Clasificación Señales Trafico

Sergi Díaz Castro

Resumen

Este trabajo se ha centrado en el estudio de uno de los módulos de la conducción autónoma, la predicción de señales de tráfico. Para ello se ha desarrollado una aplicación que permita detectar y clasificar señales de tráfico en tiempo real mediante el uso de visión por computador. La aplicación usando entrada por video y una red neuronal previamente entrenada es capaz de generar salidas prediciendo la señal que se muestra en la imagen.

Para ello, a lo largo del proyecto, se han entrenado redes neuronales con características cambiantes, para poder aprender sobre su comportamiento. Extrayendo conclusiones y resultados sobre los datos extraídos de las redes neuronales entrenadas, ha permitido generar un modelo sólido funcional que realiza predicciones en tiempo real.

Palabras clave

Señales de tráfico, inteligencia artificial, red neuronal, conducción autónoma, visión por computador, predicción, convolución.

Abstract

This work has focused on the study of one of the modules of autonomous driving, traffic sign prediction. To this end, an application has been developed to detect and classify traffic signs in real time using computer vision. The application using video input and a previously trained neural network is able to generate outputs predicting the signal shown in the image.

To this end, throughout the project, neural networks with changing characteristics have been trained in order to learn about their behaviour. By extracting conclusions and results from the data extracted from the trained neural networks, it has been possible to generate a solid functional model that makes predictions in real time.

Index Terms

Traffic signals, artificial intelligence, neural network, autonomous driving, computer vision, prediction, convolution.

1 Introducción - Contexto del trabajo

Actualmente, unos de los puntos clave de Ingeniería, portado con la rama de Computación, son los sistemas de visión artificial. Estos sistemas se encargan de analizar imágenes del mundo real para poder procesarlas y generar modelos que puedan ser interpretados por computadores. Esta tecnología está en plena expansión y actualmente la podemos encontrar en el sector de la alimentación, donde estos sistemas se encargan de él control de la calidad de los alimentos, en el sector de la electrónica, donde se encargan de la soldadura de componentes microscópicos, entre otros usos [1].

Un sector que está invirtiendo una gran cantidad de dinero y esfuerzo en la mejora de los sistemas de visión por computador, es el sector de la automoción. Donde aparte de usar esta tecnología pasa ensamblar vehículos, se desea poder implementar en la generación de conducción autónoma. Para llegar a ese punto se necesitan muchos módulos funcionando coordinada y precisamente, debido a que es un tema muy sensible, ja que los accidentes de tráfico es una de las principales causas de mortalidad.

Uno de los módulos que se deben implementar está relacionado con la percepción del entorno que tiene el vehículo, y una de sus funcionalidades es la detección y reconocimiento de señales de trafico en tiempo real. Este módulo se ha estado implementando por los principales fabricantes de vehículos en los últimos años, pero enfocado en proporcionar al conductor una mayor comodidad al volante.

Actualmente, se está trabajando en mejorar este módulo, debido a que es uno de los más importantes a la hora de implementar la conducción autónoma, por el impacto que tiene tanto en mejora de seguridad, reducción de errores debido al factor humano y reducción de costes en combustible.

En este trabajo se abordará en primer lugar cuál es el estado del arte para sistemas de clasificación de señales y conducción autónoma. Todo ello, para conocer el cómo y

- *E-mail de contacto: 1489852@uab.cat*
- *Mención realizada: Computación*
- *Trebajo tutorizado por: Xavier Otatzu Porter*
- *Curso 2021/22*

por qué surgieron este tipo de técnicas. Después, se explicará la planificación inicial. A continuación, la metodologia que se ha utilizado para la realización de este trabajo. Seguido, se explicarán las implementaciones y resultados tanto en métricas como en imágenes obtenidas sobre la aplicación final. Para concluir se añadirán futuras mejoras e implementaciones.

2 OBJETIVOS

La idea inicial que me motivo a realizar el proyecto era realizar un coche que pudiera realizar una conducción autónoma. Obviamente, esta propuesta escapa a las dimensiones de un trabajo de final de grado, por recursos, tiempo y complejidad. Por este motivo he buscado una alternativa que me permita obtener conocimientos sobre ese tipo de proyecto y que se adapte a un proyecto de final de carrera. Para ello mi idea es realizar un clasificador de señales utilizando redes neuronales.

Por lo tanto, los objetivos principales que me han motivado a realizar el trabajo sobre este tema son:

- Entender y entrenar redes neuronales y estudiar cómo obtener mejores resultados.
- Generar un algoritmo para poder detectar y classificar las señales de tráfico en tiempo real.
- Estudiar el comportamiento de las redes neuronales en entornos controlados.
- Estudiar el funcionamiento de la red neuronal en entornos sin controlar y en tiempo real

3 ESTADO DEL ARTE

En este apartado veremos el estado actual tanto de la conducción autónoma como de las redes neuronales.

