Texto

Descripción generada automáticamente con confianza baja

**UNIVERSIDAD EUROPEA DE MADRID  
ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO**

**MÁSTER UNIVERSITARIO EN**

**ANÁLISIS DE DATOS MASIVOS (BIG DATA)**

TRABAJO FIN DE MÁSTER

**ESTIMACIÓN DEL TRÁFICO RODADO A PARTIR DEL ANÁLISIS VISUAL DE LAS CÁMARAS URBANAS**

**NOMBRE:**

**IVÁN SAN MARTÍN FERNÁNDEZ**

**CURSO 2021-2022**

**TÍTULO**: ESTIMACIÓN DEL TRÁFICO RODADO A PARTIR DEL ANÁLISISIS VISUAL DE LAS CÁMARAS URBANAS

**AUTOR**: IVÁN SAN MARTÍN FERNÁNDEZ

**TITULACIÓN**: MÁSTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS (BIG DATA)

**DIRECTOR DEL PROYECTO**: Raúl Pérula Martínez

**FECHA**: Septiembre de 2022

# RESUMEN

El resumen tiene entre 150-250 palabras. Resumir consiste en ofrecer información exhaustiva sobre cómo, dónde, cuándo y por qué se aplica el proyecto. Se realiza al finalizar el trabajo.

El resumen

* Establece el tema tratado en el proyecto
* Resume los problemas planteados y las aportaciones más importantes del proyecto
* Resume las conclusiones más importantes del trabajo realizado

El resumen NO

* Da una información genérica
* Explica porque se realiza ese trabajo
* Se refiere a datos aportados en el texto del proyecto.

**Palabras clave:** hasta un máximo de 6

# ABSTRACT

Resumen en inglés.

**Key words:** Palabras clave en inglés

**AGRADECIMIENTOS**

Hace años, cuando empecé a estudiar ingeniería informática, en 2017, no había oído hablar de

aprendizaje automático. Había oído hablar de la conducción autónoma o del reconocimiento de personas, pero no sabía lo que había detrás y, siendo sincero, tampoco es algo que me interesase demasiado. Tiempo después, 3 años después exactamente, asistí a mi primer curso de aprendizaje automático y pude comprender la importancia y la magia en este campo, fue entonces cuando me di cuenta de que quería profundizar en esto. Aprendiendo sobre el Deep Learning e investigando nuevas técnicas me di cuenta de que era mi vocación.

En esto se basa mi motivación en esta tesis. Para mí es una oportunidad de aprender cómo empezar a trabajar en este mundo y cómo funciona.

Agradezco a mi director de tesis: Raúl Pérula Martínez, por su motivación y guía durante todo este proyecto y por la oportunidad de aprender sobre la investigación en Deep Learning.

Índice

[RESUMEN 4](#_Toc111118459)

[ABSTRACT 4](#_Toc111118460)

[Capítulo 1. INTRODUCCIÓN 11](#_Toc111118461)

[1.1 Visión General 11](#_Toc111118462)

[1.2 Motivación 11](#_Toc111118463)

[1.3 Objetivos 11](#_Toc111118464)

[1.4 Timeline (Falta modificar fechas) 11](#_Toc111118465)

[1.4.1 Fase inicial 11](#_Toc111118466)

[1.4.2 Fase de preparación 12](#_Toc111118467)

[1.4.3 Fase de desarrollo 12](#_Toc111118468)

[1.4.4 Fase de transición 12](#_Toc111118469)

[1.5 Esquema 13](#_Toc111118470)

[Capítulo 2. Estado del Arte (Revisar numero imágenes y links a capitulos) 14](#_Toc111118471)

[2.1 Introducción 14](#_Toc111118472)

[2.2 Funcionamiento General 14](#_Toc111118473)

[2.2.1 Extracción de características 15](#_Toc111118474)

[2.2.2 Estudio de características 15](#_Toc111118475)

[2.3 Extracción de características 15](#_Toc111118476)

[2.3.1 Capa Convolucional 15](#_Toc111118477)

[2.3.2 Capa de Pooling 16](#_Toc111118478)

[2.4 Capa de Flattening 17](#_Toc111118479)

[2.5 Estudio de las características 17](#_Toc111118480)

[2.5.1 Capas Densas 17](#_Toc111118481)

[2.6 Preprocesado 19](#_Toc111118482)

[2.6.1 Ruido Gaussiano 19](#_Toc111118483)

[2.6.2 Ruido Impulsivo 20](#_Toc111118484)

[2.6.3 Aumento de Datos 21](#_Toc111118485)

[2.7 Overfitting y underfitting 21](#_Toc111118486)

[Capítulo 3. Proyectos Relacionados 23](#_Toc111118487)

[3.1 Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction 23](#_Toc111118488)

[3.2 Estimating Urban Traffic Density Using Street Camera Images 23](#_Toc111118489)

[3.3 Boosting Multi-Vehicle Tracking with a Joint Object Detection and Viewpoint Estimation Sensor 23](#_Toc111118490)

[Capítulo 4. MATERIALES Y MÉTODOS 24](#_Toc111118491)

[4.1 Frameworks 24](#_Toc111118492)

[4.1.1 Keras 24](#_Toc111118493)

[4.1.2 PyTorch 24](#_Toc111118494)

[4.1.3 YOLOv3 24](#_Toc111118495)

[4.2 Datasets 25](#_Toc111118496)

[4.2.1 Big Data for Traffic Estimation and Prediction 25](#_Toc111118497)

[4.2.2 Cameras Florida 25](#_Toc111118498)

