta (el cuadro, ya sea chico o grande). Una plantilla es una hoja con 12 figuras, un ejemplo de cómo se ve una plantilla ya con figuras dibujadas de la clase 'proceso' es la siguiente:



En resumen, cada alumno debe entregar 39 hojas (13 de hojas de máquina, 13 de hojas cuadriculadas y 13 de hojas de raya), la siguiente tabla detalla el trabajo a entregar:

Tipo	# Figuras en hoja en blanco	# Figuras en hoja cuadriculada	# Figuras en hoja de raya	# Total de figuras
Inicio/terminal	12	12	12	36
Entrada de datos	12	12	12	36
Proceso	12	12	12	36
Decisión	12	12	12	36

Imprimir	12	12	12	36
Flecha recta arriba	12	12	12	36
Flecha recta derecha	12	12	12	36
Flecha recta abajo	12	12	12	36
Flecha recta izquierda	12	12	12	36
Flecha cuadrada arriba	12	12	12	36
Flecha cuadrada derecha	12	12	12	36
Flecha cuadrada abajo	12	12	12	36
Flecha cuadrada izquierda	12	12	12	36
Total				468

Indicaciones específicas:

- Dibujar las figuras lo más natural posible, es decir, no usar regla, no usar mucho tiempo para hacerlo estético.
- Usar pluma/bolígrafo de tinta negra o lápiz (si es con lápiz cuidar que no sea tenue).
- Cuidar que no haya superposición entre las figuras, es decir, que no se empalmen, para ello dejar un espacio alrededor de la figura trazada que consideren suficiente ya que luego se necesitará recortar las figuras.
- No dibujar otros símbolos ajenos a las figuras.
- No usar hojas recicladas, es por ello que debe ser una hoja que esté limpia por ambas páginas.
- No arrugar las hojas.

Fecha para entregar: A más tardar el lunes, 27 de enero del 2020.
Lugar para entregar: Centro de Desarrollo de Software (CDS).

Estaremos recibiendo las hojas en un horario de 10:30 horas hasta las 16:30.

Se agradece mucho su colaboración, cualquier duda pueden contactarnos por medio de correo electrónico:

- Correo de Onder Francisco: onderfra@gmail.com
- Correo de David Betancourt: davbetm@gmail.com

O bien, nos encuentran en Facebook:

- Username de Onder Francisco: OnderFraWolf
- Username de David Betancourt: dabetm

Para el etiquetado o segmentación del dataset

Deberán utilizar el programa labelImg (https://github.com/tzutalin/labelImg) para realizar el etiquetado y obtener las anotaciones en formato .xml.

Cada alumno(a) debe entregar 2 carpetas con la siguiente estructura:

- Images
  - 1.png
  - …….
- Annotations
  - 1.xml
  - …….

Donde los objetos etiquetado en 1.xml corresponden a la imagen 1.jpg de la carpeta “Images”.

Apéndice M

En este apéndice encuentran secciones referentes a la configuración de los equipos de cómputo de trabajo.

M.1 Requerimientos para configurar el entorno en Anaconda

La siguiente lista representa la lista de paquetes instalados en el entorno Conda en el que corre el sistema. Cabe decir que muchos son dependencias, ya que no se usan todos directamente.

absl-py==0.9.0

astor==0.7.1

blinker==1.4

brotlipy==0.7.0

cachetools==3.1.1

certifi==2020.4.5.1

cffi==1.14.0

chardet==3.0.4

click==7.1.2

cryptography==2.9.2

cycler==0.10.0

decorator==4.4.2

editdistance==0.5.3

efficientnet==1.0.0

essential-generators==0.9.2

fonttools==4.9.0

gast==0.2.2

google-auth==1.14.3

google-auth-oauthlib==0.4.1

google-pasta==0.2.0

graphviz==0.14

grpcio==1.27.2

h5py==2.10.0

idna==2.9

imagecodecs==2020.2.18

imageio==2.8.0

imgaug==0.4.0

importlib-metadata==1.6.0

imutils==0.5.3

joblib==0.14.1

Keras==2.3.1

Keras-Applications==1.0.8

keras-ocr==0.8.3

Keras-Preprocessing==1.1.0

kiwisolver==1.2.0

llvmlite==0.32.1

Mako==1.1.0

Markdown==3.2.2

MarkupSafe==1.1.1

matplotlib==3.2.1

networkx==2.4

numba==0.47.0

numpy==1.18.1

oauthlib==3.0.1

olefile==0.46

opencv-python==4.2.0.34

opt-einsum==0+untagged.56.g2664021.dirty

pandas==1.0.3

Pillow==7.1.2

protobuf==3.11.4

pyasn1==0.4.8

pyasn1-modules==0.2.7

pyclipper==1.1.0.post3

pycparser==2.20

pydot==1.4.1

pygpu==0.7.6

PyJWT==1.7.1

pyOpenSSL==19.1.0

pyparsing==2.4.7

PySocks==1.7.1

python-dateutil==2.8.1

python-mnist==0.7

pytz==2020.1

PyWavelets==1.1.1

PyYAML==5.3.1

requests==2.23.0

requests-oauthlib==1.2.0

rsa==4.0

scikit-image==0.17.2

scikit-learn==0.22.1

scipy==1.4.1

Shapely==1.7.0

six==1.14.0

tensorboard==2.1.1

tensorflow==2.1.0

tensorflow-estimator==2.1.0

termcolor==1.1.0

Theano==1.0.4

tifffile==2020.5.11

tornado==6.0.4

tqdm==4.46.0

urllib3==1.25.9

validators==0.15.0

Werkzeug==1.0.1

wrapt==1.12.1

zipp==3.1.0

Apéndice N

Las evaluaciones sobre el modelo de figuras y conectores fueron con el fin de detectar los principales problemas de los modelos entrenados y con ello proponer cambios sobre el modelo para mejorarlos. Y al fin de determinar el modelo de detección de figuras y conectores empleado en la liberación del sistema.

N.1 Evaluación modelo de figuras y conectores, versión 1 del sistema

Se evaluaron los modelos con mejores métricas obtenidas luego de su entrenamiento, la siguiente tabla refleja los resultados en la tarea de reconocer figuras y componentes definidos, sobre dos conjuntos (set1 y set2) conteniendo 9 imágenes de diagramas de flujo cada uno, de 3 algoritmos

diferentes (“Hola mundo”, “Cálculo del factorial” y “Secuencia de Fibonacci”), en el set 1 las imágenes no están preprocesadas, mientras que en el segundo están con binarización.

Nota: Todos los modelos entrenados aquí usaron como backbone VGG-16, la cual es una arquitectura de red neuronal convolucional de 16 capas.

Rendimiento sobre set 1 ≡ RS1.

Rendimiento sobre set 2 ≡ RS2.

El rendimiento se expresa como un ratio de diagramas completamente reconocidos correctamente, respecto a figuras y conectores.

ID	Loss	mAP	Descripción	RS1	RS2	Principales problemas
3	0.3130	0.9852	El modelo se entrenó con los hiperparámetros por defecto.	0/9	0/9	- En algunas imágenes binarizadas con fondo de raya o cuadrícula se engruesan generando que el modelo detecte figuras en el fondo. - Flechas no se reconocen. - Los bounding box para símbolos alargados solo se abarca poco parcialmente.
8	0.3271	0.9799	En el entrenamiento se usó aumento de datos (volteo horizontal y vertical).	1/9	0/9	- En algunas imágenes binarizadas los contornos se han hecho más delgados respecto a la versión original, provocando que el modelo no detecte las figuras con ese detalle. - Los bounding box para símbolos alargados solo se abarca poco parcialmente. - Flechas rectas se detectan muy poco.
13	0.3576	**0.9846**	Se modifican los anchors: {64^2, 128^2, 256^2}, {2:1,1:1, 1:2}	0/9	0/9	- Baja detección de flechas rectas. - Baja detección del símbolo proceso. - Confusión de flechas rectangulares con líneas perpendiculares en los símbolos de proceso.
18	0.3671	0.9818	Se modifica el número de regiones de interés a 15.	0/9	0/9	- Baja detección de flechas rectas. - Baja detección del símbolo proceso. - Confusión de flechas rectangulares con líneas perpendiculares en los símbolos de proceso.
23	0.3790	0.9839	Se modifica el max overlap RPN a 0.65.	0/9	0/9	- Baja detección de flechas. - Baja detección del símbolo de lectura de datos. - Confusión de flechas rectangulares con líneas perpendiculares en los símbolos de proceso.

28	0.3023	**0.9706**	Se ha cambiado el optimizador a RMSprop.	0/9	0/9	- Baja detección de flechas. - Baja detección del símbolo de lectura de datos. - Baja detección del símbolo de proceso. - Confusión de flechas rectangulares con líneas perpendiculares en los símbolos de proceso.

De acuerdo a las métrica mAP sobre el dataset de validación, el mejor modelo es el modelo con id = 13, sin embargo, los mejores resultados a partir de un análisis visual en el dataset de evaluación es el modelo con id = 8.

N.2 Evaluación modelo de figuras y conectores, versión 2 (final) del sistema

Se evaluaron los modelos obtenidos luego de su entrenamiento, la siguiente tabla refleja los resultados en la tarea de reconocer figuras y componentes definidos, sobre dos conjuntos (set10 y set12) conteniendo 56 fotos de diagramas de flujo cada uno. El set 12 tiene las fotos preprocesadas con el algoritmo de enmascaramiento de enfoque.

La especificación del contenido de los conjuntos se resume en la siguiente tabla:

Archivo	**Descripción del diagrama de flujo**	**Archivo**	**Descripción del diagrama de flujo**
1.jpg	Hola mundo, fondo de hoja de máquina.	29.jpg	Calcular área de un cuadrado, fondo de hoja de máquina.
2.jpg	Hola mundo, fondo de hoja de máquina.	30.jpg	Calcular área de un cuadrado, fondo de hoja de máquina.

3.jpg	Hola mundo, fondo de cuadrícula.	31.jpg	Calcular área de un cuadrado, fondo de cuadrícula.
4.jpg	Hola mundo, fondo de raya.	32.jpg	Calcular área de un cuadrado, fondo de raya.
5.jpg	Cálculo del factorial, fondo de hoja de máquina.	33.jpg	Suma de dos enteros, fondo de hoja de máquina.
6.jpg	Cálculo del factorial, fondo de hoja de máquina.	34.jpg	Suma de dos enteros, fondo de hoja de máquina.
7.jpg	Cálculo del factorial, fondo de cuadrícula.	35.jpg	Suma de dos enteros, fondo de cuadrícula.
8.jpg	Cálculo del factorial, fondo de raya.	36.jpg	Suma de dos enteros, fondo de raya.
9.jpg	Cálculo del e-nésimo término de la sucesión de Fibonacci, fondo de hoja de máquina.	37.jpg	Determinar si un número entero es par, fondo de hoja de máquina.
10.jpg	Cálculo del e-nésimo término de la sucesión de Fibonacci, fondo de hoja de máquina.	38.jpg	Determinar si un número entero es par, fondo de hoja de máquina.
11.jpg	Cálculo del e-nésimo término de la sucesión de Fibonacci, fondo de cuadrícula.	39.jpg	Determinar si un número entero es par, fondo de cuadrícula.
12.jpg	Cálculo del e-nésimo término de la sucesión de Fibonacci, fondo de raya.	40.jpg	Determinar si un número entero es par, fondo de raya.
13.jpg	Contar dígitos de un entero, fondo de hoja de máquina.	41.jpg	Condicional anidado, fondo de hoja de máquina.
14.jpg	Contar dígitos de un entero, fondo de hoja de máquina.	42.jpg	Condicional anidado, fondo de hoja de máquina.
15.jpg	Contar dígitos de un entero, fondo de cuadrícula.	43.jpg	Condicional anidado, fondo de cuadrícula.
16.jpg	Contar dígitos de un entero, fondo de raya.	44.jpg	Condicional anidado, fondo de raya.
17.jpg	Imprimir números pares hasta el 100, fondo de hoja de máquina.	45.jpg	Imprimir al revés una cadena, fondo de hoja de máquina.
18.jpg	Imprimir números pares hasta el	46.jpg	Imprimir al revés una cadena, fondo

	100, fondo de hoja de máquina.		de hoja de máquina.
19.jpg	Imprimir números pares hasta el 100, fondo de cuadrícula.	47.jpg	Imprimir al revés una cadena, fondo de cuadrícula.
20.jpg	Imprimir números pares hasta el 100, fondo de raya.	48.jpg	Imprimir al revés una cadena, fondo de raya.
21.jpg	Imprimir números impares hasta el 100, fondo de hoja de máquina.	49.jpg	Imprimir números impares hasta el 100, fondo de hoja de máquina, trazos de colores.
22.jpg	Imprimir números impares hasta el 100, fondo de hoja de máquina.	50.jpg	Calcular raíz cuadrada, fondo de hoja de máquina, trazos de colores.
23.jpg	Imprimir números impares hasta el 100, fondo de cuadrícula.	51.jpg	Calcular área de un cuadrado, fondo de hoja de máquina, trazos de colores.
24.jpg	Imprimir números impares hasta el 100, fondo de raya.	52.jpg	Sumar dos números, fondo de hoja de máquina, trazos de colores.
25.jpg	Calcular raíz cuadrada, fondo de hoja de máquina.	53.jpg	Hola mundo, fondo de cuadrícula, trazos de color verde.
26.jpg	Calcular raíz cuadrada, fondo de hoja de máquina.	54.jpg	Contar dígitos de un entero, fondo de cuadrícula, trazos de color rosa.
27.jpg	Calcular raíz cuadrada, fondo de cuadrícula.	55.jpg	Imprimir números pares hasta el 100, fondo de cuadrícula, trazos de color rosa.
28.jpg	Calcular raíz cuadrada, fondo de raya.	56.jpg	Cálculo del factorial, fondo de cuadrícula, trazos de color morado..

Acá se puede descargar el set 10 usado para la evaluación: https://bit.ly/3dB3rqM .

Nota: Cabe destacar que respecto la v1 del sistema, el dataset se ha modificado, removiendo las clases para flechas rectangulares, por lo que los diagramas dibujados solo usan flechas rectas para definir el flujo. Y por otra parte se ha usado un conjunto más grande para evaluar los modelos, debido a que los reconocimientos fueron más aceptables respecto lo obtenido para la versión 1.

Aclaraciones:

Rendimiento sobre set 10 ≡ R10.

Rendimiento sobre set 12 ≡ R12.

El rendimiento se expresa como un ratio de diagramas completamente reconocidos correctamente, respecto a figuras y conectores.

ID	Loss	mAP	Descripción/Cambios	Backbone	R10	R12	Principales problemas
2	0.4731	0.9679	Configuración por defecto	VGG-16	19/56	21/56	- No detectar algunas flechas.
3	0.3000	0.9607	Aumento de datos (rotación e inclinación), con anchors modificados: sizes=[64, 128, 256, 512] ratios= [[1, 1], [1, 2], [2, 1], [1, 3], [3, 1]]	VGG-16	16/56	19/56	- No detectar algunas flechas.
4	0.3471	0.9807	Aumento de datos (inclinación), con anchors modificados: sizes=[64, 128, 256, 512] ratios=[[1, 1], [1./math.sqrt(2), 2./math.sqrt(2)], [2./math.sqrt(2), 1./math.sqrt(2)], [1./math.sqrt(2), 3./math.sqrt(2)], [3./math.sqrt(2), 1./math.sqrt(2)]	VGG-16	13/56	11/56	- No detectar algunas flechas.
5	0.4311	0.9762	Aumento de datos (rotación), con anchors modificados: sizes=[32, 64, 128, 256, 512] ratios=[[1, 1], [1./math.sqrt(2), 2./math.sqrt(2)], [2./math.sqrt(2), 1./math.sqrt(2)]]	VGG-16	20/56	21/56	- No detectar algunas flechas.
6	0.4517	0.9669	Aumento de datos (rotación), con anchors modificados: sizes=[48, 64, 128, 256, 512] ratios=[[1, 1], [1./math.sqrt(2), 2./math.sqrt(2)], [2./math.sqrt(2), 1./math.sqrt(2)]]	VGG-16	23/56	21/56	- No detectar algunas flechas. - Dobles detecciones.
7	0.5479	0.9630	Aumento de datos (rotación e inclinación).	VGG-16	29/56	26/56	- No detectar algunas flechas. - Dobles detecciones.
8	0.3811	0.9689	Aumento de datos (ajustar contraste, corrección gamma, rotación e inclinación), con anchors modificados: sizes=[64, 128, 256] ratios=[[1, 1], [1./math.sqrt(2), 2./math.sqrt(2)], [2./math.sqrt(2), 1./math.sqrt(2)],	VGG-16	32/56	35/56	- No detectar algunas flechas. - Dobles detecciones.

