



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**Aplicaciones de Visión  
Artificial en Dispositivos de  
Edge Computing  
Documentación Técnica**



Presentado por Miriam Torres Calvo  
en Universidad de Burgos — 10 de septiembre  
de 2022

Tutor: Bruno Baruque Zanón



---

# Índice general

---

<b>Índice general</b>	<b>i</b>
<b>Índice de figuras</b>	<b>iii</b>
<b>Índice de tablas</b>	<b>iv</b>
<b>Apéndice A Plan de Proyecto Software</b>	<b>1</b>
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	1
A.3. Estudio de viabilidad . . . . .	1
<b>Apéndice B Especificación de Requisitos</b>	<b>3</b>
B.1. Introducción . . . . .	3
B.2. Objetivos generales . . . . .	3
B.3. Catalogo de requisitos . . . . .	3
B.4. Especificación de requisitos . . . . .	3
<b>Apéndice C Especificación de diseño</b>	<b>5</b>
C.1. Introducción . . . . .	5
C.2. Diseño de datos . . . . .	5
C.3. Diseño procedimental . . . . .	5
C.4. Diseño arquitectónico . . . . .	5
<b>Apéndice D Documentación técnica de programación</b>	<b>7</b>
D.1. Introducción . . . . .	7
D.2. Estructura de directorios . . . . .	7
D.3. Manual del programador . . . . .	12

D.4. Compilación, instalación y ejecución del proyecto . . . . .	12
D.5. Pruebas del sistema . . . . .	12
<b>Apéndice E Documentación de usuario</b>	<b>13</b>
E.1. Introducción . . . . .	13
E.2. Requisitos de usuarios . . . . .	13
E.3. Instalación . . . . .	13
E.4. Manual del usuario . . . . .	13
<b>Bibliografía</b>	<b>15</b>

---

## Índice de figuras

---

---

# Índice de tablas

---

## *Apéndice A*

---

# **Plan de Proyecto Software**

---

### **A.1. Introducción**

En este apéndice se va a mostrar la planificación del proyecto, la cuál es la base sobre la crea el proyecto *software*. Desde el punto de vista de la temporalidad y viabilidad. Siendo está una parte fundamental del proyecto, ya que permite visualizar el escenario en el que se desarrollará, de tal forma que podamos realizar una alineación estrategica de los elementos que deben de ser completados, con el objetivo de finalizarlo correctamente.

### **A.2. Planificación temporal**

La planificación temporal se

### **A.3. Estudio de viabilidad**

**Viabilidad económica**

**Viabilidad legal**





## *Apéndice B*

---

# **Especificación de Requisitos**

---

### **B.1. Introducción**

En este apéndice se recogen las necesidades funcionales que deberán de ser soportadas por el sistema que va a ser desarrollado. Con el objetivo de obtener una buena documentación, deben de identificarse y describirse los requisitos que tienen que ser satisfechos por el sistema, pero sin entrar en su proceso de realización.

### **B.2. Objetivos generales**

### **B.3. Catalogo de requisitos**

### **B.4. Especificación de requisitos**



## *Apéndice C*

---

# **Especificación de diseño**

---

### **C.1. Introducción**

En este apéndice se va a exponer cómo se han resuelto los objetivos anteriormente comentados. Así como la definición de datos que se utilizan en la aplicación, procedimientos ...

### **C.2. Diseño de datos**

### **C.3. Diseño procedimental**

### **C.4. Diseño arquitectónico**



## *Apéndice D*

---

# Documentación técnica de programación

---

## D.1. Introducción

En este apéndice van a describirse de forma detallada la documentación técnica de programación. Se describirá la estructura de directorios que posee, la instalación y ejecución, así como las pruebas que se han llevado a cabo.