3.1 Conducción autónoma

En la actualidad, algunos fabricantes importantes de atomóviles como Mercedes, Honda o Toyota, entre otros, están lanzando al mercado vehículos que producen una conducción semi-autónoma. Estos son capaces de aparcar solos, mantenerse dentro de la trazada siguiendo líneas o incluso acelerar o decelerar al encontrar obstáculos en sus proximidades, pero todo esto realizado en condiciones muy favorables para el módulo encargado de este trabajo [2].

Por ello no se puede considerar conducción autónoma hasta que se pueda fabricar un vehículo que no tenga necesidad de tener ningún módulo controlado por un humano, es decir, sin volante, sin pedales, sin cambio de marchas, etc. Esta tecnología en los próximos años esta pensada para implementarse en proveedores de servicios comerciales que puedan permitirse el elevado coste de lanzamiento. Por lo tanto, la previsión es en los próximos cinco o diez años poder haber lanzado camiones que puedan transportar materiales por autovías sin necesidad de conductor. O hasta esta presente la posibilidad de poder tener por la ciudad circulando taxis y buses sin necesidad de obtener un conductor.

Entonces puede aparecer la siguiente pregunta: ¿Por qué no se han lanzado al mercado ya vehículos con conducción autónoma?

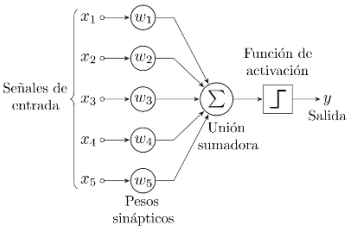
La respuesta la encontramos en la percepción del entorno de los vehículos, es decir, en los módulos de visión por computador. Aunque se podrían lanzar actualmente al mercado coches con conducción autónoma, habría problemas con la detección de entornos, como señales de tráfico en condiciones desfavorables o en ángulos muertos que no pueda llegar a percibir el coche, o llegar a perder la noción de los vehículos y/u obstáculos que le rodean en condiciones meteorológicas cambiantes. Por lo tanto, los atomóviles tendrían una inteligencia artificial no confiable, la cual podría provocar accidentes catastróficos.

3.2 Redes neuronales

Desde hace 70 años, los estudios sobre el funcionamiento del cerebro han ido en aumento en un intento de poder simular su comportamiento y reproducir su inteligencia. Rosenblatt en el 1958 implemento el primero modelo de perceptrón, el cual provee una salida ejecutando tres operaciones simples sobre unos datos de entrada obtenidos:

- Paso 1: Multiplicar las entradas por los pesos correspondientes.
- Paso 2: Sumar el resultado de todos los productos anteriores.
- Paso 3: Comparar la suma obtenida con el umbral provisto por la función de activación.

Así fue posible realizar la primera aproximación al funcionamiento del cerebro humano [3].

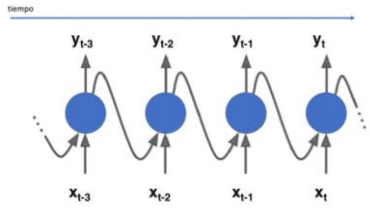


Más adelante aparecieron otros tipos de redes neuronales, que son los que se utilizan en la actualidad. Algunas de las más importantes són:

- Redes Neuronales Recurrentes (RNN)

Las redes neuronales recurrentes son redes inventadas en el 1980, que son capaces de influir en sí mismas por medio de recurrencias, es decir, no solo avanzan capas hacia adelante, sino que lo pueden realizar hacia atrás. Esto provoca que sean redes muy complejas de entrenar y suelen ser utilizadas para analizar datos de series de en-

tradas. En la siguiente figura podemos observar su comportamiento.

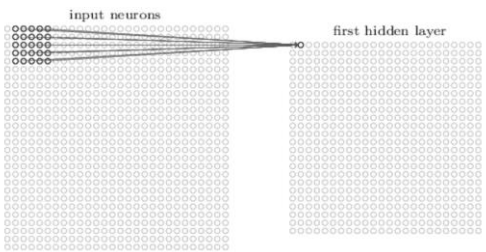


- Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales son un tipo de red bio-inspirado que simula la forma en que funciona la visión del ser humano, es una modificación del perceptrón multicapa. Estas redes neuronales son las utilizadas en el ámbito de visión por computador y, por lo tanto, las que utilizaremos en este proyecto.

El nombre de "Red Neuronal de Convolución" indica que la red emplea una operación matemática llamada convolución. La convolución es un tipo especializado de operación lineal. Las CNN explotan la correlación espacial local mediante la aplicación de un patrón de conectividad local entre las neuronas de las capas adyacentes. Estas generan salidas en cada una de sus capas a partir de entradas de la capa anterior, este proceso es continuo hasta llegar a la capa final donde obtienes el resultado deseado.