			[1./math.sqrt(2), 4./math.sqrt(2)], [4./math.sqrt(2), 1./math.sqrt(2)]				
9	0.3415	0.9542	Aumento de datos (rotación e inclinación), con anchors modificados: sizes=[64, 128, 256] ratios=[[1, 1], [1./math.sqrt(2), 2./math.sqrt(2)], [2./math.sqrt(2), 1./math.sqrt(2)], [1./math.sqrt(2), 4./math.sqrt(2)], [4./math.sqrt(2), 1./math.sqrt(2)]	VGG-16	38/56	42/56	- Dobles detecciones
10	0.3682	0.9767	Igual que 9	VGG-19	29/56	28/56	- No detectar algunas flechas. - Dobles detecciones.
11	0.4183	0.9680	Aumento de datos (con rotación e inclinación, mayor probabilidad para la inclinación), con anchors modificados: sizes=[64, 128, 256] ratios=[[1, 1], [1./math.sqrt(2), 2./math.sqrt(2)], [2./math.sqrt(2), 1./math.sqrt(2)], [1./math.sqrt(2), 4./math.sqrt(2)], [4./math.sqrt(2), 1./math.sqrt(2)]	VGG-19	26/56	27/56	- No detectar algunas flechas.
14	0.5576	0.9475	Igual que 9	VGG-11	2/56	11/56	- Doble detección con confusión entre clases. - No detectar algunos pocos scan, process y arrow_line_down.
15	0.6635	0.9477	Igual que 9	ResNet-50		3/56	- Baja detección de clases, principalmente de flechas.

De acuerdo a las métrica mAP sobre el dataset de validación, el mejor modelo es el modelo con id 4, sin embargo, los mejores resultados a partir de un análisis visual en el dataset de evaluación es el modelo con id 9.

Apéndice O

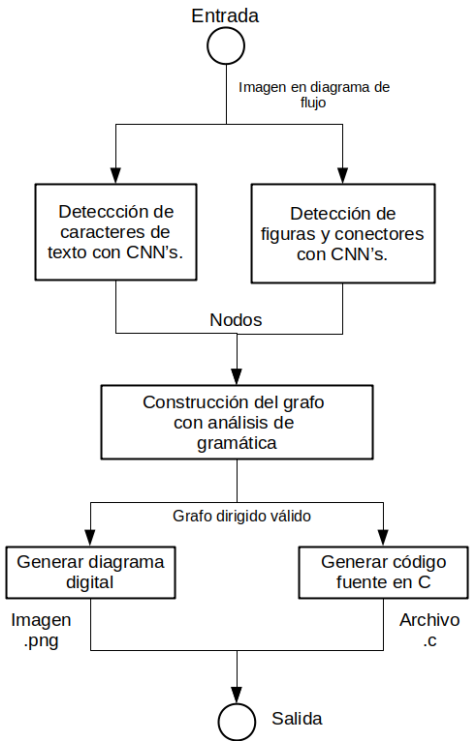
Las evaluaciones realizadas a las versiones completas de los sistemas con los que se experimentó resolver el problema del proyecto.

O.1 Evaluación del sistema para versión 1

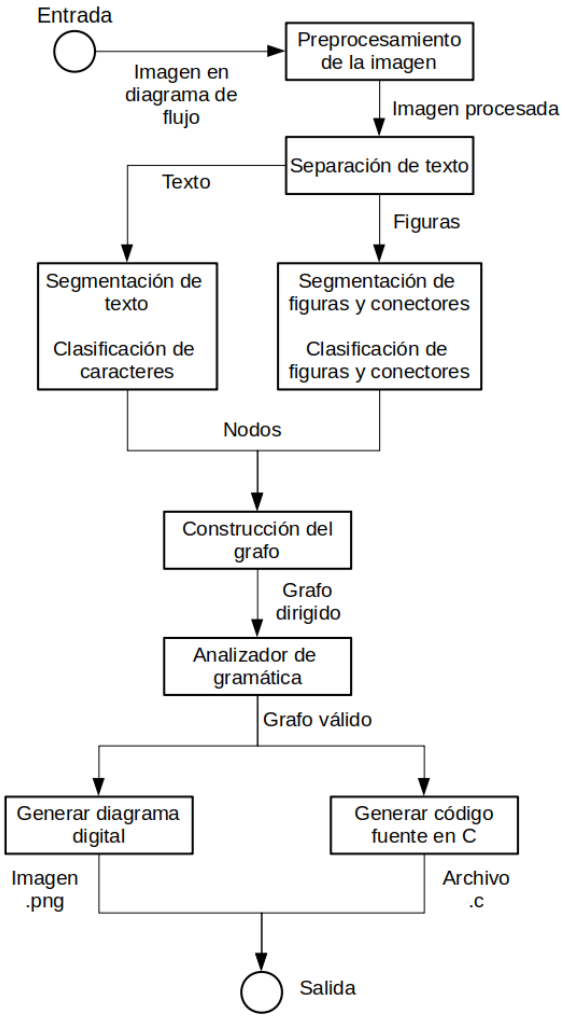
Enlace al repositorio:
https://github.com/dbetm/handwritten-flowchart-with-cnn/tree/v1.0

La implementación completa para la primer versión del sistema, utiliza el pipeline que se muestra abajo para resolver el problema de reconocimiento de diagramas de flujo hechos a mano. Hay 3 aspectos relevantes a destacar, los cuales derivan de los cambios respecto al pipeline que se propuso inicialmente. El primero es el no necesitar de un preprocesamiento de la imagen, ya que por el lado del reconocimiento de figuras y conectores, el umbralizar la imagen hacía empeorar la tarea de reconocimiento como se podrá observar en las pruebas de detección mostradas casi al final de este documento de evaluación; el segundo aspecto el cambio de la idea de segmentación, como tal no se aplica ningún algoritmo de segmentación para distinguir entre texto y no texto, en su lugar, se manda la imagen completa a dos detectores de objetos diferentes, en ese caso el de texto y el de figuras y conectores, con lo que al final de cuentas se logran tener

dos tipos de nodos reconocidos por separado (lo cual era el objetivo final de la segmentación); el tercer aspecto es sobre la unificación del paso de construcción del grafo y analizador de gramática, que se une en un paso llamado “Construcción del grafo” con análisis de gramática.



Pipeline empleado en la primer versión del sistema.



Pipeline inicialmente planeado.

Para la detección de figuras y conectores se especificó y recopiló un dataset (https://bit.ly/31lX1sV) consistiendo en fotos de plantillas (sobre 3 tipos de fondos diferentes) de 13 clases diferentes de símbolos ("arrow_line_left", "arrow_rectangle_up", "start_end", "arrow_rectangle_left", "arrow_rectangle_down", "arrow_line_up", "arrow_line_right", "arrow_line_down", "arrow_rectangle_right", "print", "scan", "process", "decision"). Además, se utilizó el modelo de detección de objetos Faster R-CNN con backbone VGG-16, se describe brevemente en el apartado de abajo.

Por otra parte, para la detección del texto se utilizó un modelo existente usado en un paquete llamado Keras OCR (https://pypi.org/project/keras-ocr/) que también se describe brevemente abajo.

Tanto el modelo de detección de texto, así como el de figuras fue implementado usando Keras con TensorFlow como backend.

Para la generación del diagrama de flujo en imagen digital se ha empleado un paquete llamado Graphviz for Python (https://pypi.org/project/graphviz/).

Los requerimientos para crear un entorno en Conda, y hacer pruebas con el sistema, se pueden encontrar en el siguiente enlace: https://bit.ly/3i1BBYd .

Finalmente, el diagrama de Conway que determina la gramática para construir el grafo es el siguiente:

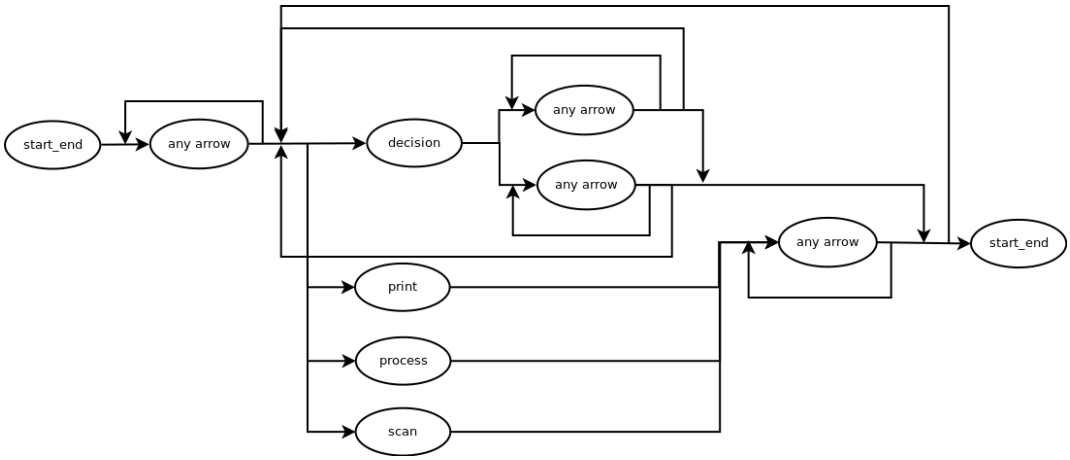
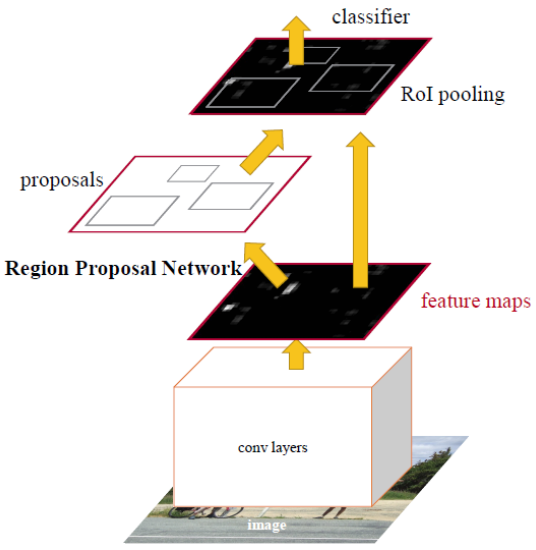


Diagrama de Conway que determina la gramática.

Observar que solo se cuenta con nodos terminales.

Modelo de detección de figuras y conectores

La idea de este modelo de detección de objetos fue presentado en https://arxiv.org/pdf/1506.01497.pdf y presenta una arquitectura para la extracción de características (backbone) que se llama VGG-16 misma que fue presentado en https://arxiv.org/pdf/1409.1556.pdf. La estructura de Faster R-CNN es la siguiente:



Se puede apreciar que consta de 2 fases para lograr la detección de objetos, y se aprovecha el mapa de características obtenido por el backbone.

Este modelo fue entrenado:

- Con 500 épocas en un tiempo aproximado de 6 horas.
- De forma local en un equipo con:
    - Software: Conda 4.7.12, Tensorflow 2.0.0, Keras 2.3.1, SO: GNU/Linux Ubuntu 18.04 LTS.
    - Hardware: 16 GB RAM, 123 SSD, Intel core i7-9700, GPU GeForce GTX 1660.
- Utilizando los kernels, para inicializar los pesos, de un modelo pre-entrenado que se encuentra en: https://github.com/fchollet/deep-learning-models/releases .

Los mejores resultados (para las métricas mAP y loss) variando algunos hiperparámetros se muestran en la siguiente tabla:

id	Descripción de la variación	mAP	loss
1	Ninguna, hiperparámetros por defecto	0.9852	0.3130
2	Aumento de datos: use_horizontal_flips=True use_vertical_flips=True	0.9799	0.3271
3	Anchors: scales={64^2,128^2,256^2} ratios={2:1,1:1,1:2}	0.9876	0.3576
4	RoIs: 15	0.9818	0.3671
5	RPN max overlap: 0.65	0.9839	0.3790
6	Optimizer: RMSprop	0.9706	0.3088

Los hiperparámetros por defecto son:

- use_horizontal_flips = False
- use_vertical_flips = False
- epoch_lenght = 32
- learning_rate = 0.00001
- anchor_box_scales = (128, 256, 512)

- anchor_box_ratios = {(1, 1), (1./math.sqrt(2), 2./math.sqrt(2)), (2./math.sqrt(2), 1./math.sqrt(2))}
- min_image_side = 600
- img_channel_mean = (103.939, 116.779, 123.68)
- img_scaling_factor = 1.0
- num_rois = 32
- rpn_stride = 16
- std_scaling = 4.0
- classifier_regr_std = (8.0, 8.0, 4.0, 4.0)
- rpn_min_overlap = 0.3
- rpn_max_overlap = 0.7
- classifier_min_overlap = 0.1
- classifier_max_overlap = 0.5
- optimizer = Adam

El mejor modelo al observar a tabla de arriba (en función de la métrica mAP) es el modelo con id igual a 3, **¿qué tan bueno fue para cada una de las clases?**

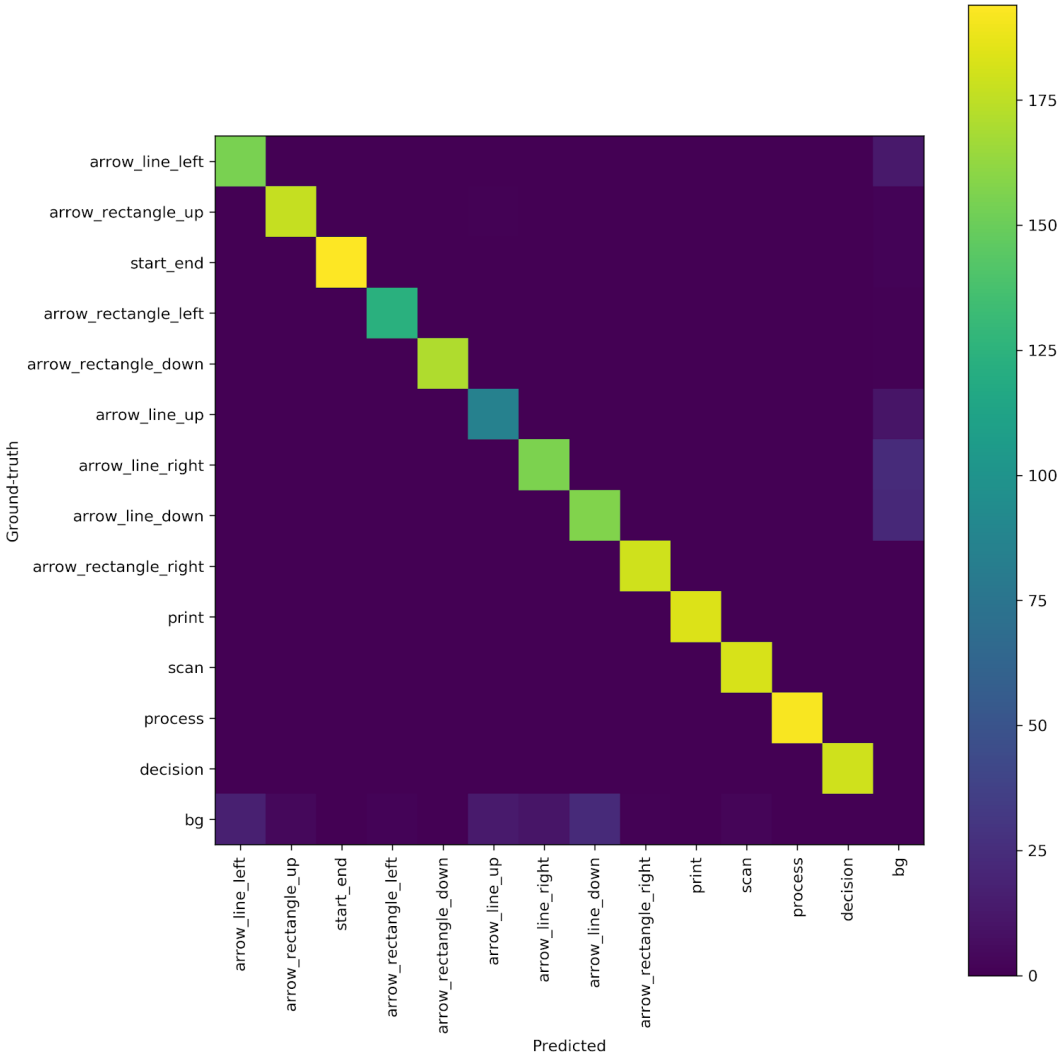
> AP por clase

Clase	**AP (Average Precision)**
Arrow line up	0.9301
Arrow line down	0.9294
Arrow line left	0.9787
Arrow line right	0.9765
Arrow rectangle up	0.9890
Arrow rectangle down	0.9990
Arrow rectangle left	1.0
Arrow rectangle right	1.0