[4.2.3 Traffic Congestion Dataset 26](#_Toc111118499)

[4.2.4 Photole and Plain Road Dataset 26](#_Toc111118500)

[4.3 Hardware 27](#_Toc111118501)

[4.3.1 Google Colab 27](#_Toc111118502)

[4.3.2 Rog Strix 27](#_Toc111118503)

[4.4 Otras herramientas 27](#_Toc111118504)

[4.4.1 GitHub 27](#_Toc111118505)

[4.4.2 ResNet50 27](#_Toc111118506)

[4.4.3 SHAP 27](#_Toc111118507)

[Capítulo 5. ARQUITECTURA 28](#_Toc111118508)

[5.1 Método propuesto 1 28](#_Toc111118509)

[5.2 Método propuesto 2 28](#_Toc111118510)

[5.2.1 Ingesta de datos 28](#_Toc111118511)

[5.2.2 Preprocesado 28](#_Toc111118512)

[5.2.3 Aumentado de datos 29](#_Toc111118513)

[5.2.4 Arquitectura de la red 30](#_Toc111118514)

[5.3 Experimentación 31](#_Toc111118515)

[5.3.1 Overfitting 31](#_Toc111118516)

[5.3.2 Funcionamiento de la red 32](#_Toc111118517)

[5.3.3 Comparación de modelos 33](#_Toc111118518)

[5.4 Resultados 34](#_Toc111118519)

[Capítulo 6. CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO 35](#_Toc111118520)

[ANEXOS 36](#_Toc111118521)

[PRESUPUESTO 38](#_Toc111118522)

[BIBLIOGRAFÍA 39](#_Toc111118523)

Índice de Figuras

Este índice ha de existir tan sólo en el caso de que tengas más de 4 apoyos visuales (gráficos, figuras, ilustraciones, fotos, etc.) en tu memoria del proyecto.

Recuerda esta sección es optativa.

Índice de Tablas

Este índice ha de existir tan sólo en el caso de que tengas más de 4 tablas de datos en tu memoria del proyecto.

Recuerda esta sección es optativa.

# INTRODUCCIÓN

*Este primer capítulo está dedicado a introducir el tema principal de la tesis y su contexto.*

*En primer lugar, repasamos el trabajo realizado. Luego presentamos nuestra motivación, propuesta y objetivos, seguido de una línea temporal que define los pasos para desarrollar este trabajo y el esquema del proyecto.*

## Visión General

En este trabajo, nos enfocamos en la estimación del tráfico rodado a partir de las cámaras urbanas. Nuestro modelo centra su atención en la cantidad de “ruido” que encontramos en la carretera captada por las cámaras de tráfico con el objetivo de poder estimar el tráfico a tiempo real y poder así prevenir accidentes de tráfico, optimizar el funcionamiento de los semáforos o incluso ahorrar energía apagando temporalmente la iluminación no aprovechada.

Más específicamente, estamos interesados en calcular la cantidad de tráfico en varias coordenadas distintas y, dependiendo del contexto, realizar las acciones oportunas.

## Motivación

Los accidentes de tráfico se cobraron la vida de 1.004 personas el pasado año. En 2021 se produjeron 921 siniestros mortales en las carreteras españolas en los que fallecieron 1.004 personas y otras 3.728 resultaron heridas graves.

Esto son unas cifras escandalosas que podrían reducirse considerablemente planteando una conducción óptima y un aprovechamiento de las carreteras.

En este trabajo proponemos una posible solución analizando el tráfico en varios puntos de nuestro país y redirigiéndolo de la manera más óptima.

## Objetivos

El principal objetivo de este proyecto es el de analizar el tráfico para, en un futuro proyecto disponer de datos suficientes para trazar un plan de redirección de tráfico y preparar los servicios oportunos en las zonas más concurridas. Estopo reducirá el número de accidentes y la mortalidad de estos considerablemente.

## Timeline (Falta modificar fechas)

Hemos dividido nuestra línea de tiempo (ver Figura 1.1) en 4 fases que describimos en detalle a continuación.

Calendario

Descripción generada automáticamente

### Fase inicial

Para comenzar, dibujamos y analizamos el proyecto. Esta fase incluye una reunión inicial, donde analizamos el proyecto, y fijamos objetivos generales y específicos, las limitaciones del proyecto y su alcance. Además, durante esta primera etapa elegimos las herramientas y marcos adecuados. La duración de esta fase inicial fue de 11 días hábiles.

### Fase de preparación

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Tabla

Descripción generada automáticamente

Esta segunda fase (Figura 1.2) consiste en revisar los modelos existentes. Primero, revisamos

obras relacionadas. Luego, determinamos unos plazos y comenzamos con el proyecto. Incluye

las herramientas de configuración seleccionadas en la fase inicial y la búsqueda de conjuntos de datos y modelos que pueden ser útil para nuestro proyecto. Estimamos 76 días hábiles para esta fase, debido a que la tarea de investigación es la más larga.

### Fase de desarrollo

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Una vez que hayamos elegido cada modelo y conjunto de datos, podemos comenzar a implementar el código. En esta fase(Figura 1.3), codificamos todos los métodos propuestos y generamos una pequeña prueba para verificar que funcionen correctamente. Estimamos alrededor de 31 días para esta fase.

### Fase de transición

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

En esta última fase (Figura 1.4), el propósito principal es hacer experimentos y analizar el

código y los métodos que hemos implementado. Lo más importante en esta fase es extraer

conclusiones de nuestro proyecto. Estimamos 28 días para esta fase.