A continuación, podemos observar la iteración de la capa de entrada con la primera capa oculta de la red neuronal.



4 PLANIFICACIÓN

Para poder llegar a los objetivos establecidos previamente, se generó una planificación inicial que ha ido variando mínimamente según iba avanzando el proyecto y se iban estudiando los resultados.

El primer paso para empezar un trabajo de final de grado siempre es buscar documentación sobre los temas que se van a tratar, eso se realizó inicialmente, buscando información de visión por computador, redes neuronales y como codificarlas.

Después de buscar la información se debía buscar una base de datos ja existente con imágenes de señales de tráfico previamente clasificadas, y que tuviera apartado de Train y de Prueba. La base de datos no se decidió no generar desde cero debido al costoso y longevo proceso que supone.

En tercer lugar, se debía preparar las imágenes para poder proporcionar las entradas adecuadas a las redes neuronales. A continuación, entrenar las tres redes neuronales con diferentes características.

Una vez entrenadas un punto clave a realizar es comparar el funcionamiento de las redes neuronales utilizando una métrica de precisión sobre el conjunto de Prueba.

Para acabar de verificar su funcionamiento será necesario comprobar el porcentaje de las redes detectando fotografías de señales de tráfico con el fondo blanco, es decir, en un entorno controlado.

Finalmente, para concluir el proyecto se generará una aplicación móvil para ver los resultados de las redes neuronales en tiempo real.

5 METODOLOGIA

En este apartado se procederá a explicar la metodología utilizada.

El lenguaje de programación empleado, la base de datos, las redes neuronales y la aplicación móvil.

5.1 Lenguaje de programación

Para toda la generación de código se ha utilizado Python 3.9.7 sobre una arquitectura Windows y utilizando Jupyter Notebook como editor.

Sobre Python se han usado las librerías tensorflow, matplotlib, os y skimage.

- Tensorflow: La principal biblioteca de código abierto para enseñarte a desarrollar y entrenar modelos de IA. Esta librería me ha permitido generar, entrenar y probar las redes neuronales [4].

- Matplotlib: Es una biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python. Esta librería me ha permitido visualizar el estado de las imágenes en todo momento [5].

- Os: Esta biblioteca provee una manera versátil de usar funcionalidades dependientes del sistema operativo. Esta librería me ha permitido poder importar las imágenes de la base de datos desde carpetas almacenadas en el sistema operativo Windows [6].

- Skimage: La principal biblioteca de código abierto para enseñarte a desarrollar y entrenar modelos de IA. Esta librería me ha permitido generar, entrenar y probar las redes neuronales.

5.2 Base de datos

La base de datos no reconocerá todas las señales, ya que será imposible atacar todas ellas debido a la gran variedad de señales que existen y las dimensiones del proyecto.

Después de realizar una búsqueda, la base de datos escogida es "Belgium TS Dataset" [8].

La base de datos dispone de dos partes, Train y Test, las cuales tienen 62 carpetas con imágenes dentro. Cada carpeta contiene una clase de imágenes, por lo tanto, se

clasifican 62 tipos de señales diferentes. La base de datos contiene 7095 que se dividen en un 64 por ciento de Train con 4575 imágenes y un 36 por ciento de Test con 2520.

```
Number of images Test: 2520
Number of images Train: 4575
Number of labels: 4575
Number of distinc traffic signals: 62
```

Las diferentes señales que se clasifican son las siguientes, podemos observar el número de clase, el número de imágenes y una de las imágenes:

5.3 Red neuronal Densa (Dense)

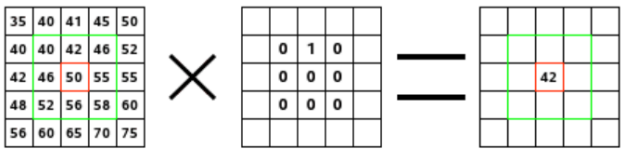
La primera red neuronal, bautizada como Dense, utiliza una capa de entrada Flatten, seguida por dos capas ocultas Dense con 150 neuronas cada una y con una activación relu. Finalmente, una capa de salida, la cual tiene el mismo número de neuronas que los resultados posibles de la red neuronal, es decir, 62. Esta última capa emplea una activación softmax la cual genera un número del 0 al 61 y el valor que tenga más próximo es el resultado de la clasificación.

El funcionamiento de esta red neuronal es simple, usa los píxeles de una fotografía para compararlo con los píxeles de otra y así poder clasificar las señales que observa.