Start end	0.9999
Process	1.0
Decision	0.9999
Print	0.9999
Scan	0.997

mAP = 0.9876

> Matriz de confusión:



> Reporte de clasificación:

	category	**precision 0.5 IoU**	**recall 0.5 IoU**
0	arrow_line_left	0.901162790697674	0.922619047619047
1	arrow_rectangle_up	0.977900552486188	0.983333333333333
2	start_end	1	0.989795918367347
3	arrow_rectangle_left	0.984	0.991935483870967
4	arrow_rectangle_down	1	0.994186046511628
5	arrow_line_up	0.85	0.885416666666667

6	arrow_line_right	0.934131736526946	0.866666666666667
7	arrow_line_down	0.872222222222222	0.877094972067039
8	arrow_rectangle_right	0.994475138121547	1
9	print	1	1
10	scan	0.983783783783784	1
11	process	1	1
12	decision	1	1
13	bg	0	0

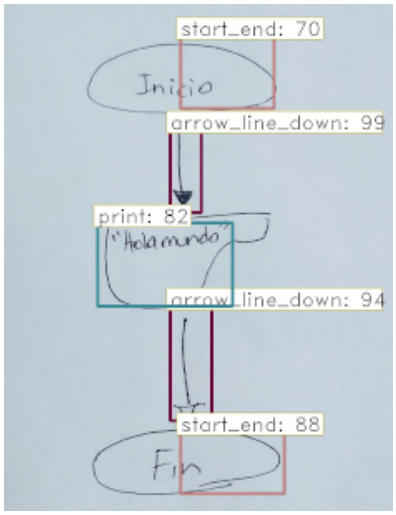
Descripción del modelo de detección de texto

Keras-ocr proporciona modelos de OCR listos para usar y una línea de capacitación integral para construir nuevos modelos de OCR (hacer transferencia de aprendizaje).

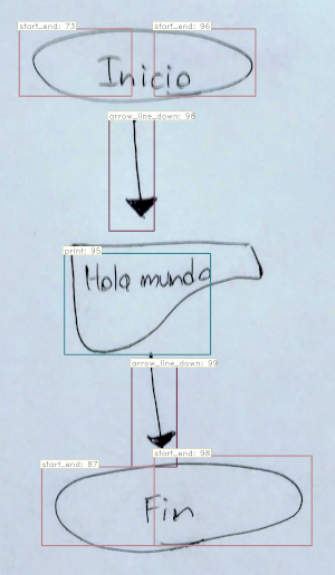
De su página de documentación (https://pypi.org/project/keras-ocr/) se rescata lo siguiente: “Esta es una versión ligeramente pulida y empaquetada de la implementación Keras CRNN (https://github.com/kurapan/CRNN) y el modelo de detección de texto CRAFT publicado (https://github.com/clovaai/CRAFT-pytorch). Proporciona una API de alto nivel para entrenar una detección de texto y una canalización de OCR.”

Pruebas de detección de figuras y conectores

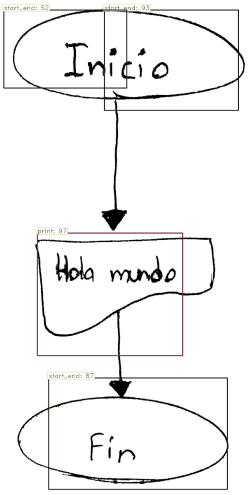
Haciendo pruebas en varios diagramas, el modelo con mejores resultados después de un análisis visual ha sido el modelo con id = 2 (donde se emplea aumento de datos). Por la tanto, para la presentación de estas pruebas de detección dicho modelo fue empleado. Entonces, sobre el modelo de figuras y conectores aquí se muestran algunos resultados, para apreciar de forma visual algunos de los principales problemas.



De las pruebas realizadas en los 3 tipos de diagramas (“Hola mundo”, “Fibonacci” y “Factorial”), el diagrama "Hola mundo" fue el único en el que se pudo hacer un reconocimiento completo de figuras y conectores.



Un problema con esta detección es que 2 detecciones se realizan en el mismo símbolo “start_end”, cubriendo solo hasta la mitad, por otro lado, el cuadro delimitador del “print” no lo encierra por completo, en otras pruebas con el "Hola mundo" no todos los símbolos son reconocidos**.**



Es posible observar que cuando se reconoce el diagrama con la imagen preprocesada hay una mayor cantidad de símbolos no reconocidos, en este caso las flechas rectas hacia abajo.

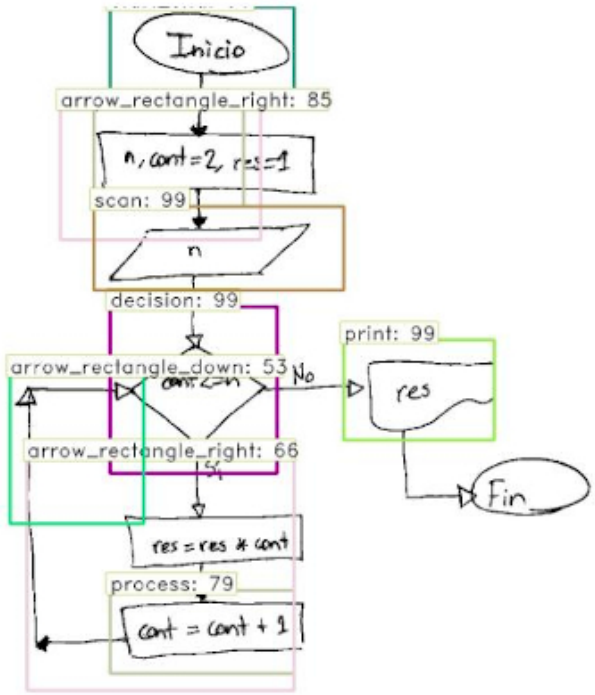


Imagen preprocesada del diagrama de flujo para el cálculo factorial, no todos los símbolos se reconocen y en 2 casos hay confusión con las flechas rectangulares.

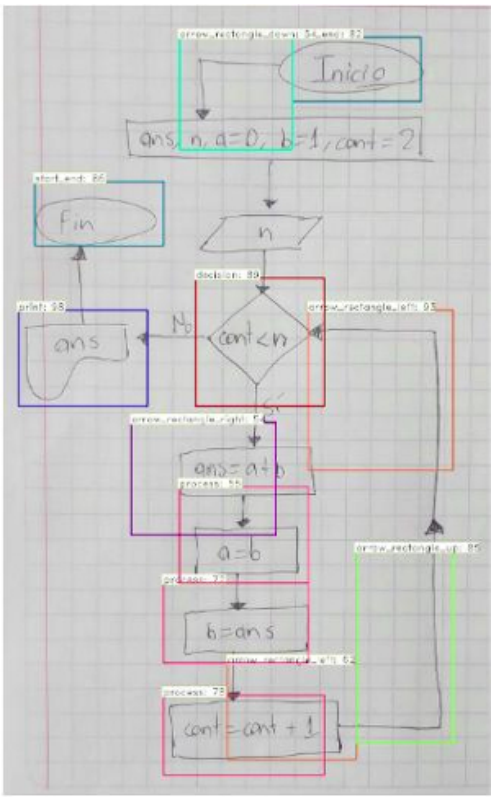
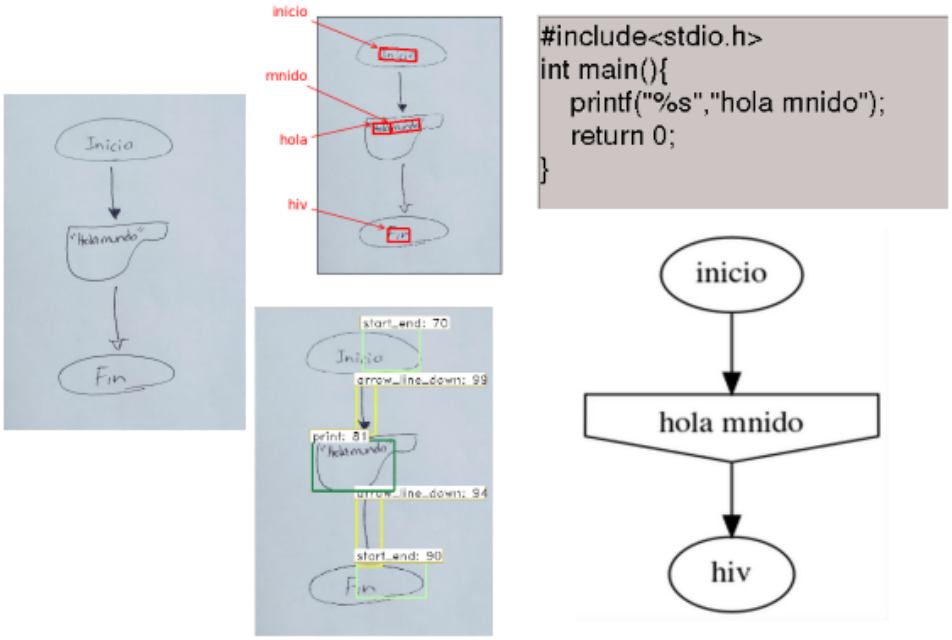


Diagrama de flujo para el n-ésimo término de la secuencia de Fibonacci, existen problemas similares en aquellos mencionados para el anterior diagrama de cálculo de factorial.

Una prueba completa, cerca de ser exitosa se ilustra en la siguiente imagen:



La localización del texto y de las figuras y conectores parece suficiente, pero la clasificación del texto y resto de símbolos no.

Ventajas

- No se requiere el uso de un algoritmo de segmentación, ni un flujo de preprocesamiento de la imagen, siendo evidente que es más eficiente sin ellos.
- No requiere de configuraciones extras para usar el modelo de detección de texto, solo instalar el paquete Keras OCR.
- El modelo de detección de figuras y conectores es capaz de reconocer independientemente de los 3 tipos de fondos diferentes encontrados en el dataset.

Desventajas

- El modelo de texto no es capaz de reconocer caracteres como operadores matemáticos o comillas, solo alfanuméricos, por lo que vuelve incapaz al sistema de reconocer diagramas con operaciones matemáticas, lo cual es una limitación grave.
- En el contexto de detección de diagramas de flujo, el modelo de figuras y texto tiene una importante limitación en la detección de los símbolos respecto a cuando se reconocen en las fotos de las plantillas empleadas en el entrenamiento.
- El modelo de reconocimiento de figuras en los diagramas los bounding boxes no son lo suficientemente precisos en flechas rectangulares, figuras de proceso relativamente alargados y símbolos de inicio/fin, ya que algunas veces solo se abarca la mitad del objeto detectado.
- Existe confusión mayormente entre flechas rectangulares y procesos ya que flechas rectas son perpendiculares a la figura de proceso del que se origina.
- El símbolo de impresión en la reconstrucción del diagrama digital es diferente del empleado en el conjunto definido de símbolos a utilizar.

- El diagrama reconstruido no será exactamente igual al diagrama de flujo de entrada.

Conclusiones

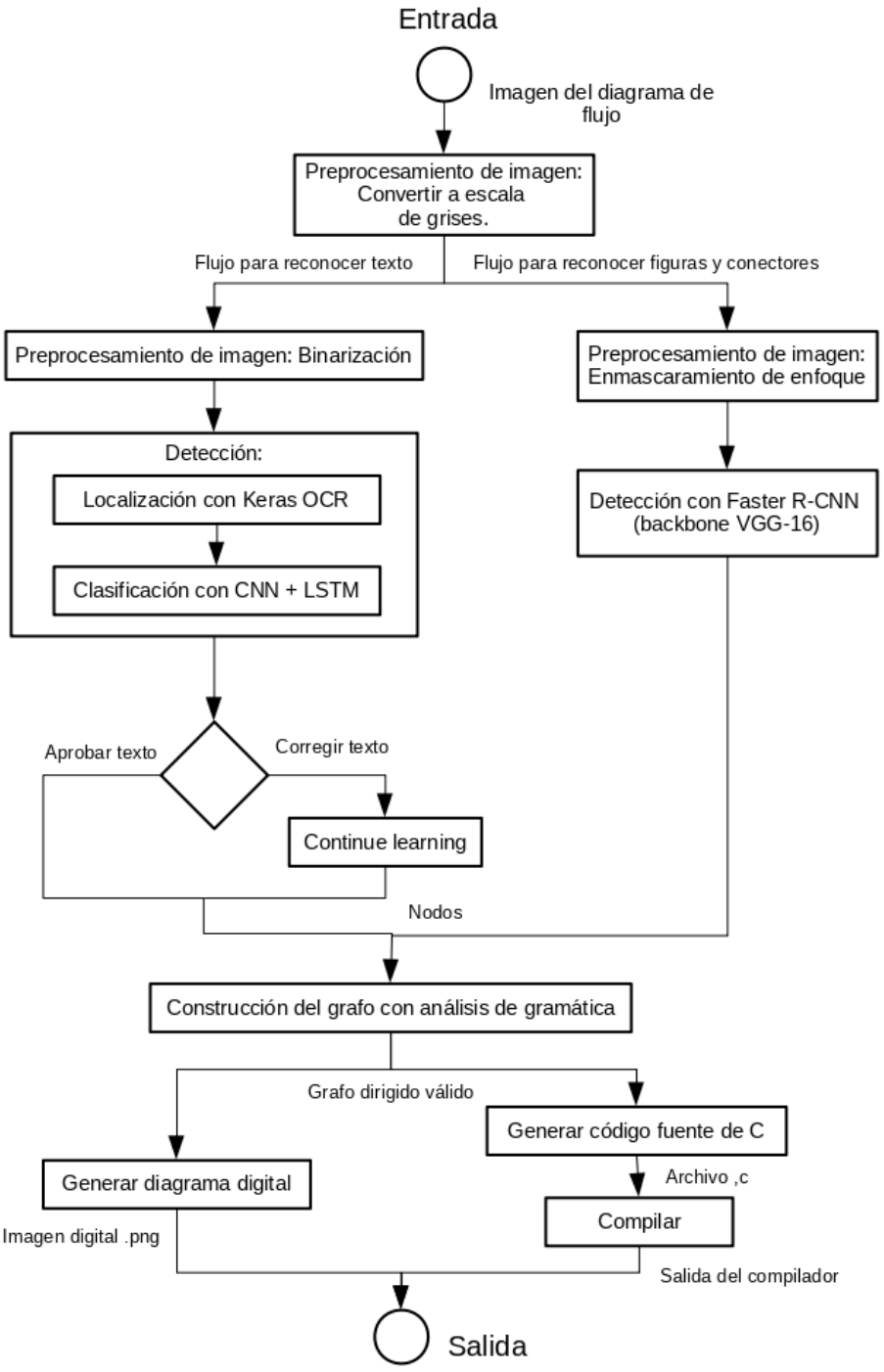
- De todas las pruebas realizadas, el "Hola mundo" ha sido parcialmente el más cercano a ser reconocido de forma exitosa**.**
- Es deseable una mejora en la precisión de los delimitadores de objetos en el modelo de figuras y conectores.
- Tener la imagen preprocesada (con umbralización) mejora muy ligeramente el reconocimiento de objetos solo en pocos casos, y en la mayoría lo empeora. Por esta razón, se decidió eliminarlo del pipeline.
- La mayoría de los reconocimientos confusos son con flechas rectangulares, ya que figuras como los procesos tienen líneas perpendiculares entre sí. Esto representa una ambigüedad que podría ser resuelta al normalizar solo el uso de flechas rectas.
- Es necesario mejorar la detección tanto en el modelo de texto (ya sea entrenando con un dataset existente) así como en el de las figuras y las flechas, ya que en el contexto de los diagramas hay una cantidad considerable de símbolos que no se reconocen, entonces se piensa que en el entrenamiento deben existir instancias donde los símbolos se encuentren en el contexto en el que serán reconocidos, en este caso en diagramas de flujo.
- La variación de parámetros que dieron mejores resultados en métricas y análisis visual son el modificar los anchors, agregando una dimensión de 64 y quitando la de 512 y el aumento de datos, respectivamente.