## Esquema

Este trabajo consta de 6 capítulos: En el capítulo 1 presentamos la motivación, metas y calendario. En el capítulo 2 explicamos la estructura del algoritmo utilizado. En el capítulo 3 revisamos los trabajos relacionados. Se presentan los materiales y métodos utilizados. en el capítulo 4. Se describe la propuesta de nuestra arquitectura con experimentos cuantitativos en el capítulo 5 y experimentos cualitativos con imágenes reales. Resumimos las conclusiones y trabajos adicionales en el capítulo 6.

# Estado del Arte (Revisar numero imágenes y links a capitulos)

*En este capítulo se explicará de forma general el funcionamiento del algoritmo a utilizar en el proyecto, las redes neuronales convolucionales.*

## Introducción

Para comenzar es importante conocer el significado y la utilidad de las Redes Neuronales Convolucionales o Convolutional Neural Network (CNN).

A lo largo de los años se ha investigado mucho sobre cómo automatizar ciertas tareas monótonas para el ser humano, entre las que se encuentran algunas como vigilancia a través de cámaras de seguridad, lograr una conducción autónoma o la detección de anomalías en imágenes. El problema de estas tareas es que requieren de entender datos algo complejos debido a su estructura espacial, como imágenes o vídeos.

Para solventar este problema nacen las CNN, redes neuronales utilizadas para problemas que contengan datos con una estructura espacial.

## Funcionamiento General

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Imagen 1. Estructura de una CNN. [i](#i)

Para entender el funcionamiento de estas redes comenzaremos explicando cómo se estructuran. Las CNN se componen de 2 partes principales (Imagen 1):

### Extracción de características

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Imagen 2. Extracción de características. [i](#i)

En esta primera parte se extraen las características principales de la imagen a través de capas convolucionales y de pooling, explicadas en los puntos 2.1 y 2.2. (Imagen 2)

Tras esta extracción de características, se adaptan los datos para su estudio a través de la capa de Flattening que, simplemente, consiste en una capa que redimensiona los datos en una única dimensión.

### Estudio de características

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Imagen 3. Estudio de características. [i](#i)

Por último, tras extraer las características de la imagen se realiza un estudio de estas características haciendo uso de una red neuronal básica compuesta de capas densas. (Imagen 3)

## Extracción de características

Para la extracción de características hace falta entender 2 pasos importantes. Por un lado se extraen las características de una imagen y, por otro lado, se resumen esas características.

### Capa Convolucional

La capa convolucional es la encargada de extraer las características de una imagen. Para esto hace uso de filtros convolucionales o kernel que funcionan de la siguiente manera:

Para facilitar la comprensión supondremos que únicamente usaremos 1 filtro, de tal forma que partimos de una imagen y del filtro que utilizaremos para realizar la convolución (Imagen 4).

Imagen que contiene Aplicación

Descripción generada automáticamente

Imagen 4. Parámetros de entrada en capa convolucional. [iii](#iii)

Para llevar a cabo la convolución se coloca el kernel sobre la imagen y se multiplica cada uno de los píxeles que abarca por sus respectivas posiciones en el kernel, a continuación, se suman sus puntuaciones y se coloca el resultado en el píxel central (Figura 5).

Imagen que contiene biombo, juego, reloj, tren

Descripción generada automáticamente

Imagen 5. Convolución 3x3. [iii](#iii)

Por ejemplo, para extraer las líneas verticales utilizaríamos el siguiente filtro:

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 0 |
| 0 | 2 | 0 |
| 0 | 1 | 0 |

Hasta hace unos años todo este proceso se realizaba de forma manual eligiendo los filtros en función de las características que buscábamos. Pero, gracias a la aparición de las redes neuronales, estos filtros ya pueden calcularse de forma automática de igual forma que actualizamos los pesos en una red perceptrón multicapa (Punto 4.1).

### Capa de Pooling

Tras realizar esta extracción, se resumen estas características aplicando el denominado Pooling. Para esta explicación supondremos que realizaremos un Max-Pooling de 2x2.

Esto, simplemente, consiste en calcular el valor máximo de varios píxeles vecinos de tal forma que, si tenemos una matriz de 4x4 y aplicamos este pooling, nos quedaría una matriz resultado de 2x2.

Imagen que contiene Tabla

Descripción generada automáticamente

Imagen 6. Max-Pooling 2x2. [iii](#iii)

## Capa de Flattening

Tras varias capas convolucionales y de pooling nos falta realizar el estudio de las características obtenidas. Para esto antes deberemos adaptar los datos a la entrada de las siguientes capas (Capas Densas). Para esto, simplemente, redimensionaremos los datos a una única dimensión.

## Estudio de las características

### Capas Densas

Diagrama

Descripción generada automáticamente

Imagen 7. Perceptrón multicapa. [iv](#iv)

Son las capas principales a la hora de realizar el estudio de estas características.

Estas capas son simplemente funciones con varios parámetros o pesos a actualizar. Es decir, cada una de las capas contiene varias neuronas (Imagen 8) que consisten en funciones lineales a las cuales se les aplica una función de activación, como puede ser una función ReLu, sigmoidal, etc.