## D.2. Estructura de directorios

- `/`: es la raíz del proyecto dónde se encuentran tanto el README, la licencia y las carpetas contenedoras del código, documentación y las pruebas previas.
- `/codigo`: es la carpeta que contiene todo el código funcional del proyecto.
- `/codigo/checkpoints`: es la carpeta contenedora de los modelos de detección en formato Tensorflow, Tensorflow Lite y Tensor-RT.
- `/codigo/checkpoints/custom-416`: carpeta que contiene el modelo de detección de las matrículas, que posee el tamaño 416.
- `/codigo/checkpoints/custom-416/saved_model.pb`: modelo en formato Tensorflow(.pb) de las matrículas.
- `/codigo/checkpoints/heads-416`: carpeta con el modelo de las cabezas en formato Tensorflow.
- `/codigo/checkpoints/heads-416/keras_metadata.pb`: punto de control del modelo de conversión a .pb.

- `/codigo/checkpoints/heads-416/saved_model.pb`: modelo detector de cabezas en formato Tensorflow(.pb)
- `/codigo/checkpoints/yolov4-416`: carpeta con el modelo oficial de YOLOv4 en formato Tensorflow.
- `/codigo/checkpoints/yolov4-416/keras_metadata.pb`: punto de control del modelo de conversión a .pb.
- `/codigo/checkpoints/yolov4-416/saved_model.pb`: modelo detector de YOLOv4 en formato Tensorflow(.pb)
- `/codigo/checkpoints/custom_tfl-416`: carpeta con el modelo de detección de las matrículas en formato Tensorflow previo a la conversión a TensorFlow Lite.
- `/codigo/checkpoints/custom_tfl-416/saved_model.pb`: modelo detector de matrículas en formato Tensorflow(.pb) preparado para su conversión a .tflite.
- `/codigo/checkpoints/custom_tflv2-416`: carpeta con el modelo de detección de las matrículas en formato Tensorflow previo a la conversión a TensorFlow Lite.
- `/codigo/checkpoints/custom_tflv2-416/saved_model.pb`: modelo detector de matrículas en formato Tensorflow(.pb) preparado para su conversión a .tflite.
- `/codigo/checkpoints/custom-416-int8.tflite`: modelo de detección de las matrículas en formato TensorFlow Lite(.tflite).
- `/codigo/checkpoints/custom-416v2.tflite.tflite`: modelo de detección de las matrículas en formato TensorFlow Lite(.tflite)
- `/codigo/checkpoints/models_trt.txt`: fichero con los enlaces de los modelos de TensorRT.
- `/codigo/core`: carpeta con los ficheros de configuración utilizados durante el proyecto.
- `/codigo/core/backbone.py`: fichero de Python que contine las funciones relacionadas con la red YOLOv4
- `/codigo/core/commom.py`: fichero de Python que contine la clase BatchNormalization, para los ajustes de la red YOLOv4
- `/codigo/core/config.py`: fichero de Python que contine permite la selección de los ficheros de etiquetas de cara al uso del modelo
- `/codigo/core/functions.py`: fichero de Python que contine las funciones utilizadas en a lo largo de la detección de los objetos.
- `/codigo/core/utils.py`: fichero de Python que contine las funciones relacionadas con la red YOLOv4.
- `/codigo/core/yolov4.py`: fichero de Python que retorna el modelo de YOLO correspondiente.

- `/codigo/data`: carpeta que contine la información necesaria para la detección.
- `/codigo/data/anchors`: carpeta que contiene los anchors de lass diferentes redes.
- `/codigo/data/anchors/baseline_anchors.txt`: fichero de anchors.
- `/codigo/data/anchors/baseline_tiny_anchors.txt`: fichero de anchors tiny.
- `/codigo/data/anchors/yolov3_anchors.txt`: fichero de anchors yolov3.
- `/codigo/data/anchors/yolov3_anchors.txt`: fichero de anchors yolov4.
- `/codigo/data/classes`: carpeta que contiene los fichero de etiquetas de los diferentes modelos.
- `/codigo/data/classes/coco.names`: fichero con las etiquetas del modelo oficial de YOLOV4.
- `/codigo/data/classes/custom.names`: fichero con las etiquetas del modelo de detección de matrículas.
- `/codigo/data/classes/heads.names`: fichero con las etiquetas del modelo de detección de las cabezas.
- `/codigo/data/classes/voc.names`: fichero de etiquetas del modelo de voc.
- `/codigo/data/classes/yymnist.names`: fichero de etiquetas del modelo de yymnist, detector de números.
- `/codigo/data/dataset`: carpeta que contiene los ficheros etiquetados a la hora de evaluar un modelo (ruta de la imagen posición detectada y valor de la clase).
- `/codigo/data/dataset/head.txt`: fichero de evaluacion del modelo de las cabezas.
- `/codigo/data/dataset/license_plate.txt`: fichero de evaluación del modelo de las matrículas.
- `/codigo/data/dataset/val2017.txt`: fichero de evaluación del modelo coco.
- `/codigo/data/images`: carpeta que contiene diferentes imagenes para su detección.
- `/codigo/data/video`: carpeta que contien diferentes vídeos para su detección/contabilización.
- `/codigo/deep_sort`: carpeta que contiene los diferentes ficheros en Python para su evaluacion con Object Tracking.
- `/codigo/deep_sort/detection.py`: fichero Python que contiene las funciones de detección para Obejct Tracking.