O.2 Evaluación del sistema para versión 2 (final)

Enlace del repositorio donde se encuentra la implementación:
https://github.com/dbetm/handwritten-flowchart-with-cnn/

La implementación completa para la segunda versión (final) del sistema, se basa en lo diseñado e implementado para versión 1 del sistema (**https://github.com/dbetm/handwritten-flowchart-with-cnn/tree/v1.0**). El pipeline que se muestra abajo se emplea en esta segunda versión para resolver el problema de reconocimiento de diagramas de flujo hechos a mano usando redes neuronales convolucionales.

Hay dos diferencias importantes respecto a la primer versión, la primera es que se incluye el preprocesamiento de la imagen, por el lado del reconocimiento de figuras y conectores una vez que los algoritmos probados sobre las imágenes de entrada hicieron una mejora considerable evidente en pruebas con un set de 56 diagramas de flujo y esto se logró aplicando enmascaramiento de enfoque que mejora la nitidez de las imágenes, haciendo capaz al modelo de extraer más características y por otra parte la necesidad de umbralización para el texto; la segunda es la forma de resolver la detección del texto, esto es, usando el modelo de Keras OCR solo para la localización y entrenando un modelo propio, con redes neuronales convolucionales y 1-D BLSTM, con un dataset existente para la clasificación de los caracteres, aunque al final fue necesario implementar una técnica llamada “Continual learning” con el fin de que el modelo siga aprendiendo y se adapte al estilo de escritura de quien lo utiliza, y así en algún momento sea capaz de obtener resultados aceptables (e incluso perfectos) en la mayoría de los casos.



Pipeline empleado en la versión final del sistema.

Para la detección de figuras y conectores se especificó y recopiló un dataset (https://bit.ly/31lX1sV) consistiendo en fotos de plantillas (sobre 3 tipos de fondos diferentes) de 13 clases diferentes de símbolos

("arrow_line_left", "arrow_rectangle_up", "start_end", "arrow_rectangle_left", "arrow_rectangle_down", "arrow_line_up", "arrow_line_right", "arrow_line_down", "arrow_rectangle_right", "print", "scan", "process", "decision"). Para esta versión del sistema se reestructuró a una nueva versión, agregando más imágenes y quitando las clases de flechas rectangulares ("arrow_line_down", "arrow_rectangle_right", "arrow_rectangle_left", "arrow_rectangle_up"), incluyendo de diagramas de flujo, el mismo se puede encontrar aquí: https://n9.cl/6kgs .

Se volvió a utilizar el modelo de detección de objetos Faster R-CNN con backbone VGG-16, se describe brevemente en el apartado de abajo.

Por otra parte y como ya se mencionó antes, para la localización del texto se utilizó un modelo existente usado en un paquete llamado Keras OCR (https://pypi.org/project/keras-ocr/) que también se describe brevemente abajo.

Tanto el modelo de clasificación de texto, así como el de figuras fue implementado usando Keras con TensorFlow como backend.

Para la generación del diagrama de flujo en imagen digital se ha empleado un paquete llamado "Graphviz for Python" (https://pypi.org/project/graphviz/).

Los requerimientos para crear un entorno en Conda, y hacer pruebas con el sistema, se pueden encontrar en el siguiente enlace: https://github.com/dbetm/handwritten-flowchart-with-cnn/blob/master/requirements.txt .

Finalmente, el diagrama de Conway que determina la gramática para construir el grafo es el siguiente:

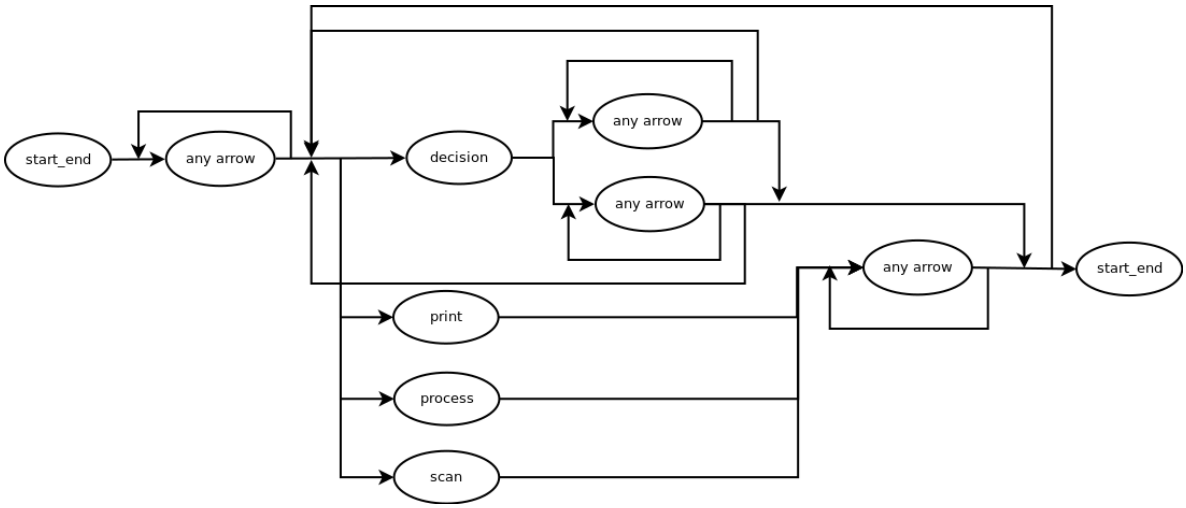
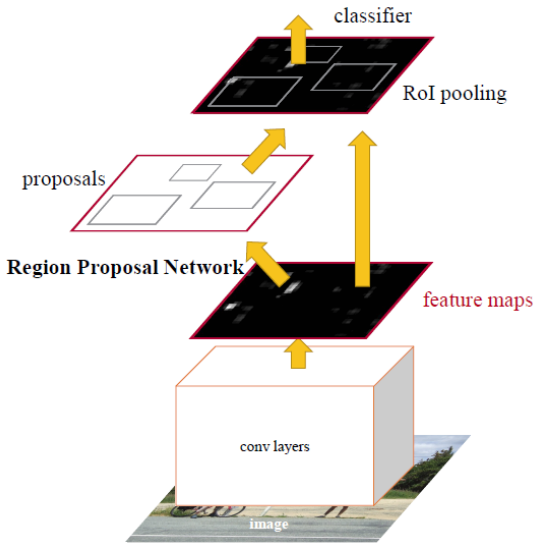


Diagrama de Conway que determina la gramática

Observar que solo se cuenta con nodos terminales.

Modelo de detección de figuras y conectores

Este modelo de detección de objetos fue presentado en https://arxiv.org/pdf/1506.01497.pdf y presenta una arquitectura para la extracción de características (backbone) que se llama VGG-16 que fue presentado en https://arxiv.org/pdf/1409.1556.pdf, su estructura es la siguiente:



Se puede apreciar que consta de 2 fases para lograr la detección de objetos, y se aprovecha dos veces el mapa de características obtenido por el backbone.

Para el entrenamiento se tomó en cuenta la conclusión sobre la versión 1 del sistema de que los mejores resultados se obtenían cuando se modificaban los anchors y se hacía aumento de datos, aunado a que se hizo mejora en el tipo de aumento de datos ajuste de contraste, corrección gamma, ligera rotación e inclinación).

Para el entrenamiento del mejor modelo (en la evaluación del modelo de figuras aquel con id = 9) fue necesario un entrenamiento:

- Con 700 épocas en un tiempo aproximado de 12 horas.
- De forma remota (en la nube):
    - Software: Colab de Google.
    - Hardware: 12.72 GB RAM, 68.4 GB almacenamiento secundario, Intel(R) Xeon(R) CPU @ 2.20GHz, GPU Tesla P100.
- Utilizando los kernels, para inicializar los pesos, de un modelo pre-entrenado que se encuentra en: https://github.com/fchollet/deep-learning-models/releases .

Los hiperparámetros para el modelo son:

- **use_data_augmentation = True**
- epoch_lenght = 32
- learning_rate = 0.00001
- **anchor_box_scales = (64, 128, 256)**

- **anchor_box_ratios = [**

  [1, 1],

  [1./math.sqrt(2), 2./math.sqrt(2)],

  [2./math.sqrt(2), 1./math.sqrt(2)],

  [1./math.sqrt(2), 4./math.sqrt(2)],

  [4./math.sqrt(2), 1./math.sqrt(2)]]
- min_image_side = 600
- img_channel_mean = (103.939, 116.779, 123.68)
- img_scaling_factor = 1.0
- num_rois = 32
- rpn_stride = 16
- std_scaling = 4.0
- classifier_regr_std = (8.0, 8.0, 4.0, 4.0)
- rpn_min_overlap = 0.3
- rpn_max_overlap = 0.7
- classifier_min_overlap = 0.1
- classifier_max_overlap = 0.5
- optimizer = Adam

En negrita se resaltan aquellos que son diferentes respecto a los hiperparámetros por defecto empleados en la versión 1 del sistema.

La pérdida mínima alcanzada para este modelo entrenado fue:

+ Pérdida total: 0.341573134099416
    + loss_cls_rpn: 0.0072
    + loss_regr_rpn: 0.033
    + loss_cls_det: 0.210
    + loss_regr_det: 0.090

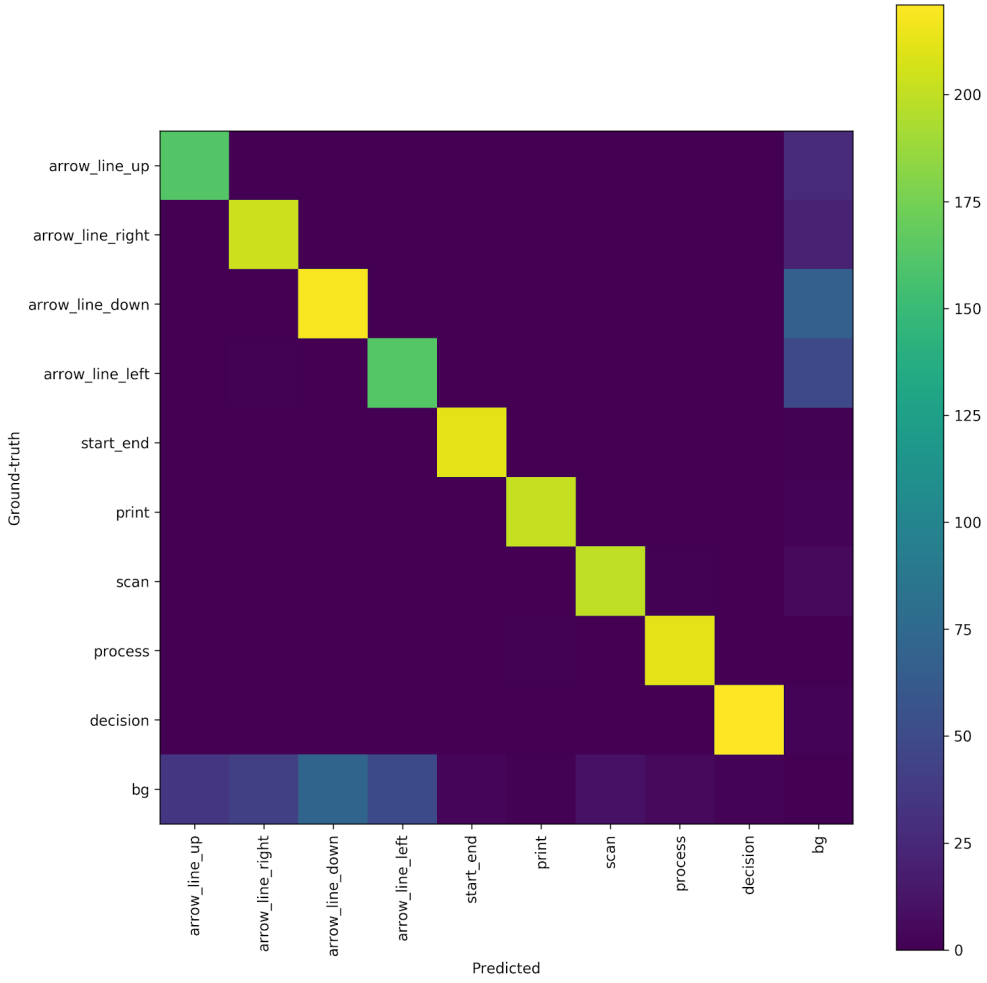
Métricas del modelo obtenidas sobre el dataset de validación:

> AP (Average Precision) por clase:

- ❑ scan AP: 0.9940048180531599
- ❑ arrow_line_left AP: 0.8927451831815845
- ❑ arrow_line_right AP: 0.9630576008754724
- ❑ start_end AP: 0.9999134592501244
- ❑ decision AP: 0.9996940497071582
- ❑ print AP: 0.9998320137081854
- ❑ process AP: 0.9999134592501244
- ❑ arrow_line_down AP: 0.8478354022766538
- ❑ arrow_line_up AP: 0.8914715994457478

Las flechas hacia arriba, abajo e izquierda tienen los valores más bajos, respecto a las casi perfectos resultados del resto de símbolos.

> Matriz de confusión:



Aquí se puede notar que la mayoría de las confusiones existentes ya sea que no se hayan detectado o se hayan detectado objetos en el fondo, corresponden a las flechas.

> Reporte de clasificación:

	Clase	**precision 0.5 IoU**	**recall 0.5 IoU**
0	arrow_line_up	0.822335025380711	0.857142857142857
1	arrow_line_right	0.825910931174089	0.902654867256637

2	arrow_line_down	0.755172413793103	0.763066202090592
3	arrow_line_left	0.765258215962441	0.765258215962441
4	start_end	0.981481481481482	0.995305164319249
5	print	0.990196078431373	0.990196078431373
6	scan	0.947867298578199	0.966183574879227
7	process	0.968036529680365	0.995305164319249
8	decision	0.991031390134529	0.991031390134529
9	bg	0	0

Notar que se ha empleado un umbral de 0.5 para el IoU. El símbolo de impresión y decisión tienen el valor más alto en cuestión de precisión, mientras que la flecha abajo e izquierda los más bajos.