Diagrama

Descripción generada automáticamente

Imagen 8. Neurona. [iv](#iv)

Una vez tenemos una de estas funciones por cada neurona se introduce una serie de entradas a cada una de estas neuronas. A continuación, se utiliza el resultado obtenido en cada una de ellas como dato de entrada a las neuronas de la siguiente capa. (Imagen 9)

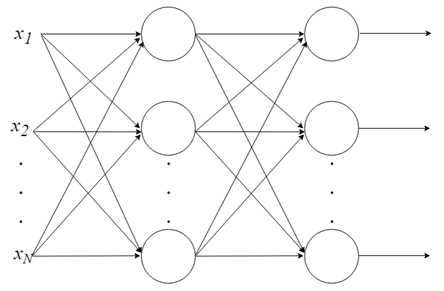


Imagen 9. Esquema Perceptrón Multicapa. [ii](#ii)

Este proceso se repite hasta finalizar con la capa de salida que, como su propio nombre indica, nos aporta el resultado final de la red neuronal. Este resultado puede ser una clasificación o un número (regresión).

Una vez obtenemos este resultado, se compara con el resultado real y se calcula cómo de grande ha sido el error para, en función del error, adaptar los pesos de cada una de las neuronas optimizando la función con el objetivo de minimizar este error y lograr que, con los datos reales, el error sea mínimo. A este proceso se le llama back propagation.

## Preprocesado

Ahora que ya hemos visto cómo funcionan las redes convolucionales vamos a explicar algunos métodos de preprocesado para mejorar el rendimiento de la red.

### Ruido Gaussiano

En todas las imágenes tenemos, en mayor o menor medida, el llamado ruido gaussiano (Imagen 10) debido a la radiación electromagnética o a falta de iluminación y/o temperatura. Esto confunde a la red a la hora de entrenar y de predecir el resultado.

Foto en blanco y negro de una mujer con un sombrero en la cabeza

Descripción generada automáticamente con confianza media

Imagen 10. Ruido gaussiano. [vi](#vi)

Este ruido podemos reducirlo aplicando un filtro gaussiano (Imagen 12). Que consiste en aplicar una máscara similar al filtro de media (Imagen 11).

Calendario

Descripción generada automáticamente con confianza media

Imagen 11. Máscara de filtro gaussiano. [vi](#vi)

Imagen que contiene foto, motor, cocina, hidrante

Descripción generada automáticamente

Imagen 12. Filtro gaussiano. [vi](#vi)

### Ruido Impulsivo

Por otro lado, tenemos el ruido impulsivo, que se produce normalmente en la cuantificación que se realiza en el proceso de digitalización.

Imagen que contiene firmar, computadora, reloj

Descripción generada automáticamente

Imagen 13. Ruido impulsivo. [vi](#vi)

Para reducir este ruido podemos aplicar el filtro de la mediana (Imagen 14), que consiste en aplicar la mediana para cada píxel de sus píxeles vecinos.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen 14. Filtro de la mediana. [vi](#vi)

### Aumento de Datos

Por otro lado, tenemos el aumento de datos (data augmented). Actualmente, uno de los principales problemas a la hora de entrenar cualquier red neuronal es la falta de datos. Para entrenar estos modelos es necesario disponer de un mínimo de datos que, normalmente, cuesta obtener.

Para esto se aplica este preprocesado, el cual consiste en generar una mayor cantidad de datos de los que tenemos en nuestro dataset original. Para esto, en el caso de las imágenes, podemos aplicar rotaciones, zooms, volteos y/o desplazamientos.



Imagen 15. Aumento de datos. [v](#v)

## Overfitting y underfitting

Lo primero en este punto es entender qué es el overfitting y el undedrfitting. El underfitting consiste en entrenar el modelo muy poco tiempo de forma que no le da tiempo a aprender, mientras el overfitting es lo contrario, es decir, entrenar demasiado tiempo el modelo con los mismos datos de tal forma que en lugar de aprender de estos datos los memoriza, por lo que al utilizar el modelo con datos nuevos no sabrá cómo clasificarlos.

El overfitting puede detectarse observando la gráfica del accuracy (Imagen 16) y comparando el accuracy de los datos de entrenamiento con los datos de test.

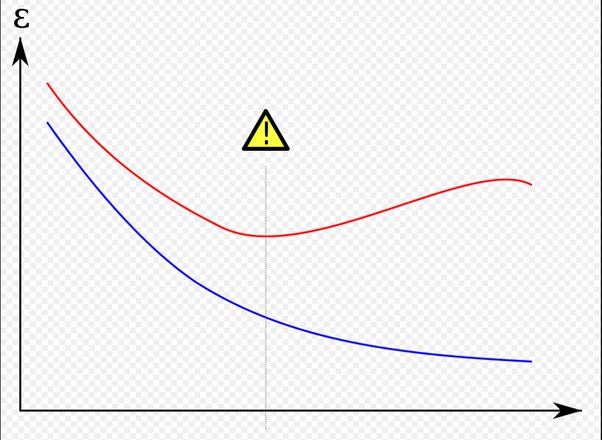


Imagen 16. Detectando el overfitting mediante el accuracy. La línea roja es el accuracy del conjunto de test mientras la línea azul es el accuracy del conjunto de entrenamiento. [vii](#vii)

Para solucionar esto tenemos varios métodos:

* **Aumento de datos**: Explicado en el punto 5.3.
* **Dropout**: Este método consiste en, inutilizar un porcentaje de las neuronas de la red forzando al resto de neuronas a aprender para suplir la falta de las neuronas desconectadas.
* **Early Stopping**: Consiste en parar el entrenamiento tras varias épocas en las que baja el accuracy obtenido con el conjunto de test, ya que esto implica que comienza a haber overfitting.