- `/codigo/deep_sort/iou_matching.py`: fichero Python que tiene las funciones de la medida iou para Object Tracking.
- `/codigo/deep_sort/kalman_filter.py`: fichero Python que contiene el algoritmo del filtro de Kalman[1].
- `/codigo/deep_sort/linear_assignment.py`: Fichero Python que contiene las funciones relacionadas con la asignación lineal.
- `/codigo/deep_sort/nn_matching.py`: Fichero Python con funciones de ajuste del algoritmo de vecinos más cercanos[2].
- `/codigo/deep_sort/preprocessing.py`: Fichero Python con las funciones del preprocesado para Object Tracking.
- `/codigo/deep_sort/track.py`: fichero Python que contiene las funciones necesarias para detectar los objetos y sus etiquetas correspondientes, con su respectivo número de identificación.
- `/codigo/deep_sort/tracker.py`: fichero Python que contiene las funciones necesarias para detectar los objetos y sus etiquetas correspondientes, con su respectivo número de identificación.
- `/codigo/detections`: carpeta con los resultados de las detecciones obtenidas mediante la línea de comandos.
- `/codigo/detections/images`: carpeta con el resultado de las imágenes detectadas.
- `/codigo/detections/videos`: carpeta con el resultado de los vídeos detectados.
- `/codigo/mAP`: carpeta que contiene los resultados de las evaluaciones de los modelos, así como scripts de ayuda para ello.
- `/codigo/mAP/extra`: carpeta con los scripts de ayuda para la evaluación del modelo.
- `/codigo/mAP/extra/intersect-gt-and-pred.py`: fichero Python que calcula la intersección entre la posición real del objeto y la obtenida por el modelo, con el objetivo de evaluar la calidad del modelo.
- `/codigo/mAP/extra/remove_space.py`: fichero Python que elimina los espacios de las etiquetas de las clases de los modelos.
- `/codigo/mAP/ground-truth`: carpeta que contiene los ficheros .txt de cada imagen a evaluar con sus posiciones originales en formato YOLO, junto el nombre de la etiqueta que le corresponde.
- `/codigo/mAP/predicted`: carpeta que contiene los ficheros .txt de cada imagen a evaluar con sus posiciones detectadas en formato YOLO, junto el nombre de la etiqueta que le corresponde.
- `/codigo/mAP/results_custom_tf_complete`: carpeta con los resultados de la evaluación del modelo de las matrículas.
- `/codigo/mAP/results_heads_tf_complete`: carpeta con los resultados de la evaluación del modelo de las cabezas.