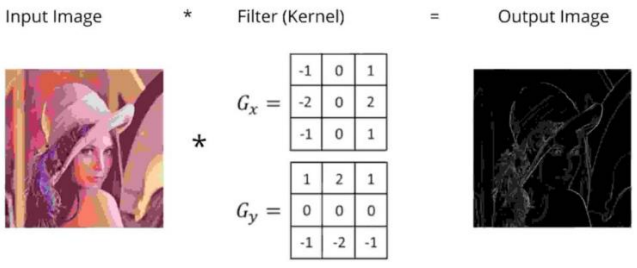
5.4 Red neuronal Convolucional (CNN)

La segunda red neuronal, llamada CNN, utiliza tres capas ocultas Conv2D que utiliza mascaras para poder tratar las imágenes y así poder generar desenfoques, afilamientos o detectar bordes. A continuación, emplea MaxPooligon2D para reducir el tamaño de la imagen y así realizar las entradas de la siguiente capa Conv2D con diferentes posiciones de la imagen. Después de realizar este proceso 3 veces se aplica una capa de entrada Flatten seguido de una Dense con 100 neuronas, con activación relu, imitando el funcionamiento de la red neuronal Dense pero habiendo realizado un tratamiento anterior. Finalmente, para obtener la salida se ha vuelto a aplicar softmax. El funcionamiento de esta red neuronal es aprenderse formas para después poder relacionarlas y poder clasificar las imágenes.

Para ver el funcionamiento de la convolución podemos observar estas dos imágenes a continuación. En la primera podemos observar los valores de los píxeles de una imagen, donde se le aplica un filtro con ponderaciones y obtenemos la imagen de salida con los píxeles.



En la segunda imagen podemos observar el funcionamiento, pero en vez de ver los valores de los píxeles podemos observar el cambio aplicado en la imagen a raíz de aplicar estos dos filtros.



5.5 Red neuronal Convolucional 2 (CNN2)

La tercera red neuronal, llamada CNN2. Tiene el mismo comportamiento que la CNN pero aplicando un DropOut para eliminar periódicamente algunas neuronas y variar su comportamiento a la hora de clasificar. También previamente se han tratado las imágenes para realizar data augment y obtener imágenes con diferentes dimensiones, torcidas o volteadas.

5.6 Aplicación Movil

Finalmente, se ha desarrollado una aplicación web y móvil. Para el desarrollo se ha utilizado lenguaje HTML y javascript combinado con css.

La aplicación carga las redes neuronales mencionadas previamente y clasifica las imágenes recibidas por una cámara en tiempo real.

A continuación, podemos visualizar la interfaz de la aplicación web.

6 IMPLEMENTACIÓN Y RESULTADOS

En este apartado se explicará la implementación de la metodología y los resultados obtenidos. También veremos como se han ido tomando las decisiones según se ha ido avanzando en el proyecto.

6.1 Tratado de imagenes

En este punto ya se había escogido la Base de Datos, "Belgium TS Dataset" y la teníamos descargada en local. También se había decidido que todo el código relacionado con el entrenamiento y los resultados de las redes neuronales se ejecutaría en Python.

Así que el siguiente paso, era empezar a trabajar con las imágenes para proporcionarle las características necesarias para poder completar el entrenamiento sin problemas.

Inicialmente, necesitábamos cargar las imágenes en el código, separándolas en "images_train", "labels_train", "images_test" y "labels_test". Dónde "images_train" e "images_test" son numpy arrays que almacenan matrices con los píxeles de las imágenes en cada posición, y "labels_test" y "labels_train" són numpy arrays que almacenan el número de clase de cada imagen.

Una vez las imágenes cargadas el siguiente paso ponerles el formato deseado y que todas sean iguales. Para ello, inicialmente se tiene que realizar un "resize" para hacer que las imágenes tengan 100 píxeles de altura y 100 píxeles de amplitud. El siguiente paso para la conversión de imágenes es transformarlas a blanco y negro para que los valores de los píxeles sean de 0 a 1 dependiendo la tonalidad de blanco y negro, así se eliminan los colores de las imágenes, lo que complicaría su clasificación debido a que los valores de los píxeles estarían acotados en el rango de 0 a 255. Después de ejecutar el proceso podemos observar su transformación respecto a la imagen del apartado 5.2.

6.2 Modelo Dense

El primer modelo con el que se ha empezado a trabajar es el modelo Dense explicado en el punto 5.3.