# Proyectos Relacionados

Hoy en día, la tarea de reconocimiento de imágenes y vídeos está cobrando una gran importancia. La razón de esto es su aplicabilidad en muchos campos y entornos, como ayudar a prevenir peligrosos accidentes de tráfico en carreteras concurridas. Por esto aparecieron muchos estudios y proyectos que propusieron soluciones a este problema en diferentes campos de aplicación. En las próximas secciones revisamos en detalle algunos de los proyectos más interesantes.

## Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction

## Estimating Urban Traffic Density Using Street Camera Images

## Boosting Multi-Vehicle Tracking with a Joint Object Detection and Viewpoint Estimation Sensor

# MATERIALES Y MÉTODOS

*En este capítulo, presentaremos los materiales que hemos utilizado para este proyecto. Significa qué marcos, conjuntos de datos, herramientas de hardware y software hemos elegido y el motivo de esta selección.*

## Frameworks

Para llevar a cabo este estudio ha sido imprescindible el uso de frameworks, sin estos no sería posible. Por esa razón, los detallamos en las siguientes secciones.

### Keras

Por un lado, hemos utilizado “Keras” [2]. Es un importante y poderoso framework destinado al Aprendizaje Automático. Hemos seleccionado "Keras" porque es muy popular, flexible y habíamos trabajado con él antes. Este framework permite crear modelos lineales añadiendo capas como una secuencia y es con el que hemos desarrollado nuestra red convolucional.

### PyTorch

PyTorch [17] se ha convertido en una herramienta popular en la comunidad de investigación de aprendizaje profundo por combinar un enfoque en la usabilidad con cuidadosas consideraciones de rendimiento, este framework podría haber sido una buena opción debido a su potencia y flexibilidad. PyTorch es un framework de aprendizaje autotmático de código abierto para Python, Java y C++ programado en C++, CUDA y Python. Su desarrollo ha sido llevado a cabo principalmente por Facebook.

### YOLOv3

Por otro lado, tenemos que mencionar otra importante herramienta, ”YOLOv3” [19]. Nuestro proyecto consiste en estimar el tráfico basado en frames recopilados por una cámara urbana en tiempo real.

Por tanto, una de las primeras propuestas pensadas fue detectar cada uno de los vehículos existentes en el frame y, en función del número calculado, devolver una mayor o menor cantidad de tráfico estimado.

De haber llevado a cabo esta opción habría sido imprescindible un detector de objetos rápido. Comparamos YOLOv3 con otros detectores de objetos (Figura 3.1), como RetinaNet-50 y RetinaNet-101, y YOLOv3 obtiene los mejores tiempos sin perder precisión. Así que este detector es una buena opción para esta propuesta.

Esta idea se desprecia debido a la complicación que supone relacionar el número de vehículos con la extensión captada de la carretera.

Interfaz de usuario gráfica

Descripción generada automáticamente

## Datasets

Para nuestro estudio, hemos comparado muchos conjuntos de datos (Figura 3.2) y hemos elegido los mejores para nuestro proyecto. En las siguientes secciones, los revisamos y describimos brevemente.

### Big Data for Traffic Estimation and Prediction

### Cameras Florida

“Cameras Florida” consiste en un dataset que puede ser de gran utilidad en nuestro proyecto. Este dataset contiene los registros de 3.668 cámaras ubicadas en Florida, por lo que es una importante fuente de datos de la cual podríamos obtener varios frames de cada una de las cámaras para llevar a cabo el entrenamiento de nuestro modelo.

Este dataset no lo hemos elegido debido a la complicación de etiquetar esa cantidad de datos con el tiempo del que disponemos, pero es una buena opción para futuras ampliaciones del proyecto.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Al igual que para Florida podemos obtener datasets de otros estados de Estados Unidos de páginas similares, por lo que la cantidad de datos disponible es inmensa.

### Traffic Congestion Dataset

Este dataset consiste en 4.238 frames de cámaras urbanas de 500 x 500 etiquetadas en función de si encontramos una gran congestión en la carretera o no.

Este es el dataset que hemos elegido debido a la comodidad que nos proporciona el tener todos los frames ya etiquetados.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

### Photole and Plain Road Dataset

Este dataset incluye 370 imágenes de diferentes carreteras en diferentes estados. Tras el análisis del dataset podemos comprobar que ninguna de ellas incluye tráfico. Por tanto, podría ser de gran utilidad como complemento a algún dataset desbalanceado, es decir, con una mayor cantidad de imágenes de carreteras congestionadas.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## Hardware

En cuanto al hardware se han utilizado máquinas con recursos limitados, lo que ha supuesto una limitación a la hora de entrenar nuestro modelo y obtener resultados.

### Google Colab

Para este proyecto hemos aprovechado los recursos que nos ofrece Google de manera gratuita haciendo uso de su GPU para agilizar los cálculos. Estos tienen las siguientes propiedades:

* RAM: 12.68 GB
* Disco: 78.19 GB

### Rog Strix

Además de los recursos ofrecidos por Google Colab también se han utilizado los recursos del ordenador persona Rog Strix que consiste en:

* RAM: 16 GB
* Disco: 0.5 TB
* Procesador: AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz
* GPU: Geforce GTX

## Otras herramientas

### GitHub

Para gestionar las versiones hemos hecho uso de GitHub, un gestor de versiones muy popular en los últimos años debido a la comodidad y facilidad de uso.

En estos repositorios hemos guardado cada uno de los cambios realizados tanto en el código como en la documentación.