Modelos de detección de texto

Keras-ocr proporciona modelos de OCR listos para usar y una línea de capacitación integral para construir nuevos modelos de OCR (hacer transferencia de aprendizaje).

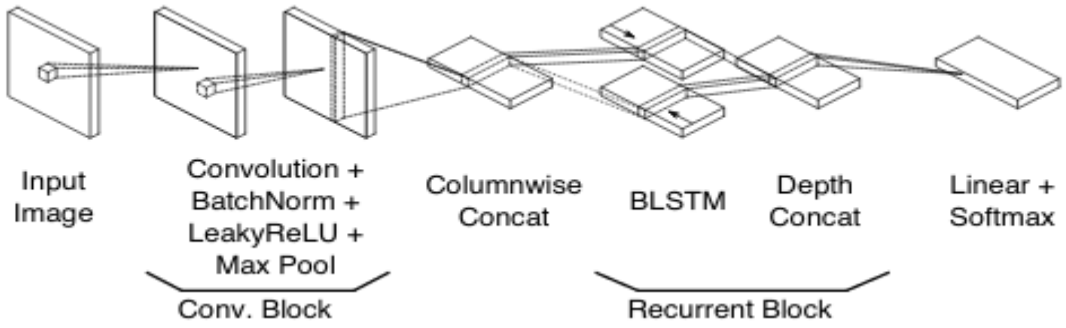
Keras OCR es empleado dentro de la tarea de detección de texto para localización de los caracteres. De su página de documentación (https://pypi.org/project/keras-ocr/) se rescata lo siguiente: "Esta es una versión ligeramente pulida y empaquetada de la implementación Keras CRNN (https://github.com/kurapan/CRNN) y el modelo de detección de texto CRAFT publicado (https://github.com/clovaai/CRAFT-pytorch). Proporciona una API de alto nivel para entrenar una detección de texto y una canalización de OCR."

Mientras que para la clasificación y así poder completar la tarea de detección se utilizó una implementación en Keras del modelo publicado puigcerver (http://www.jpuigcerver.net/pubs/jpuigcerver_icdar2017.pdf). La arquitectura del modelo está constituido por bloques de convolución donde cada bloque contiene una capa convolucional bidimensional, para reducir el sobreajuste se aplica "Dropout" (técnica de regularización para evitar sobreajuste) en la entrada de algunas capas convolucionales después de cada capa convolucional se aplica la normalización por lotes, además de unidades lineales rectificadoras con fugas (LeakyReLU) como función de activación en los bloques convolucionales. Por último, una capa de agrupación máxima (Maxpool) para reducir la dimensionalidad de las imágenes de entrada. Después de los bloques de convolución se aplican los bloques recurrentes los cuales están formados por capas bidireccionales 1D-LSTM que procesan la imagen de entrada en columna en orden de izquierda a derecha y de derecha a izquierda. En la salida de los bloques BLSTM se aplica nuevamente "Dropout".

Finalmente, después de los bloques recurrentes cada columna debe asignarse a una etiqueta de salida, para eso, la profundidad es transformada a un tamaño igual al número de caracteres + 1 para el símbolo de espacio en blanco.

Todos los parámetros de la red neuronal son entrenadas para minimizar la pérdida CTC (Connectionist Temporal Classification), se usó el algoritmo RmsProp para incrementar la actualización de parámetros del modelo utilizando los gradientes de la pérdida de CTC.

A continuación, se muestra visualmente la arquitectura antes descrita:



Recuperada desde http://www.jpuigcerver.net/pubs/jpuigcerver_icdar2017.pdf .

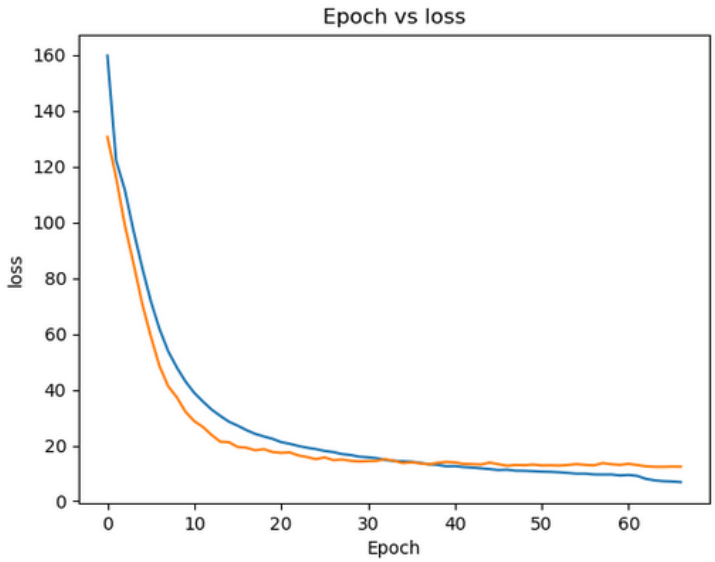
La métricas utilizadas para evaluar el modelo de clasificación fueron CER (Character Error Rate) y WER (Word Error Rate). La siguiente tabla muestra las métricas mencionadas, estas métricas fueron obtenidas en http://www.jpuigcerver.net/pubs/jpuigcerver_icdar2017.pdf .

CER	Validation	**5.1** [4.6–5.7]
	Test	**8.2** [7.6–8.9]
WER	Validation	**17.9** [16.3–19.7]
	Test	**25.4** [23.9–27.0]
Avg. Runtime (min.)		**3.8** [3.8–3.8]
# of parameters (Mi.)		9.3

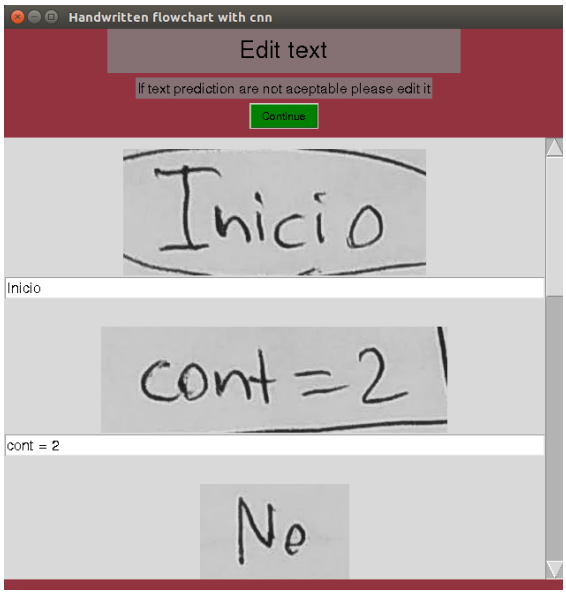
- CER indica cuánto % de caracteres no se clasificaron correctamente.
- WER indica qué tan buena es la reproducción de las palabras en el texto.

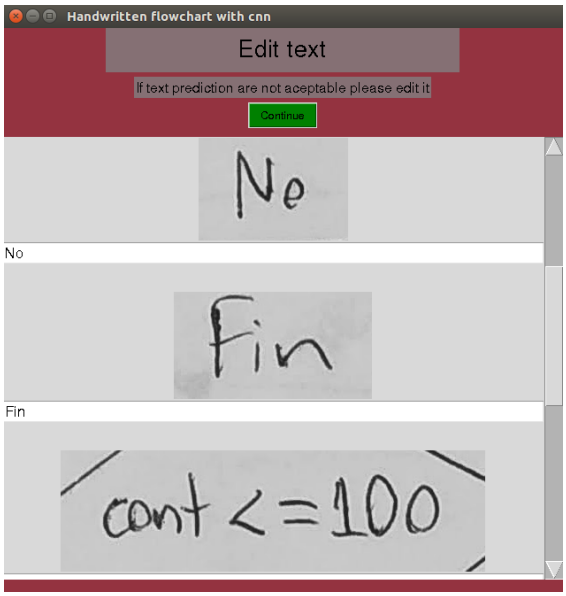
Al entrenar el modelo localmente con una GPU GTX 1660 con un total de 66 épocas, usando el callback de Early Stop, con una pérdida en el set de validación de 12.3811.

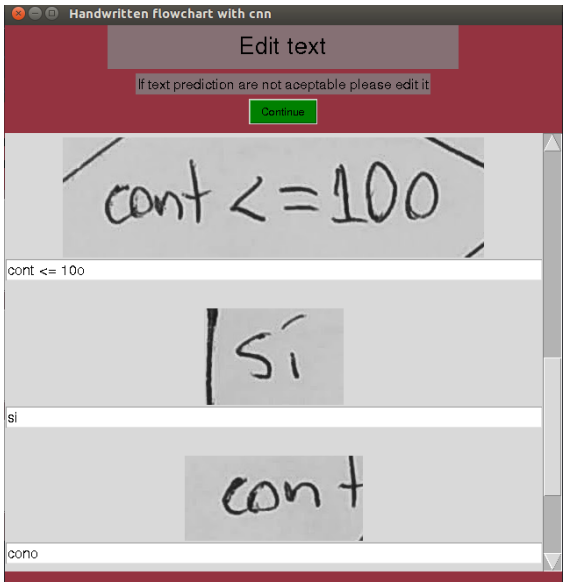
Uno de los problemas presentados durante el entrenamiento fue el sobreajuste (overfitting) de un modelo. A continuación, una gráfica que muestra el descenso de la pérdida CTC, la línea azul representa la pérdida en entrenamiento, y la naranja representa la pérdida sobre el dataset de validación. Se puede observar que conforme avanzaba el entrenamiento la pérdida en la validación no desciende y en la pérdida del entrenamiento sí, por lo que se detuvo el entrenamiento ya que era un comportamiento de sobreajuste.

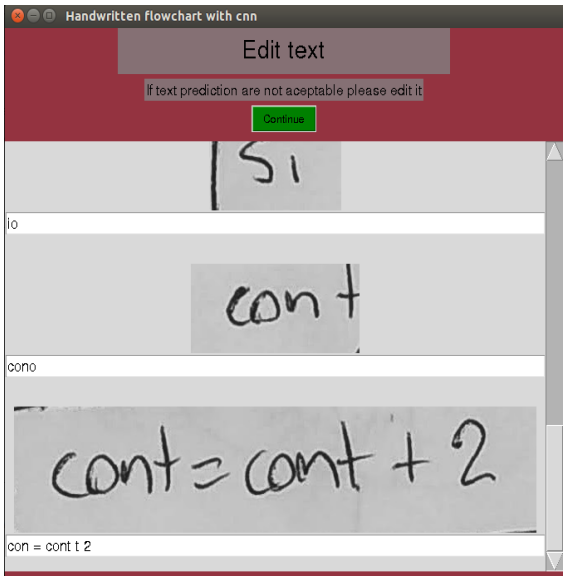


Dentro del pipeline, como ya se ha mencionado, existe un módulo para aprobar el texto, en las siguientes capturas se muestran los resultados de la detección de texto justo antes de que el usuario lo edite, cabe resaltar que el modelo de texto se le han aplicado varios entrenamientos "continual learning", sin embargo, para mayor precisión deberán hacerse más de tales entrenamientos.




Handwritten flowchart with cnn
Edit text
If text prediction are not aceptable please edit it
Continue
No
No
Fin
Fin
cont <=100


Handwritten flowchart with cnn
Edit text
If text prediction are not aceptable please edit it
Continue
cont <=100
cont <= 10o
si
si
cont
cono



En la mayoría de los casos la detección es aceptable. Sin embargo, es necesario corregir lo necesario con el fin de construir el código fuente y diagrama de flujo en formato digital.

Pruebas de detección de figuras y conectores

A continuación se muestran resultados de detecciones sobre diversos diagramas de flujo hechos a manos que representan varios algoritmos.

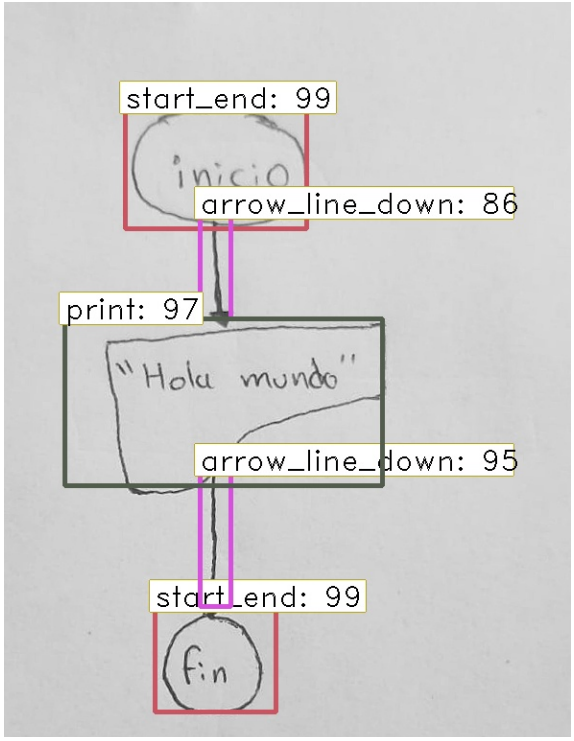


Diagrama de “Hola mundo” sobre fondo de hoja de máquina.

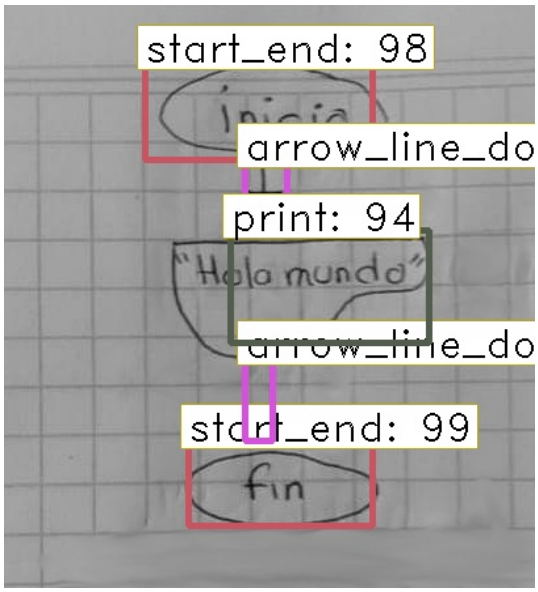


Diagrama de “Hola mundo” sobre fondo cuadriculado.

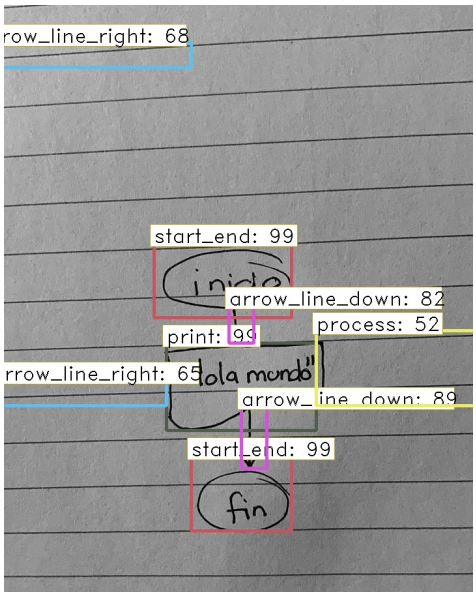


Diagrama de “Hola mundo” sobre fondo de raya, el detalle aquí es la detección de figuras en el fondo, esto debido a que es un fondo de una hoja con rayas suficientemente visibles para provocar confusión.