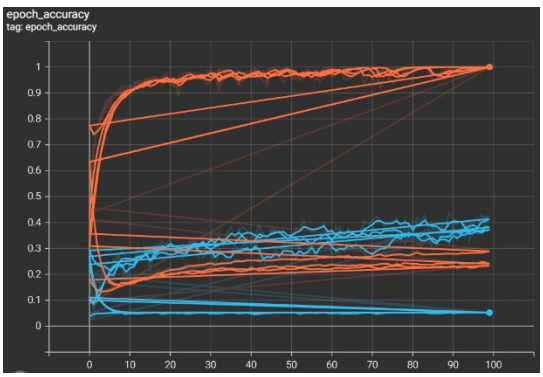
Después de crear el modelo con la función de Tensorflow Sequential donde se indica el comportamiento mencionado previamente con su capa de entrada Flattern, sus dos capas ocultas Dense con 150 neuronas y activación relu y finalmente su salida donde mediante Softmax genera el número en el que se clasifica la imagen.

A continuación, compilamos el modelo con la función compile y lo entrenamos con la función fit. En esta función le pasamos por parámetro las imágenes y las etiquetas con las que trabajara, es decir, "images_train" y "labels_train" y el numero de iteraciones que queremos que realice, marcado en el atributo epochs = 100.

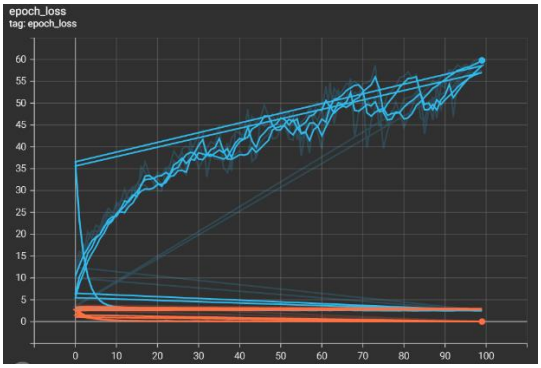
Una vez entrenada la red neuronal, encontramos una red con un accuracy muy alto, del 100%. Pero en verdad no podemos fiarnos de este valor, ya que la red se aprende los píxeles de las imágenes y después los compara con los píxeles de la imagen a clasificar y así obtiene la clasificación. Por lo tanto, en el momento que la red encuentra una imagen torcida o deformada no sabrá predecir el valor.

Para comprobar que la teoría es cierta vamos a visualizar las graficas a continuación.

En la primera gráfica podemos observar la precisión del modelo, a simple vista podemos ver como el conjunto de train va aumentando de 0 a 100% de precisión, pero, en cambio, en el de test el porcentaje va disminuyendo. Esto nos indica que efectivamente la red neuronal se aprende las imágenes del set de train para a la hora de clasificar nuevas obtendrá unos resultados malísimos. En el caso de las gráficas cuando hablamos de test no estamos trabajando con "images_test" sino que con un porcentaje del conjunto train escogido aleatoriamente.



En la segunda gráfica podemos ver la métrica loss, que nos indica la desviación entre las predicciones realizadas por la red neuronal y los valores reales de las observaciones utilizadas durante el aprendizaje. Cuanto más pequeño sea este valor, mejores resultados obtendremos. En esta gráfica podemos observar como el valor de loss del conjunto de train es muy grande y va en aumento, así confirmándonos doblemente los malos resultados de esta red neuronal.



6.3 Modelo CNN

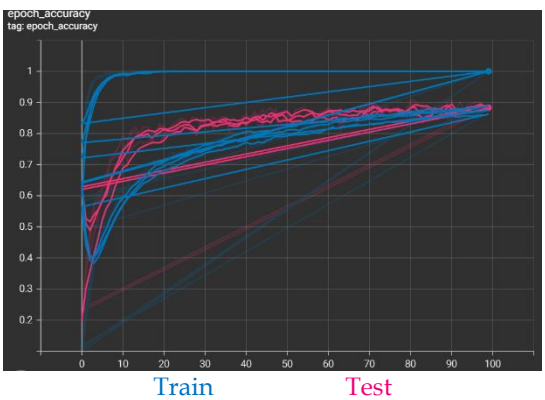
El segundo modelo desarrollado es el modelo CNN explicado en el punto 5.4.

Generamos el modelo con la función de Tensorflow Sequential donde se indica su comportamiento donde tenemos sus tres capas de entrada Conv2D seguidas cada una de ellas por una capa de entrada MaxPooling2D. Después de haber pasado por estas seis capas de entrada y haber tratado las imágenes, entra en su capa de entrada Flattern, después se aplica una capa oculta Dense con 100 neuronas y activación relu y finalmente su salida donde mediante Softmax genera el número en el que se clasifica la imagen.