### ResNet50

### SHAP

Diagrama

Descripción generada automáticamente con confianza baja

SHAP (SHapley Additive exPlanations) es un enfoque teórico utilizado para explicar el resultado de cualquier modelo de aprendizaje automático (Figura X). Conecta la asignación óptima de créditos con explicaciones locales utilizando los valores clásicos de Shapley de la teoría de juegos y sus extensiones relacionadas.

Esta herramienta nos será muy útil para poder estudiar el funcionamiento de nuestro modelo y si éste funciona como esperamos.

# ARQUITECTURA

## Método propuesto 1

El primer método propuesto consiste en hacer uso de YOLOv3, un framework que detecta los objetos existentes en una imagen a tiempo real.

Este método simplemente trata de contar el número de vehículos que encontramos en una imagen y, en función de si es un número grande o pequeño, calcular si la carretera de la imagen está congestionada o no.

Sin embargo, nos encontramos con que el método tiene un problema añadido. No todas la imágenes están tomadas del mismo punto de visita ni son de la misma carretera, lo que implica que, para realizar el cálculo, debemos tener en cuenta la extensión de la carretera captada por la cámara. Por esto decidimos buscar otra solución más sencilla.

## Método propuesto 2

Tras la complicación que supone el primer método propuesto pensamos que, a veces, lo más sencillo es lo más eficaz. Por tanto, decidimos implementar una red neuronal y ver el resultado que obtenemos.

### Ingesta de datos

Antes de empezar a implementar el modelo debemos comenzar cargando los datos, para lo que utilizaremos Keras.

Este apartado es simple pero imprescindible, ya que los datos del dataset elegido no tenían un formato compatible con Keras, por lo que, antes de cargar los datos, hemos tenido que adaptar el formato.

Keras obtiene el etiquetado en función de la jerarquía de carpetas, por lo que hemos tenido que redefinir su estructura en función del etiquetado de cada imagen de la siguiente forma:

### Preprocesado

Al implementar la red nos damos cuenta de que el tamaño de las imágenes (500 x 500) puede darnos 2 problemas principales:

* Reducirá la velocidad de entrenamiento y de predicción de la red considerablemente.
* Necesitaremos de un modelo mucho más complejo, lo que implica una mayor cantidad de datos necesaria para el entrenamiento del mismo.

Por tanto, al cargar los datos redimensionamos estas imágenes a 32 x 32. Es decir, reduciremos los píxeles de 250.000 a 1.024.

Una carretera con coches y edificios de fondo

Descripción generada automáticamente

Además, normalizamos el valor de los píxeles, de tal forma que, en lugar de valores entre 0-255, tengan valores entre 0-1.

### Aumentado de datos

Actualmente, tenemos 2.119 imágenes de cada tipo en nuestro dataset, pero podemos obtener una mayor cantidad de datos aumentandolos. Esto nos será muy útil para reducir el overffitting y, de esta forma, poder entrenar durante más tiempo la red logrando que obtenga una mayor precisión.

Esta técnica consiste en hacer pequeñas modificaciones en cada una de las fotos como realizar zoom, volteos, desplazamientos… De esta forma obtendremos una mayor cantidad de datos y nuestro modelo será más efectivo.

Imagen borrosa de un edificio

Descripción generada automáticamente con confianza mediaImagen borrosa de una persona

Descripción generada automáticamente con confianza baja

### Arquitectura de la red

Para obtener la arquitectura final del proyecto hemos probado 2 opciones. Por un lado hemos implementado una CNN desde 0 como baseline y, por otro lado, hemos aprovechado una red ya entrenada (ResNet50) para adaptarla a nuestro código.

#### Red sin preentrenar

Diagrama

Descripción generada automáticamente

Como observamos en la imagen X, nuestra red, como cualquier red convolucional, consta de 2 partes principales: La extracción de características y su estudio. (Anexo X)

**Extracción de características**

Diagrama

Descripción generada automáticamente

Esta primera parte se compone de 3 capas de convolución de 3x3. En estas capas se utiliza una función de activación Relu ya que, actualmente, es la que mejor rendimiento obtiene en la gran mayoría de capas intermedias de una red. Por otro lado, hemos utilizado un padding, lo que significa que se añaden píxeles alrededor de la imagen para que, al realizar la convolución, no se reduzca el tamaño.

Por otro lado, estas capas convolucionales se intercalan con capa de pooling de 2x2, cuya función, como se ha comentado en puntos anteriores, es resumir las características obtenidas.

**Estudio de las características**

Diagrama

Descripción generada automáticamente

Tras finalizar esta primera parte, utilizamos las características obtenidas como entrada a una red multiperceptrón. Esta red se compone de 4 capas intermedias de 512, 256, 64 y 32 neuronas y en las que también hemos utilizado la función Relu como función de activación.

Por último, utilizamos 1 única neurona que nos proporcionará el resultado en formato binario (0 ó 1). Para esta última neurona utilizamos la función de activación sigmoide, ya que esta función nos ofrece resultados de 0 a 1 de forma proporcional.

Además, se han hecho pruebas añadiendo capas de dropout (explicadas en el punto X) para ver si mejoraba el rendimiento. Sin embargo, lo único que se ha conseguido es que la red pierda mucha estabilidad en el entrenamiento sin mejorar el rendimiento.

#### Red preentrenada

Otro modelo que hemos probado ha sido haciendo uso de la red preentrenada “ResNet50” (Anexo X), un modelo explicado en el punto X. Simplemente, hemos añadido una capa de salida de 1 única neurona y con la función de activación sigmoide para adaptarlo a nuestras necesidades.