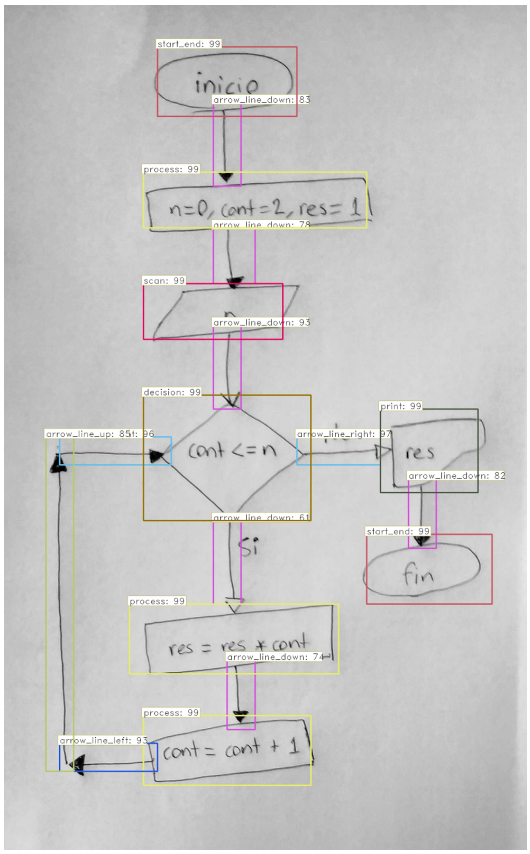


Diagrama de “Cálculo del factorial” sobre un fondo de máquina.

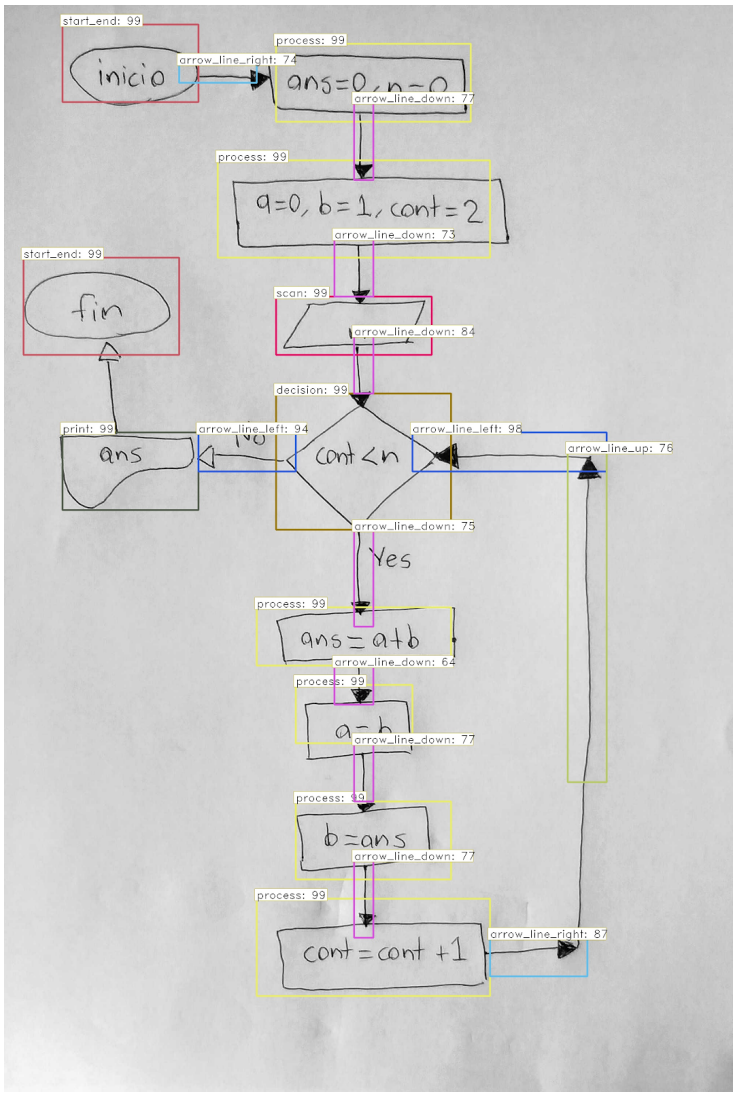


Diagrama de “Cálculo del n-ésimo término de la sucesión de Fibonacci” en fondo de máquina, se puede apreciar que no se ha detectado una única flecha hacia arriba y que el cuadro delimitador (bounding box) no abarca completamente la flecha hacia arriba más grande del diagrama.

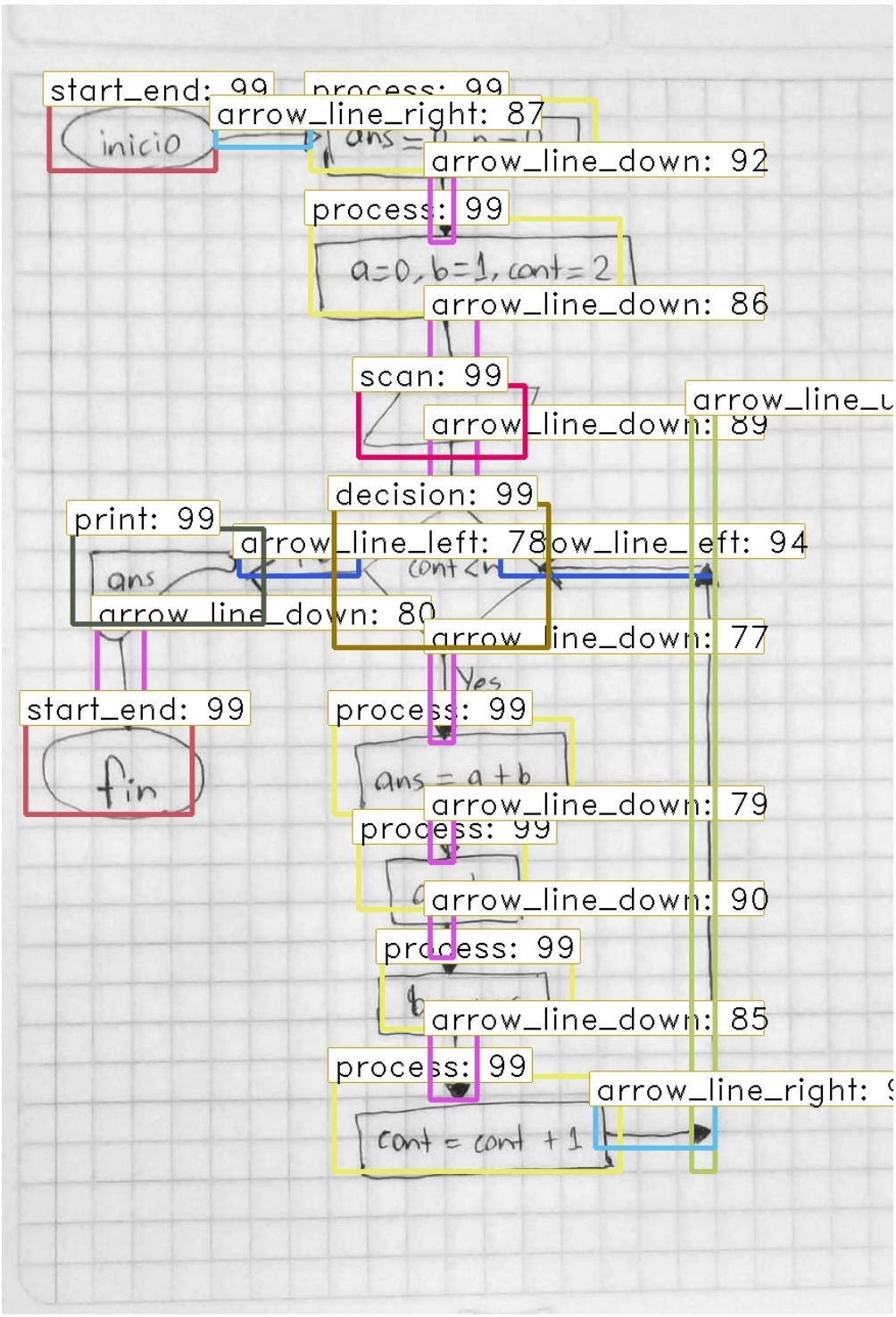


Diagrama de “Cálculo del n-ésimo término de la sucesión de Fibonacci” en fondo de cuadrícula.

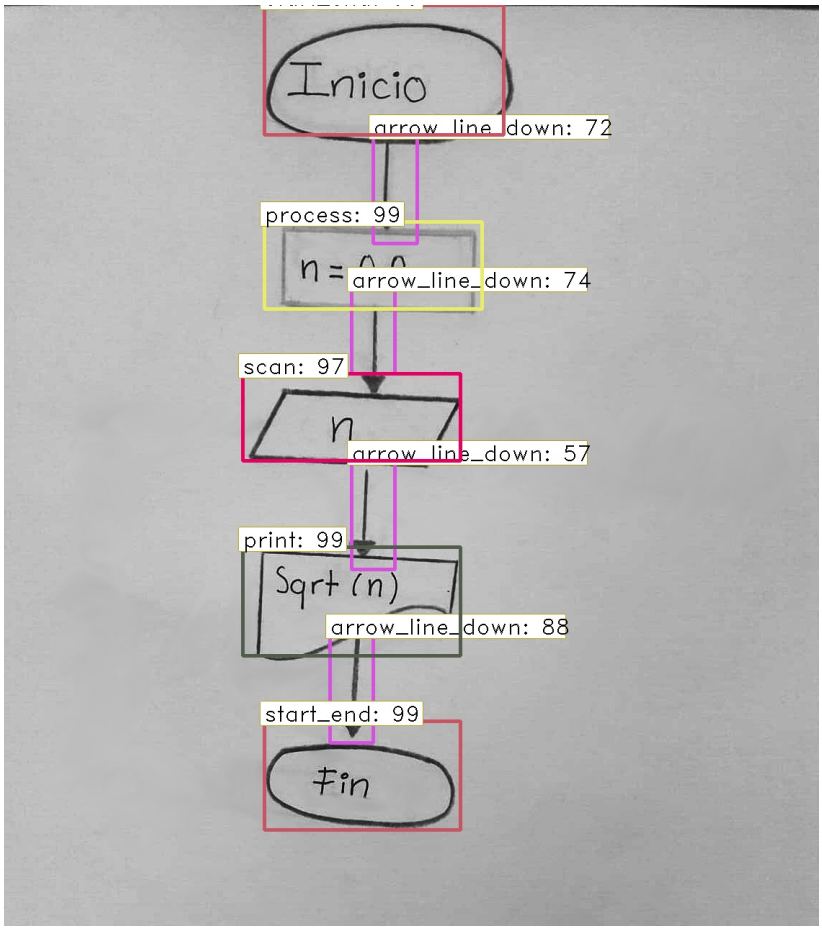
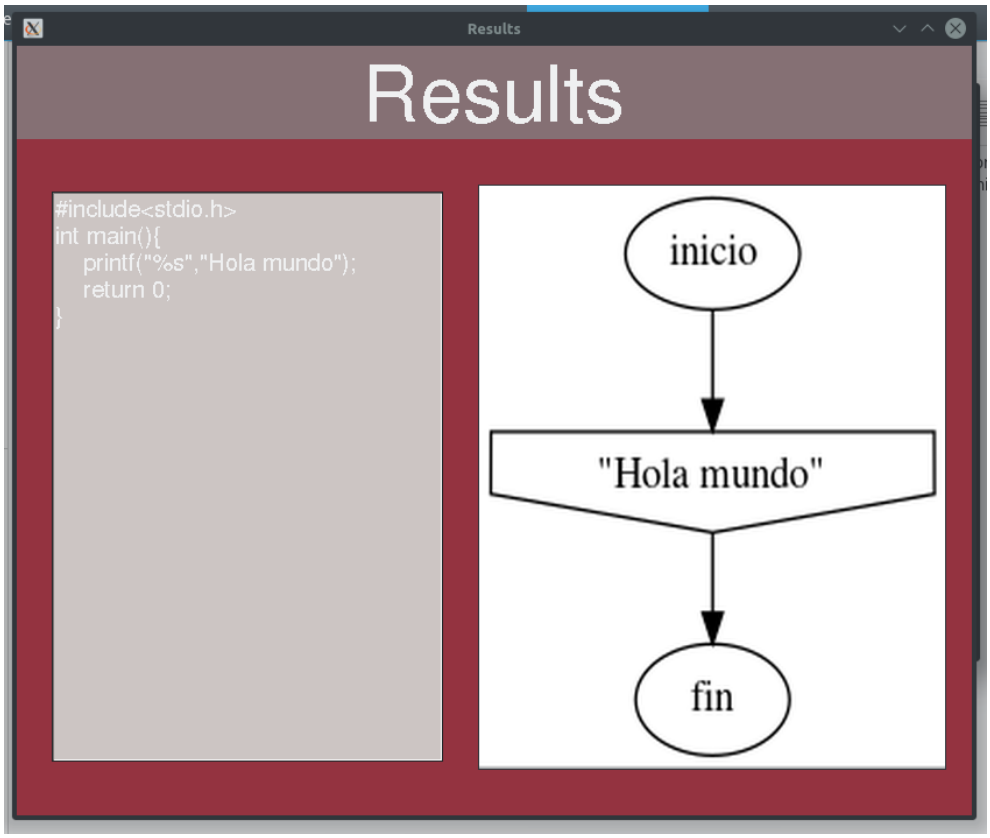


Diagrama del “Cálculo de la raíz cuadrada” sobre fondo de máquina.

Resultados de pruebas completas:

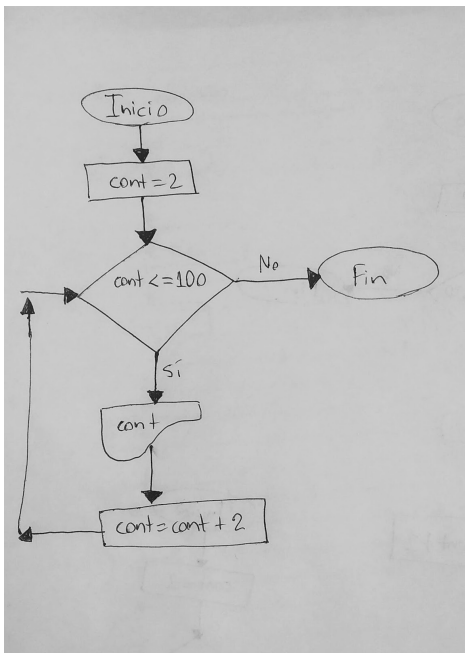
> Hola mundo

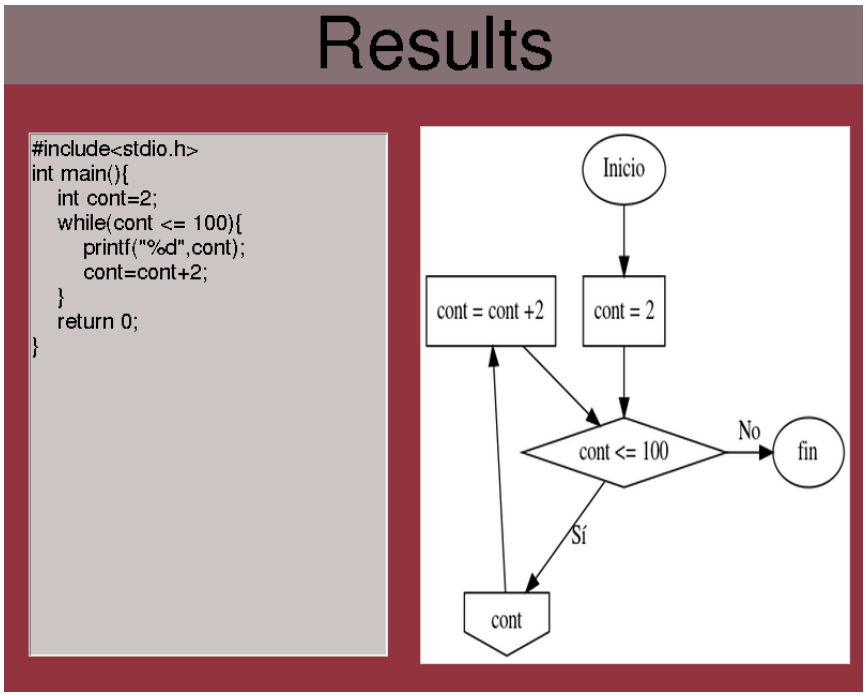
Diagrama de flujo “Hola mundo”.