- `/codigo/mAP/main.py`: fichero Python que representa el resultado de la evaluación del modelo.
- `/codigo/model_data`: carpeta que contiene el modelo `mars-small128.pb`, utilizado en la inicialización de Object Tracking.
- `/codigo/static`: carpeta que contiene los ficheros 'estaticos' para Flask.
- `/codigo/static/css`: carpeta que contiene los diferentes ficheros de estilos[3] usados a lo largo de la app Flask.
- `/codigo/static/js`: carpeta que contiene los diferentes scripts de JavaScript[4] utilizados a lo largo de la app Flask.
- `/codigo/static/detections`: carpeta que contiene las imagenes, videos etiquetados tras su detección, así como los ficheros CSV de las posiciones.
- `/codigo/static/imgs`: carpeta con todas las imagenes usadas a lo largo de la app Flask.
- `/codigo/temp`: carpeta que almacena los ficheros de detección temporales, generados al inicio de las detecciones en la app Flask.
- `/codigo/templates`: carpeta que contiene los ficheros .html usados a lo largo de la app Flask.
- `/codigo/tools`: carpeta que contiene los scripts Python utilizados como herramientas a la hora de detectar.
- `/codigo/tools/freeze_model.py`: script Python que convierte el gráfico del modelo de TensorFlow a uno con extensión .pb.
- `/codigo/tools/generate_detections.py`: script Python que obtiene las 'cajas' en las cuáles se encuentran los objetos que han sido detectados por el modelo.
- `/codigo/train`: carpeta que tiene los scripts de Python y de GoogleColab, así como los ficheros necesarios para llevar a cabo el entrenamiento de un modelo de YOLOv4.
- `/codigo/trt`: carpeta que contiene el script de GoogleColab de conversión del fichero de pesos de YOLOv4 (.weights) a un modelo de TensorRT.
- `/codigo/app.py`: fichero de Python que es la propia app de Flask.
- `/codigo/convert_tflite.py`: fichero de Python que convierte el modelo deseado a uno de TensorFlow Lite.
- `/codigo/convert_trt.py`: fichero de Python que convierte el modelo deseado a uno de TensorRT.
- `/codigo/detect.py`: fichero de Python que detecta objetos en una imagen, según un modelo de detección.
- `/codigo/detectVideo.py`: fichero de Python que detecta objetos en un vídeo, según un modelo de detección.

- `/codigo/evaluate.py`: fichero de Python que evalúa un modelo de detección, con el objetivo de medir su calidad a 1 hora de predecir.
- `/codigo/objectTracker.py`: fichero de Python que contabiliza objetos en un vídeo, según un modelo de detección.
- `/codigo/preprocessDataEvaluate.py`: fichero de Python que obtiene las posiciones de las imágenes en el formato necesario para su evaluación.
- `/codigo/save_model_tflite.py`: fichero de Python que convierte un fichero de pesos en formato `.weights` a un modelo de TensorFlow Lite.
- `/codigo/save_model.py`: fichero de Python que convierte un fichero de pesos en formato `.weights` a un modelo de TensorFlow.

### D.3. Manual del programador

En esta subsección se describen todos los recursos utilizados para poder llevar a cabo el proyecto. De tal forma que un futuro desarrollador/mantenedor del proyecto no tenga inconvenientes a la hora de retomar el proyecto y conocerlo.

#### Entorno de desarrollo

Para poder continuar con el desarrollo del proyecto, será necesario contar con el siguiente *software* instalado en el equipo:

- Python 3.7
- Bibliotecas de Python
- VSCode

A continuación, se comentará de forma detallada la instalación de los diferentes requerimientos.

### D.4. Compilación, instalación y ejecución del proyecto

### D.5. Pruebas del sistema

## *Apéndice E*

---

# Documentación de usuario

---

### **E.1. Introducción**

En este apéndice van a detallarse los requerimientos de la aplicación, su instalación y consejos de cara a usarlo de manera correcta.

### **E.2. Requisitos de usuarios**

### **E.3. Instalación**

### **E.4. Manual del usuario**



---

## Bibliografía

---

- [1] ScienceDirect, “Kalman filter,” 2019. [Online]. Available: <https://www.sciencedirect.com/topics/earth-and-planetary-sciences/kalman-filter#:~:text=The%20Kalman%20Filter%20is%20an,the%20uncertainty%20of%20the%20estimates>.
- [2] IBM, “K-nearest neighbors algorithm.” [Online]. Available: <https://www.ibm.com/topics/knn>
- [3] Mozilla, “What is css?” [Online]. Available: [https://developer.mozilla.org/es/docs/Learn/CSS/First\\_steps/What\\_is\\_CSS](https://developer.mozilla.org/es/docs/Learn/CSS/First_steps/What_is_CSS)
- [4] Moxilla, “Javascript.” [Online]. Available: <https://developer.mozilla.org/es/docs/Web/JavaScript>