## Experimentación

### Overfitting

Como hemos comentado en capítulos anteriores el overfitting es un problema muy común en el entrenamiento de modelos. Es por esto que hemos prestado especial atención en detectarlo y evitarlo. Para esto, por un lado, hemos monitorizado cómo evoluciona tanto el accuracy como el loss tanto del conjunto de entrenamiento como del conjunto de validación.

Gráfico, Gráfico de líneas, Histograma

Descripción generada automáticamente

En la Figura X podemos observar como el accuracy de ambos conjuntos se mantienen similares, lo que implica que no está memorizando los datos del conjunto de entrenamiento y, por tanto, no existe overfitting.

Por otro lado, comprobamos que el loss del conjunto de validación no tiene una tendencia a aumentar en ningún momento. Esto es otra señal de que el entrenamiento está funcionando correctamente sin sobreajustar los parámetros.

Por otro lado, hemos hecho uso del early\_stopping, que pone el foco en el loss del conjunto de validación y, si comienza a empeorar, para automáticamente el entrenamiento.

### Funcionamiento de la red

A pesar de todas estas comprobaciones los resultados del modelo podrían no tener una fiabilidad del 100%. Es decir, si los datos utilizados contienen algún sesgo que diferencie ambas clases (con tráfico y sin tráfico) el modelo podría estar aprendiendo a diferenciarlos por este sesgo. Es decir, supongamos que en el conjunto de datos las imágenes con tráfico siempre se encuentran en una misma carretera y las imágenes sin tráfico se encuentran en otra distinta. En este caso, el modelo podría estar aprendiendo a diferenciar las imágenes en función de la carretera captada en lugar de prestando atención al tráfico que encontramos en la imagen.

Para asegurarnos de que esto no ocurre veremos en que está poniendo el foco nuestro modelo con un subconjunto de datos. Para esto haremos uso de la herramienta SHAP (explicada en el punto X), una herramienta que nos facilitará esta tarea.

Un conjunto de imágenes de una persona

Descripción generada automáticamente con confianza mediaUn conjunto de imágenes de una persona

Descripción generada automáticamente con confianza media

Un conjunto de imágenes de una persona

Descripción generada automáticamente con confianza media

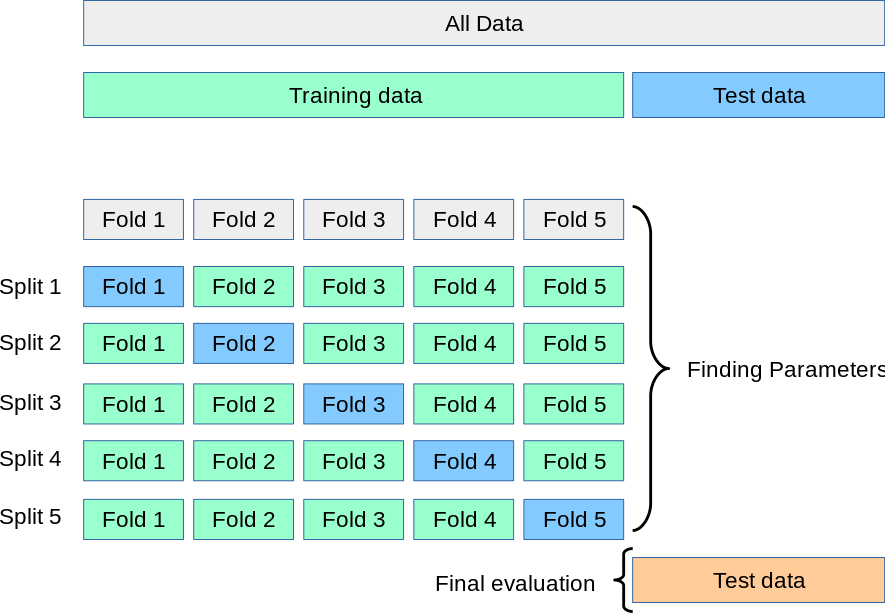
Como vemos en la Figura X, esta herramienta pinta cada píxel de un color: Rojo si el píxel decanta la solución hacia la clase “Con tráfico” y azul si la decanta hacia la clase “Sin tráfico”.

Con estas imágenes podemos comprobar que, efectivamente, nuestro modelo está poniendo el foco de atención en los vehículos para clasificar la imagen.

### Comparación de modelos

Al tener 2 modelos distintos queremos comprobar cuál de los 2 obtiene mejores resultados. Para esto necesitaremos utilizar unas métricas fiables. Podríamos utilizar el accuracy, pero esta métrica es fiable únicamente si los datos están perfectamente equilibrado, por eso suele utilizarse el AUC, es decir, el área cubierta por la curva ROC. Este parámetro sí tiene en cuenta el posible desequilibrio de datos y, por tanto, es una métrica mucho más fiable.

Una vez decidido el parámetro podemos tener otro problema, y es el problema de la aleatoriedad. Es decir, podría darse que un modelo peor obtenga mejores resultados simplemente porque los datos se hayan distribuido a su favor mejorando, de forma excepcional, su resultado. Es por esto que normalmente se utiliza el “cross validation” o “validación cruzada”, simplemente consiste en generar varios subconjuntos y entrenar el modelo varias veces utilizando un subconjunto diferente como conjunto de validación en cada una de las iteraciones, como se muestra en la Figura X.