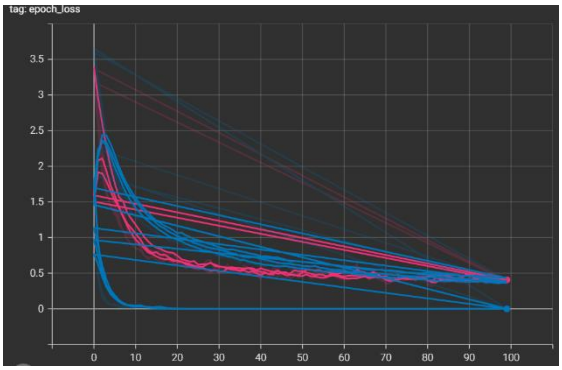
A continuación, compilamos el modelo con la función compile y lo entrenamos con la función fit. En esta función le pasamos por parámetro las imágenes y las etiquetas con las que trabajara, es decir "images_train" y "labels_train" y el número de iteraciones que queremos que realice, marcado en el atributo epochs = 100.

Después de entrenar el modelo, encontramos que esta sí que es capaz de diferenciar formas y así poder clasificar las imágenes según la relación de formas y llegar a tener un accuracy del 100%. A simple vista podemos observar que esta red neuronal será fiable y obtendrá bastante rendimiento. Pero no nos podemos quedar aquí, vamos a comprobar los resultados con las gráficas.

En la primera gráfica podemos observar como el accuracy de entrenamiento vuelve a ser del 100%, pero en este caso sí que encontramos que el acuraccy de test va en aumento y aunque no obtenemos un 100% se obtienen resultados del 90%.



De igual manera que en la gráfica de accuracy, podemos observar que en la de loss la desviación va disminuyendo y obtiene unos buenos valores.



6.4 Modelo CNN2

El tercer y ultimo modelo desarrollado es el modelo CNN2 explicado en el punto 5.5.

Generamos el modelo de la misma manera que el modelo CNN, pero en la capa oculta Dense se le aplican 250 neuronas con un Dropout del 0.5, con esto conseguiremos que en cada iteración se vayan activando y desactivando neuronas periódicamente. Finalmente, igual que en las otras dos, encontramos una capa de salida que clasifica la imagen en una de las 62 clases utilizando softmax.

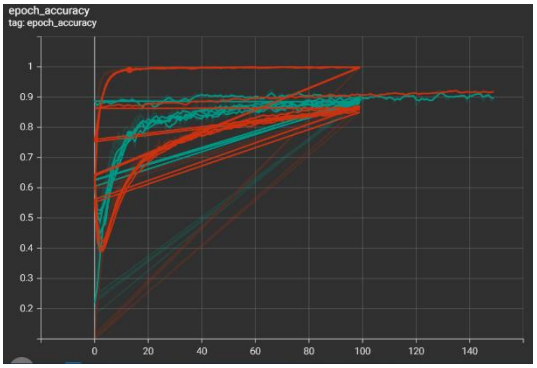
Antes de compilar el modelo, para esta red neuronal se han modificado las imágenes aplicando un filtro de data augment. Después de aplicar el filtro obtenemos imágenes como las siguientes.

A continuación, compilamos el modelo con la función compile y lo entrenamos con la función fit. En esta función le pasamos por parámetro las imágenes y las etiquetas con las que trabajara, es decir, "images_train" y "labels_train" y el numero de iteraciones que queremos que realice, marcado en el atributo epochs = 150.

Después de entrenar el modelo, al haber utilizado data augment, puede diferenciar más fácilmente las formas al haber obtenido los datos de imágenes modificadas. El problema de esta red es que obtenemos un accuracy del 85% debido a la diferencia entre las formas y la escasa cantidad de datos en la base de datos a entrenar. Si obtuviéramos una base de datos más grande o aumentáramos la que disponemos actualmente, esta red sería la mejor

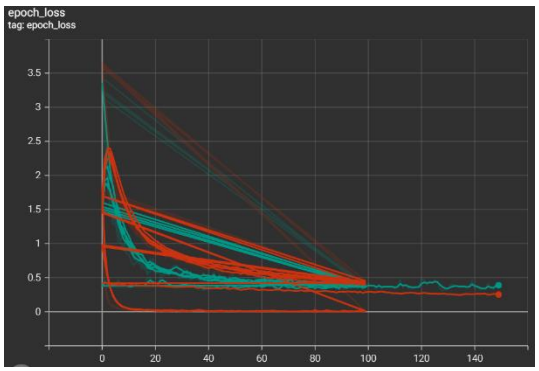
hasta el momento sin ningún tipo de dudas. Pero al haber escogido esta base de datos y trabajar con ella debemos realizar más pruebas en todos los modelos para decidir cuál es el mejor. También debemos observar las gráficas de este modelo.

En la primera gráfica podemos observar como train y test se comportan de la misma manera, ambos crecen en porcentaje hasta el 90%, esto nos provocara que esta red neuronal nunca nos aportara un 100% de precisión.