Resultados al reconocer un diagrama de flujo “Hola mundo”.

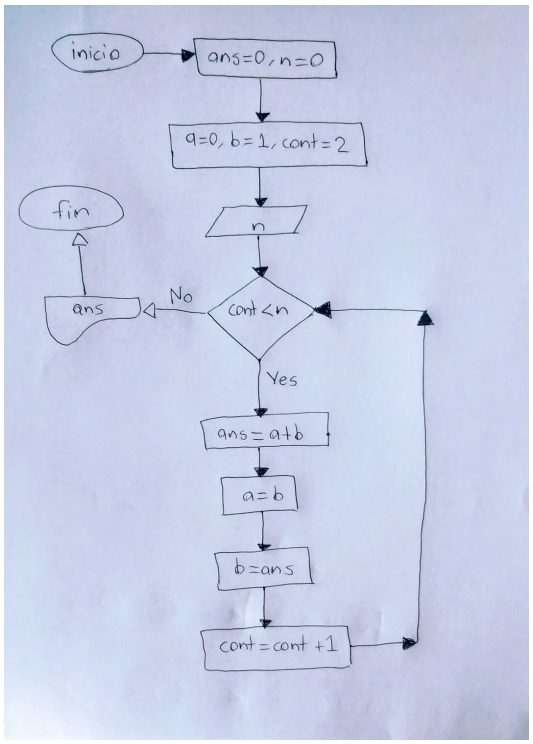
> Imprimir números naturales pares hasta el 100

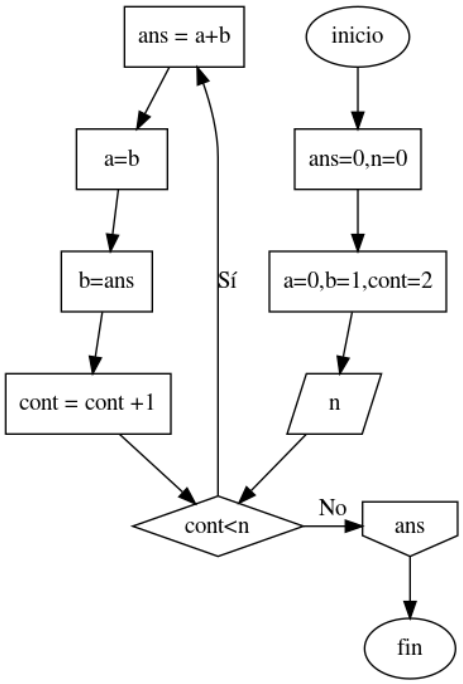




Resultados, código en C y diagrama reconstruido luego de reconocer los componentes del diagramas de flujo.

> Cálculo del n-ésimo elemento de la sucesión de Fibonacci






```c
#include<stdio.h>
int main(){
    int ans=0,n=0;
    int a=0,b=1,cont=2;
    scanf("%d",&n);
    while(cont<n){
        ans=a+b;
        a=b;
        b=ans;
        cont=cont+1;
    }
    printf("%d",ans);
    return 0;
}
```

Ventajas

- El modelo de figuras y conectores en el conjunto de evaluación de 56 diagramas de flujo (https://bit.ly/2VlhXwj), que contiene varios algoritmos hechos por varios dibujantes, es capaz de reconocer 42 diagramas totalmente bien.
- El implementar el "continual learning" en el modelo de texto, ayuda a mejorar continuamente el reconocimiento de texto adaptándose a la forma de escribir del usuario que haga suficientes pruebas.
- El diagrama de flujo puede trazarse con colores, ya que gracias al preprocesamiento se convertirá en grises, y posiblemente se pueda reconocer sin problemas.

Desventajas

- Inicialmente el modelo de detección de texto (en concreto el clasificador), será poco preciso clasificando los caracteres, entonces implica uso de entrenamientos "Continual learning" para mejorar.
- El símbolo de impresión en la reconstrucción del diagrama digital es diferente del empleado en el conjunto definido de símbolos a utilizar.
- El diagrama reconstruido no será exactamente igual al diagrama de flujo de entrada, pero sí equivalente.
- En una computadora personal con Ubuntu 18.04 LTS, 8 GB RAM, procesador Intel CORE i3-5005U CPU @ 2.00 GHz, el tiempo que se toma desde que se da click en predecir, hasta ver los resultados es de alrededor 2 minutos, lo cual impide satisfacer uno de los requerimientos de desempeño definido en el SRS, el cual era realizar el reconocimiento completo en menos de 10 segundos.

Conclusiones

- El haber mejorado el dataset para el entrenamiento de figuras y conectores ha sido positivo, además del uso de aumento de datos y modificación de anchors.

- El reconocimiento de diagramas ha mejorado considerablemente respecto a la versión 1, donde solo se reconocía parcialmente correcto el diagrama para imprimir “Hola mundo”.
- El entrenamiento en la nube resulta ser más lento que en local debido a que los archivos se deben leer desde otro servidor de almacenamiento, así que la comunicación en red genera un retraso considerable.
- El preprocesamiento usando enmascaramiento de enfoque para detectar figuras y conectores, genera imágenes más nítidas que permiten extraer una mayor cantidad de características de las imágenes y así detectar una mayor cantidad de objetos.
- En ciertos diagramas donde las rayas o cuadrícula está pintada a un nivel de intensidad parecido a los trazos, se corre el riesgo de detectar objetos en el fondo.

Apéndice P

Como producto esperado de la metodología experimental se generó un informe para dar a conocer resultados al tratar de resolver el problema en cuestión.

P.1 Informe de resultados

Introducción

El presente informe de evaluación se refiere a los resultados del proyecto de trabajo terminal titulado "Reconocimiento de diagramas de flujo trazados a mano usando redes neuronales convolucionales".

El proyecto se desarrolló por David Betancourt Montellano y Onder Francisco Campos García para la evaluación de las materias de Trabajo Terminal I y II del programa académico de ingeniería en sistemas computaciones de la UPIIZ, el proyecto se desarrolló en el periodo de agosto del 2019 a junio del 2020.

El proyecto fue aprobado con el objetivo de diseñar e implementar un canal (pipeline) de reconocimiento usando técnicas de procesamiento de imágenes, redes neuronales convolucionales y análisis de gramática que sea capaz de reconocer completamente un diagrama de flujo trazado a mano.

Para alcanzar esta meta se definieron 5 objetivos específicos:

- Hacer reconocimiento de figuras, texto y conectores plasmados en una imagen digital de un diagrama de flujo trazado a mano.
- Usar redes neuronales convolucionales (CNNs por sus siglas en inglés) para la tarea de detección (localización y clasificación).
- Usar análisis de gramática para verificar la estructura del diagrama de flujo.
- Poder generar código fuente de C a partir del diagrama de flujo.
- Poder digitalizar el diagrama de flujo, esto es, reconstruir los elementos del diagrama de flujo y generar una imagen digital.

Descripción del proyecto

Mediante el uso de técnicas de procesamiento de imágenes, reconocimiento de patrones (redes neuronales convolucionales) y análisis de gramática se busca el

reconocimiento de diagramas de flujo trazados a mano, enfrentando dicho problema bajo el enfoque *off-line* [8] encontrado así en el estado del arte.

Se diseñará e implementará un canal de reconocimiento (pipeline) destacando el uso de redes neuronales convolucionales (CNNs por sus siglas en inglés) para tratar de aumentar la precisión en el reconocimiento de figuras, conectores y texto, siendo éstos los componentes de un diagrama de flujo [14] y con ayuda del análisis de gramática se pretende verificar la estructura del diagrama de flujo con el fin de detectar errores en el mismo. Para poder ver el resultado de la propuesta en acción se desarrollará con el lenguaje de programación Python, mismo que recibirá de entrada una imagen digital del diagrama de flujo trazado a mano, hará el procesamiento de la imagen, el reconocimiento, análisis de gramática y finalmente regresará el resultado en una imagen del diagrama de flujo una vez que fue reconstruido en una imagen digital y su equivalente código fuente en C, esto si es que se ha logrado reconocer de forma global el diagrama de flujo con una estructura correcta, asimismo, es importante mencionar que los programas que se generen de pruebas seguirán el paradigma de programación estructurada y que se usará un conjunto definido de símbolos para la construcción del diagrama de flujo.

Para el desarrolo del proyecto fue necesario del siguiente equipo de computo:

- 2 Laptops con 8 GB RAM, 1 TB HDD, Intel CORE i3-5005U CPU @ 2.00 GHz.
- 1 PC de escritorio, 8 GB RAM, 240 SSD, 1TB HDD, AMD Ryzen 7 2700 CPU @ 3.2 GHz, GeForce GTX 1660 SUPER GPU.
- 1 PC de escritorio, 16 GB RAM,240 SSD 1TB HDD, i7 cpu@ 3.0 GHz, GeForce GTX 1660 GPU.
- Colab de Google. Equipo: GPU Tesla P100, RAM 12.72 GB y espacio en disco de almacenamiento secundario 68.4 GB.

Discusión de los resultados y valoración

Para se cumpla con el objetivo general del proyecto es necesario que cada uno de los objetivos específicos previamente establecidos se cumplan, teniendo así una lista de

objetivos instrumentales (sub-objetivo) que pueden ser evaluados individualmente, de esta forma que se pueden usar índices más específicos al ámbito del subjetivo, teniendo así una evaluación más exacta.

Con la suma de las evaluaciones de los objetivos instrumentales podemos determinar los resultados de una evaluación global del proyecto.

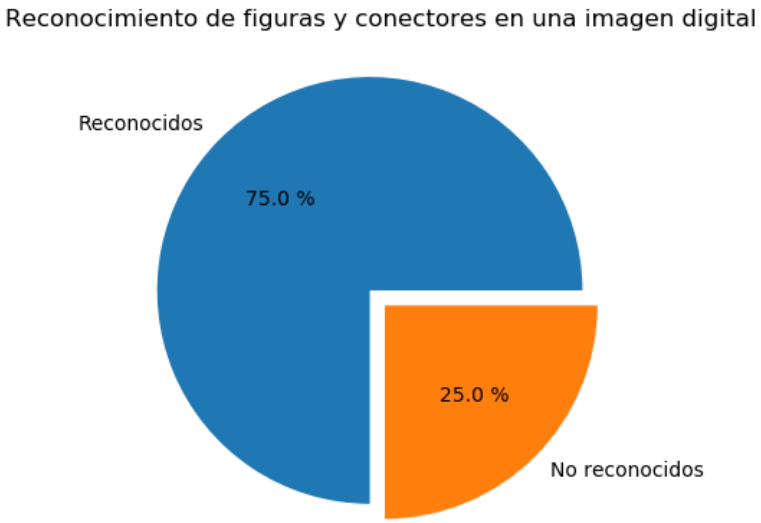
A continuación se describe la forma en la que cada uno de los objetivos instrumentales se evaluaron:

1. **Hacer reconocimiento de figuras, texto y conectores plasmados en una imagen digital de un diagrama de flujo trazado a mano.**

   Para la evaluación de este sub-objetivo es necesario dividirlo en dos partes "hacer reconocimiento de figuras y conectores en una imagen digital de un diagrama de flujo trazado a mano" y "hacer reconocimiento de texto en una imagen digital de un diagrama de flujo trazado a mano".

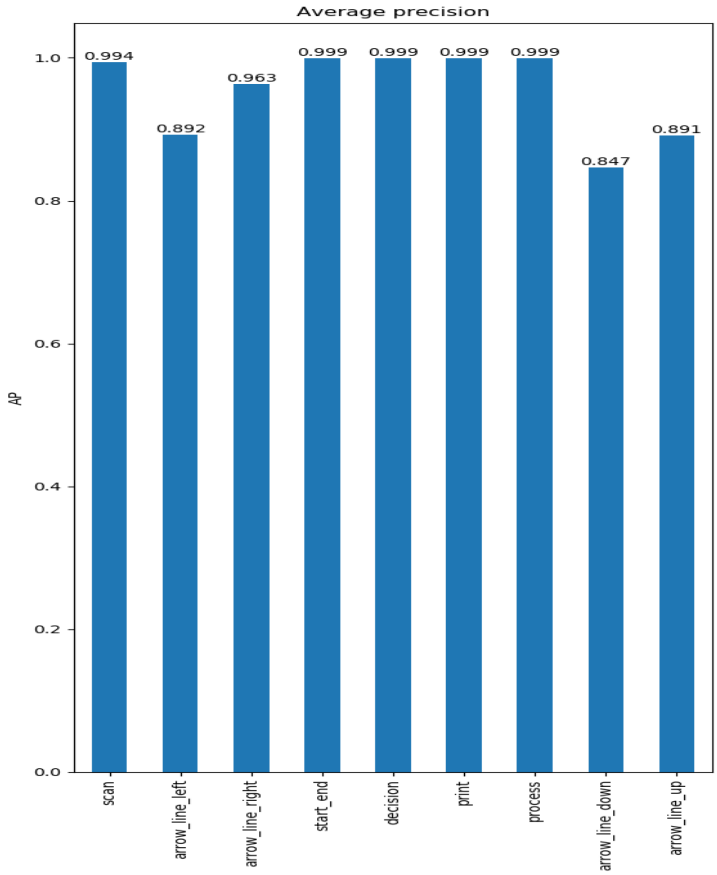
   a. **Hacer reconocimiento de figuras y conectores en una imagen digital de un diagrama de flujo trazado a mano.**

      La siguiente gráfica muestra que el 75% de las imágenes probadas (en un conjunto de prueba de 56 diagramas de flujo), se reconocieron todas las figuras y conectores que existían en ellas.



      La siguiente gráfica muestra la precisión promedio obtenida por cada clase, tomando como positivo verdadero cuando el bounding box del objeto

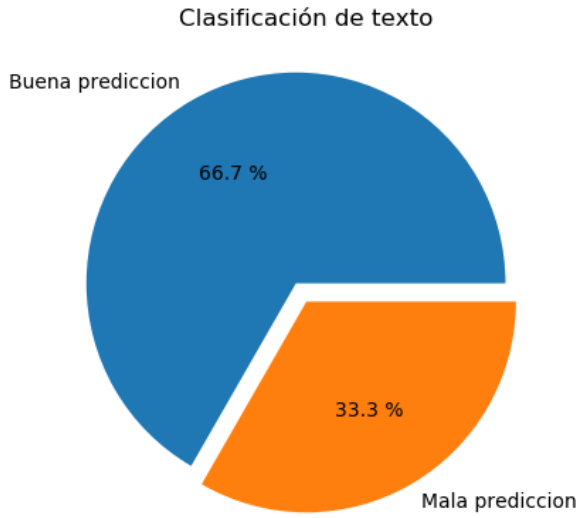
reconocido tiene un Intersection over Union (IoU) del 50% con el objeto que se esperaba, se puede observar que las clases de flechas hacia arriba, abajo e izquierda tienen los valores más bajos, respecto a las casi perfectos resultados del resto de clases.



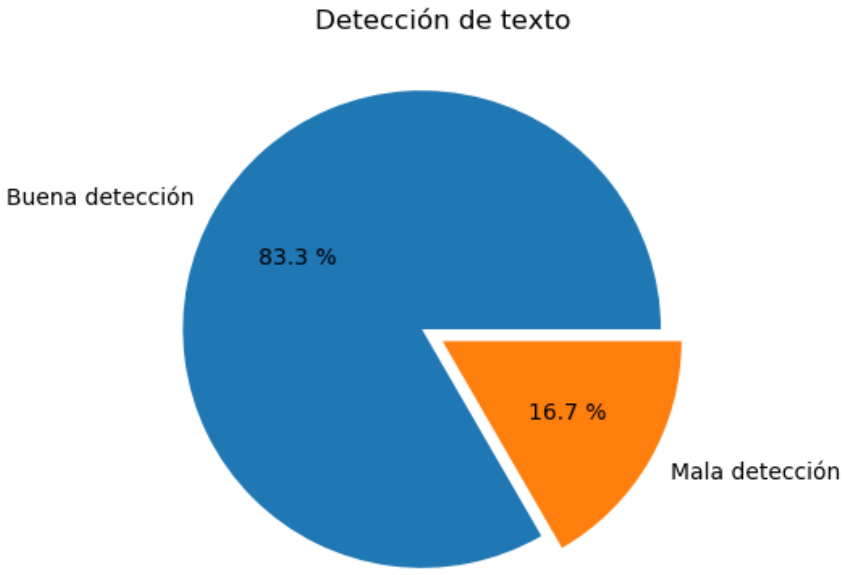
b. Hacer reconocimiento de texto en una imagen digital de un diagrama de flujo trazado a mano

A continuación se muestra en forma de gráfica los índices más representativos para evaluar el reconocimiento de texto.