<https://scikit-learn.org/stable/modules/cross_validation.html>

Una vez tenemos todo esto, comprobamos cuál de los 2 modelos obtiene mejores resultados mediante el modelo de Wilcoxon. Este modelo con los valores AUC de las iteraciones del cross validation (Imagen X) calcula la probabilidad de que uno de los modelos sea más eficaz que el otro.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Como observamos en la imagen X, el valor de p-value es 1, lo que implica que el modelo sin preentrenar tiene una probabilidad del 100% de obtener mejores resultados que el modelo preentrenado, por lo que este será nuestro modelo final.

## Resultados

Por último, utilizaremos el conjunto de test para obtener el rendimiento de nuestro modelo final.

Accuracy: 94%

Loss: 0.19

AUC: 0.9888

Como podemos observar, nuestro modelo únicamente falla 1 imagen de cada 20, asumiendo que los datos pueden tener imágenes difíciles de clasificar y teniendo en cuenta la escasez de datos, tiempo y material es un resultado muy bueno para el objetivo que persigue el proyecto.

# CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO

En este proyecto se han estudiado técnicas muy importantes para el reconocimiento de imágenes. Se ha comenzado con un preprocesado de los datos reduciendo el tamaño y normalizando los valores para, más tarde generar más datos haciendo aumentado de datos.

Una vez tenemos los datos, hemos implementado y entrenado el modelo. Por último, hemos comprobado el funcionamiento del modelo y hemos obtenido el rendimiento de nuestro modelo.

Este rendimiento ha sido muy bueno teniendo en cuenta la escasez de datos, tiempo y material del que disponemos.

Podemos concluir que, para la estimación de tráfico, podemos obtener muy buenos resultados haciendo uso de las redes neuronales convolucionales. Sin embargo, estos resultados todavía pueden mejorarse de varias maneras.

Una propuesta para un futuro trabajo es obtener los datos de las cámaras urbanas de varios estados de EEUU y etiquetarlos ayudándonos del clustering. Una vez tengamos los datos, dispondremos de una cantidad inmensa que implicarán una importante mejora del rendimiento.

Otra propuesta sería ayudarnos de un modelo como YOLOv3 para terminar de afinar el resultado, por ejemplo si en una imagen clasificada como “Con tráfico” no se detecta ningún vehículo seguramente no estará bien etiquetada.

# ANEXOS

def cnn\_model2():

  visible = layers.Input(shape=X[0].shape)

  x = layers.Conv2D(16,(3,3),padding="same",activation="relu")(visible)

  x = layers.MaxPooling2D((2,2))(x)

  x = layers.Conv2D(32,(3,3),padding="same",activation="relu")(x)

  x = layers.MaxPooling2D((2,2))(x)

  x = layers.Conv2D(64,(3,3),padding="same",activation="relu")(x)

  x = layers.MaxPooling2D((2,2))(x)

  x = layers.Flatten()(x)

  x = layers.Dense(512,activation="relu")(x)

  x = layers.Dense(256,activation="relu")(x)

  x = layers.Dense(64,activation="relu")(x)

  x = layers.Dense(32,activation="relu")(x)

  output = layers.Dense(1, activation='sigmoid')(x)

  model = keras.Model(inputs=visible, outputs=output)

  return model

def cnn\_model():

  #Descargamos el modelo

  URL = 'https://tfhub.dev/google/imagenet/resnet\_v2\_50/feature\_vector/4'

  feature\_extractor =  hub.KerasLayer(

      URL,

      input\_shape=X[0].shape

  )

  #Bloqueamos el entrenamiento de los pesos ya entrenados de esta red

  feature\_extractor.trainable=False

  #Adaptamos la red a nuestras necesidades

  model = tf.keras.models.Sequential([

    feature\_extractor,

    tf.keras.layers.Dense(1,activation='sigmoid')

  ])

  return model

# PRESUPUESTO

Si se trata de un proyecto siempre debe incluir este apartado.

El Presupuesto supone la evaluación económica total del proyecto.

No olvides que tu tiempo también vale dinero, no sólo hay que incluir el coste de los materiales empleados.

# BIBLIOGRAFÍA

* i “Understanding of Convolutional Neural Network (CNN) — Deep Learning.” *Medium*, 4 March 2018, [https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. Accessed 31 July 2022](https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.%20Accessed%2031%20July%202022).
* ii Pereira, Jonathan. “Perceptrón Multicapa y Algoritmo de Retropropagación - Artículos sobre MQL5.” *MQL5*, <https://www.mql5.com/es/articles/8908> . Accessed 31 July 2022.
* iii Bagnato, Juan Ignacio. “Convolutional Neural Networks: La Teoría explicada en Español.” *Aprende Machine Learning*, 29 November 2018, <https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/> . Accessed 31 July 2022.
* iv *Tema 8. Redes Neuronales*, <http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t8neuronales.pdf> . Accessed 31 July 2022.
* v Gandhi, Arun. “Data Augmentation | How to use Deep Learning when you have Limited Data — Part 2.” *Nanonets*, 19 May 2021, <https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/> . Accessed 31 July 2022.
* vi *Tema 3: Filtros*, <http://alojamientos.us.es/gtocoma/pid/tema3-1.pdf> . Accessed 31 July 2022.
* vii “The train accuracy of a detection model (supervised, NN) after 10 epochs is 100% whereas the test accuracy is only 82%. Am I overfitting the model?” *Quora*, <https://www.quora.com/The-train-accuracy-of-a-detection-model-supervised-NN-after-10-epochs-is-100-whereas-the-test-accuracy-is-only-82-Am-I-overfitting-the-model> . Accessed 31 July 2022.