Train Test

De igual manera que en la gráfica de accuracy, podemos observar que en la de loss la desviación va disminuyendo y obtiene unos buenos valores, pero no llega a 0% que sería lo óptimo.



Train Test

6.5 Aplicación Web y Mobil

Al haber finalizado la implementación de los tres modelos se ha desarrollado una aplicación para poder obtener resultados en tiempo real. Para ello se han exportado los modelos e importados utilizando javascript.

La aplicación usa la cámara de un dispositivo móvil para grabar el exterior y en tiempo real transforma la imagen para obtener las características mencionadas en el punto 6.1. Al instante de transformarlas, las predice y devuelve un resultado en tiempo real. Para ejecutar la aplicación es necesario levantar un servidor para que pueda trabajar con las redes neuronales. Para ello se ha empleado el siguiente comando de Python en el directorio donde se encuentra el archivo index.html.

```
python -m http.server 4200
```

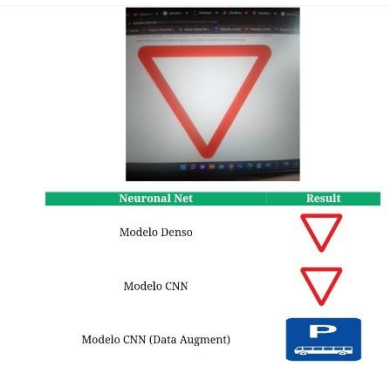
Esto nos permite acceder a la aplicación web en la url localhost:4200. Para poder acceder desde cualquier dispositivo sin tener el código en local, se ha utilizado ngrok, un programa para poder servir páginas web [9]. Entrando en la url cambiante que nos proporciona ngrok con cada ejecución podemos trabajar desde cualquier dispositivo con la aplicación.

```
ngrok

Session Status                online
Account                       sdiazc7@gmail.com (Plan: Free)
Update                        update available (version 3.0.4, Ctrl-U to update)
Version                       3.0.3
Region                        Europe (eu)
Latency                       42.0351ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://6cd4-80-174-219-216.eu.ngrok.io -> http://localhost:4200

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

A continuación, podemos observar una captura de la aplicación móvil prediciendo.



6.6 Comparar resultados

Visualizando y estudiando las gráficas de los tres modelos entrenados, podemos observar a simple vista que el modelo Dense será el peor de los tres y los otros dos obtienen resultados muy similares. Por lo tanto, para poder decidir cuál de los dos modelos escoger, vamos a realizar tres comprobaciones:

- **Porcentaje de accuracy con "images_test":** Para ello se va a ejecutar el comando predict de cada una de las redes neuronales y después comprobar las etiquetas obtenidas y las que son correctas para así calcular el porcentaje.

  Una vez realizado este proceso podemos observar que efectivamente CNN y CNN2 están muy parejos, pero CNN empieza a destacar con mejores porcentajes. También podemos empezar a obtener la conclusión que Dense no es una opción por su mal rendimiento.

```
Accuracy Dense:  79.12698412698413 %

Accuracy CNN:  94.16666666666667 %
```

```
Accuracy CNN2:  89.96031746031746 %
```

- **Comprobación en tiempo real:** Pare este proceso se ha utilizado la aplicación creada y se han comparado resultados.

  Para sorpresa se ha encontrado un buen rendimiento de la red Dense, pero uno malísimo de la CNN2 en tiempo real. La red CNN obtiene buenos resultados como era de esperar. A continuación, se adjuntan algunas capturas de como se han realizado estas comprobaciones.

- **Porcentaje de accuracy con un conjunto de imágenes personalizado "white_images":** Para este proceso se ha creado una base de datos personalizada, escogiendo una o dos imagens por clase y se ha hecho el mismo proceso que en el punto anterior.

  Una vez realizado el proceso podemos observar que los resultados son malísimos en las tres redes neuronales, pero aun así vuelve a destacar la CNN por encima de las otras dos.