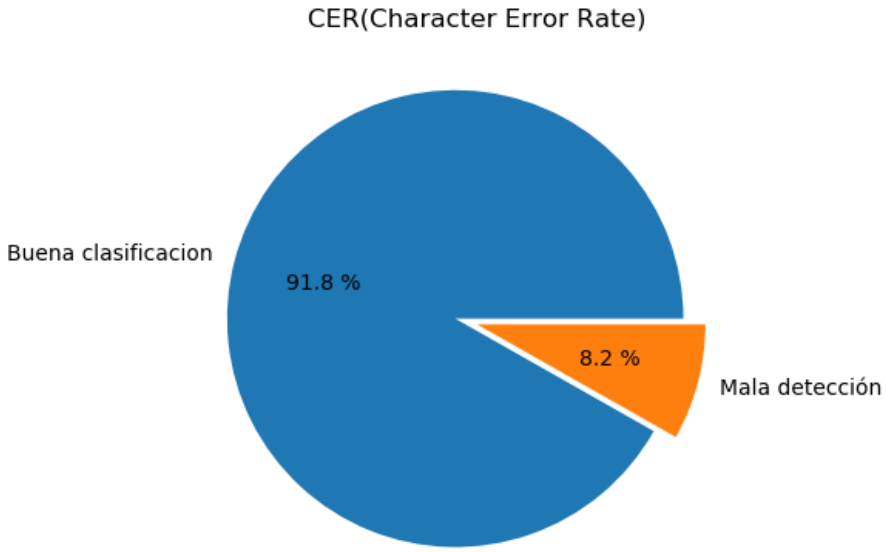
En la siguiente gráfica se muestra que el 66.7% de las pruebas realizadas en la tarea de clasificación de texto fue acertada y un 33.3% de las pruebas no lo fue.



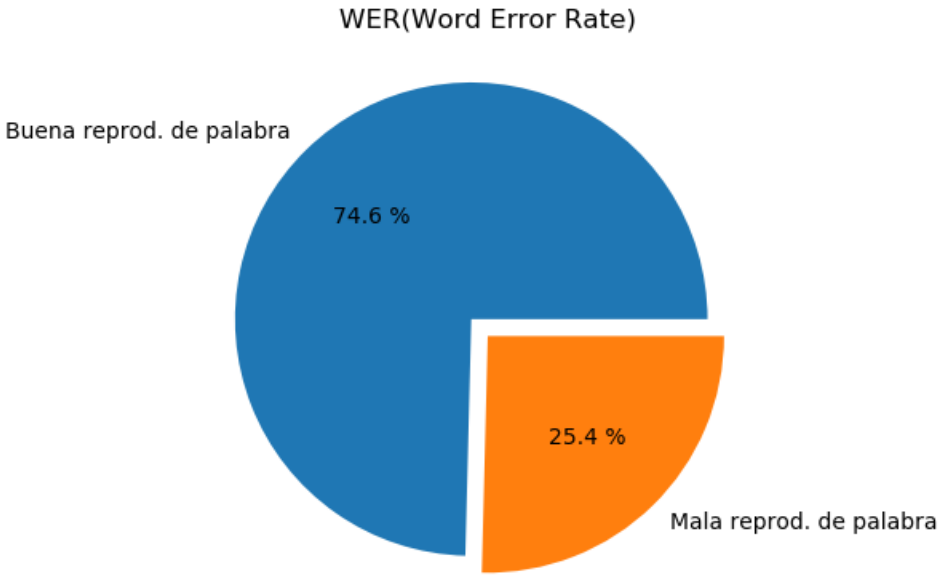
En siguiente gráfica se hace referencia a la tarea de detección de texto donde se puede observar que el 83.3% de las pruebas se detectó correctamente todo el texto en un imagen.



Otro de los índices que se consideran para evaluar la clasificación de texto es el CER, el cual indica cuánto porcentaje de los caracteres no se clasificaron correctamente por ende cuantos si fueron clasificados correctamente, la siguiente gráfica muestra dichos porcentajes:



Por último se muestra una gráfica que muestra el WER obtenido al clasificar el texto el WER indica que tan buena es la reproducción de las palabras en el texto reconocido.



Con base a los índices antes mostrados se puede determinar una evaluación para este objetivo específico dejando como resultado cumplimiento del objetivo, además de que brindarnos información de qué tan bien se está

haciendo la tarea, con los índices CER y WER se puede ver que que el objetivo además de cumplimiento muestra buenos resultados, sin embargo, se nota que de las pruebas totales que se hicieron existe un 33.3% de falla aún con un buen CER y WER, ésto se debe al contexto referente a las diferencias entre el dataset con el que fue entrenado y la entrada de datos para predecir. Para sobrellevar lo anterior se usó una técnica conocida como "Continual learning" el cual consiste que a partir de nuevas entradas en las predicciones siga aprendiendo ya con el contexto de los diagramas de flujo y así poder tener una mejor predicción conforme se le vaya alimentando de datos, esto al hacer entrenamiento con la técnica antes mencionada, por parte de la detección las no detección se deben a que el modelo que las detecta no está entrenado para detectar todos los caracteres así que se usa una técnica para unir los caracteres faltantes .

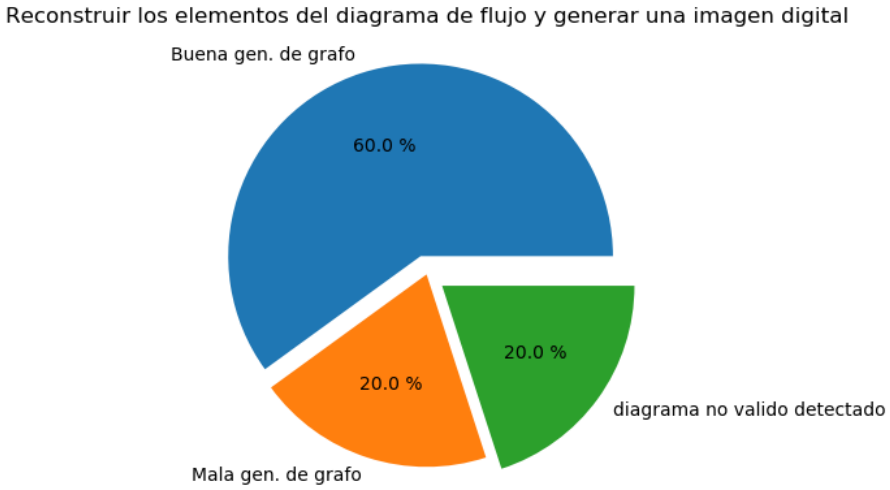
2. **Usar redes neuronales convolucionales (CNNs por sus siglas en inglés) para la tarea de detección (localización y clasificación).**

   Este objetivo especifico es evaluado por su complimiento de forma booleana ya que es referente al hecho de si se usó redes neuronales convolucionales o no para la localización y clasificación de los elementos que el proyecto necesita, siendo así evaluado como cumplido ya que tanto para la detección de figuras y conectores como para texto, las arquitecturas de los modelos usan bloques de convolución en sus arquitecturas.

3. **Usar análisis de gramática para verificar la estructura del diagrama de flujo.**

   Para la evaluación de este objetivo está directamente ligado a la generación del grafo ya que este se va construyendo a partir de las reglas gramaticales que los diagramas de flujo poseen, por lo que si en el momento de construir el grafo se detecta que no cumple una regla se termina el proceso de construcción del grafo ya que dicho diagrama no es válido, en caso de que termine el proceso tenemos un grafo que cumple con las reglas gramaticales por lo que no es necesario hacer un

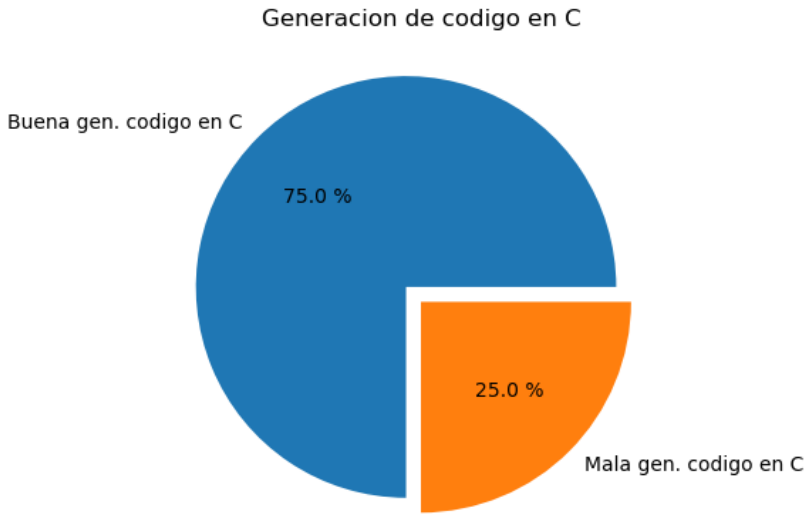
análisis, la siguiente gráfica muestra el porcentaje de los eventos ocurridos en las pruebas realizadas.



Usando la información anterior para evaluar este objetivo,se tiene que el objetivo cumple con su propósito, se puede ver que el 80% de resultados son correctos ya que los grafos que se generan correctamente como los que se clasifican como no válido son resultados esperados, aunque se cumple con el objetivo aún tiene fallas expuestas en el 20% de las pruebas ya que son de grafos no generados, por lo que se podría mejorar.

4. **Poder generar código fuente de C a partir del diagrama de flujo.**

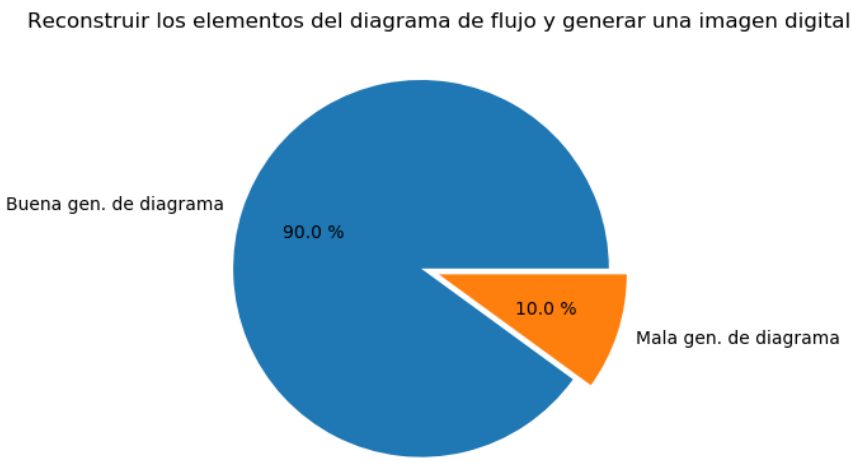
   Para este objetivo específico el cumplimiento se basa en si a partir de lo el grafo generado a partir de la detecciones de figuras, conectores y del texto se genera el código fuente en el lenguaje de programación C equivalente del diagrama plasmado en la imagen que se analizó teniendo así la siguiente gráfica que muestra del total de pruebas realizadas cuantas resolvieron su tarea correctamente y cuáles no.



Con esta información se puede dar una evaluación al objetivo siendo esta evaluada como aceptada ya que desempeña su tarea, aunque se puede mejorar el proceso por el cual se genera el código fuente en C, para reducir el número de casos en los que no se genera correctamente el código fuente.

5. **Poder digitalizar el diagrama de flujo, esto es, reconstruir los elementos del diagrama de flujo y generar una imagen digital.**

   Para este último objetivo específico el cumplimiento se basa en si a partir del grafo generado y de la detecciones de figuras, conectores y del texto se reconstruyen los elementos del diagrama de flujo para generar una imagen digital equivalente del diagrama plasmado en la imagen que se analizó teniendo así la siguiente gráfica que muestra del total de pruebas realizadas cuantas resolvieron su tarea correctamente y cuáles no.



Con la información presentada como indicado para evaluar este objetivo se determina que el objetivo desempeña su tarea, aunque se puede mejorar el proceso por el cual se genera la imagen del diagrama de flujo digital, para reducir el número de casos en los que no se genera correctamente la imagen.

Conclusiones y recomendaciones

Con base a las evaluaciones de los los objetivos particulares se generó una evaluación general siendo ésta la suma de los objetivos particulares, las cuales se mostraran a continuación en forma de tabla marcando como cumplida si el objetivo satisface la tarea o requerimiento establecido.

Objetivo particular	Evaluación
Hacer reconocimiento de figuras, texto y conectores plasmados en una imagen digital de un diagrama de flujo trazado a mano.	Cumplida
Usar redes neuronales convolucionales (CNNs por sus siglas en inglés) para la tarea de detección (localización y clasificación).	Cumplida
Usar análisis de gramática para verificar la estructura del diagrama de flujo.	Cumplida
Poder generar código fuente de C a partir del diagrama de flujo.	Cumplida
Poder digitalizar el diagrama de flujo, esto es, reconstruir los elementos del diagrama de flujo y generar una imagen digital.	Cumplida

Ya que todos los objetivos particulares se cumplen, la evaluación del objetivo general tiene como resultado cumplimiento del objetivo, así como la evaluación del objetivo general se realizó con base a los objetivos particulares, así también se hacen las recomendaciones:

Objetivo particular	Recomendación
Hacer reconocimiento de figuras, texto y conectores plasmados en una imagen digital de un diagrama de flujo trazado a mano.	● Para el reconocimiento de figuras y conectores las recomendaciones son las siguientes: ○ Mejorar los índices de generación correcta de diagramas de flujo y códigos fuentes al hacer más robustos dichos módulos. ○ Uso del algoritmo edit distance para corregir automáticamente los start_end, ya que siempre son "inicio" y "fin. ○ Aplicar “Continual learning” también al modelo de figuras, con el fin de hacerlo óptimo al estilo de trazado de un usuario. ○ Entrenar el modelo de figuras agregando más símbolos, como el de continue, para poder reconocer un diagrama que está en varias fotos. ● Para el reconocimiento de figuras y conectores las recomendaciones son las siguientes: ○ Usar un modelo más robusto y entrenado con todos los caracteres que el proyecto necesita. ○ Usar la nueva tecnología de Transformers en el modelo de clasificación de texto.

Usar redes neuronales convolucionales (CNNs por sus siglas en inglés) para la tarea de detección (localización y clasificación).	No hay recomendaciones.
Usar análisis de gramática para verificar la estructura del diagrama de flujo.	Mejorar el algoritmo de construcción del grafo.
Poder generar código fuente de C a partir del diagrama de flujo.	Mejorar el algoritmo de generacion de codigo fuente en C.
Poder digitalizar el diagrama de flujo, esto es, reconstruir los elementos del diagrama de flujo y generar una imagen digital.	Mejorar el algoritmo de generación de la imagen del diagrama de flujo digital.

Referencias

[1] W. Szwoch and M. Mucha, “Recognition of Hand Drawn Flowcharts” Advances in Intelligent Systems and Computing Image Processing and Communications Challenges 4 , vol. 184, pp. 65–72, 2013.

[2] O. Cairó Battistutti, Metodología de la programación , 3rd ed. México, D.F.: Alfaomega, 1995, pp. 3, 4-8.