```
Accuracy Dense white_images:  31.11111111111111 %

Accuracy CNN white_images:  38.888888888888886 %

Accuracy CNN2 white_images:  31.11111111111111 %
```

Habiendo hecho estas tres comprobaciones, no se puede estar contento con los resultados debido a la baja precisión obtenida al predecir el conjunto de "white_images". Pero, ¿podemos saber a qué se debe esto? Pues a la hora de observar el conjunto de train de la base de datos que hemos utilizado, podemos observar como hay señales que tienen muchas imágenes para clasificar, pero en cambio otras que tienen muy pocas, y esto está balanceado de la misma manera en el conjunto de test. Por este motivo, a la hora de escoger 1 o 2 imágenes para cada señal, las señales que tienen pocos valores van a errar y bajan los porcentajes.

Por lo tanto, debido al mejor accuracy en el punto 1 y 3 de la red neuronal CNN y el mal comportamiento de la red CNN2 en tiempo real, se ha escogido la CNN como definitiva y se ha decidido modificar la base de datos para conseguir mejores resultados, explicado en el punto 6.7.

6.7 Modificación de la base de datos

Como hemos podido observar en el punto anterior, la culpable de no obtener exelentes resultados es la base de datos, por lo tanto, en un intento de mejorar el rendimiento, se han eliminado las señales que tenían pocas imágenes para entrenar. La base de datos resultante formada por "Custom_Training" y "Custom_Testing" han quedado de la siguiente manera.

Finalmente, se ha vuelto a aplicar el proceso del punto 6.3 con el modelo CNN, solo que en este caso se ha hecho el entrenamiento en 50 epochs para evitar un posible overfitting debido al menor nombre de clases. Y estos han sido los resultados.

Con el conjunto de Test:

```
Accuracy CNN:  96.04444444444445 %
```

Y con el conjunto nombrado White:

```
Accuracy CNN white_images:  67.3076923076923 %
```

Aumentado el 38% conseguido previamente en un 67%.

Finalmente, se ha incluido este modelo en la aplicación web como el principal, manteniendo los tres anteriores como secundarios

7 CONCLUSIONES

En este apartado veremos las conclusiones del trabajo y futuras implementaciones.

7.1 Conclusiones

En el proyecto hemos desarrollado el clasificador de señales y se han logrado todos los objetivos propuestos inicialmente. Ya que he logrado obtener un conocimiento inmenso sobre el funcionamiento de las redes neuronales y como mejorar su actuación.

Pero también cabe resaltar algunos aspectos que se deberían mejorar a la hora de poder realizar el módulo de conducción autónoma totalmente fiable y funcional.

Primero de todo se debería generar una base de datos con todas las señales que se pueda encontrar un vehículo en cualquier país y no solo 62 como en el proyecto. También cabe destacar que este es un proceso muy costoso, ya que se deben proporcionar muchas más de 4500 imágenes para el entrenamiento para evitar posibles errores del vehículo. Este proceso puede llevar muchos años para completarlo.

Y en segundo lugar, el módulo debería poder reconocer señales en cualquier tipo de entorno y poder enfocarlas, en el proyecto realizado solamente las reconoce si están perfectamente recortadas o enfocadas.

2.2 Futuras implementaciones

Como futura implementación se prevee el desarrollo de un algoritmo que pueda detectar en tiempo real si en una imagen global de una calzada de tráfico hay una señal de tráfico o no, y en el caso de que la haya poder enfocarla para eliminar el resto de ruido de la imagen y poder hacer la clasificación sin redundancias.

AGRADECIMIENTOS

Transmitir mi más sincero agradecimiento a todos aquellos que me han ayudado a lo largo de esta etapa y han colaborado en esta investigación.

En primer lugar, a mi tutor, Xavier Otatzu Porter, por su ayuda en la planificación, información y organización a lo largo del Trabajo de Fin de Grado.

En segundo lugar, a mi familia, mi madre Susana, mi padre Manel y mis hermanos. A mis amigos y al resto de mis familiares, que han estado a lo largo de toda mi carrera apoyándome en todo momento y animándome a seguir adelante.

También, expresar mi más sentido agradecimiento a la Universidad Autónoma de Barcelona por acogerme dentro de sus aulas y hacerme sentir como en casa.

Finalmente, no puedo olvidar los agradecimientos a la empresa COTECNA INSPECCIÓN, SL y a todos los compañeros que han compartido experiencias conmigo durante este año. Por permitirme obtener una primera inmersión en el mundo laboral y hacerme crecer como ingeniero y como persona.

A todos ellos mil gracias.

BIBLIOGRAFIA

[1] Sistemas de Visión Artificial: tipos y aplicaciones. [Online]. Available: https://www.edsrobotics.com/blog/sistemas-de-vision-artificial-tipos-aplicaciones/

[2] Estado del arte en conducción autonoma [Online]. Available: https://autohub.de/es/nachrichten/stand-der-dinge-beim-autonomen-fahren/

[3] Estado del arte – Inteligencia artificial [Online]. Available: https://inteligenciartificiallmca.wordpress.com/2017/06/10/1-3-estado-del-arte/

[4] Tensorflow – librería Python [Online]. Available: https://www.tensorflow.org/?hl=es-419

[5] Matplotlib – librería Python [Online]. Available: https://matplotlib.org/

[6] Os – librería Python [Online]. Available: https://docs.python.org/es/3.10/library/os.html

[7] Skimage – librería Python [Online]. Available: https://scikit-image.org/

[8] "Belgium TS Dataset" [Online]. Available: https://btsd.ethz.ch/shareddata/

[9] Ngrok [Online]. Available: https://ngrok.com/