## ¡Resultado de imagen de uc3m logo

Universidad Carlos III de Madrid  
Campus de Colmenarejo

Redes de Neuronas Artificiales

Curso 2021-22

Práctica 2: Parte 1. Problema de Clasificación con

Perceptrón multicapa

**Fernando Bermúdez Albarrán (100405854@alumnos.uc3m.es)**

**Andrei García Cuadra (100405803@alumnos.uc3m.es)**

**Jorge Ríos Marfil (100405942@alumnos.uc3m.es)**

Índice

[**Introducción**](#_8br9iub0r7v2) **2**

[Metodología de trabajo](#_tg21pgnh5lce) 2

[Jerarquía de Directorios](#_5s7advphoot6) 2

[Descripción del caso](#_vke7egq21o5f) 2

[Preprocesado de datos](#_9xb15vza11td) 2

[Forma de trabajo](#_e3zqbsdrlv9g) 3

[**Experimentación**](#_1q3pj2obae) **3**

[**Conclusiones**](#_su0yqknnmzjc) **8**

# 

# Introducción

## Metodología de trabajo

Se ha establecido una metodología basada en SCRUM, realizando las reuniones stand-up diariamente y estableciendo unos hitos reducidos para realizar una progresión lineal basada en la constancia del proyecto. Estipulando que el proyecto se debe basar en nuestra creatividad e ingenio, hemos optado por esta metodología ya que nos permite obtener diversos puntos de vista según progresa el tiempo y la asignatura.

## Jerarquía de Directorios

A continuación se detallan los directorios y archivos del proyecto.

├── memoria.pdf

├── proyecto.zip

│ ├── pesos.h5 # Fichero de pesos

│ ├── main\_p1.py # Fichero con script utilizado para entrenar el modelo

│ ├── vehicleDataPrepare.py # Fichero con script utilizado para normalizar los datos

│ ├── vehicleTest.csv # Fichero con datos de test

│ ├── vehicleTrain.csv # Fichero con datos de train

│ ├── output.csv # Fichero con predicciones y salidas

│ ├── screenshots/ # Carpeta con todas las pruebas realizadas

## Descripción del caso

Se pretende cubrir la necesidad de clasificación documental sobre trabajos fotográficos de vehículos captados en su circulación. Para ello, se han estipulado las siguientes categorías:

OPEL (212 instancias)

SAAB (217 instancias)

BUS (218 instancias)

VAN (199 instancias)

Para poder realizar la clasificación, se nos entrega un dataset con 18 valores de entrada y la label correcta(Es decir, opel, saab, bus o van). A partir de eso tenemos que normalizar los datos y clasificarlos haciendo uso de un perceptron multicapa.

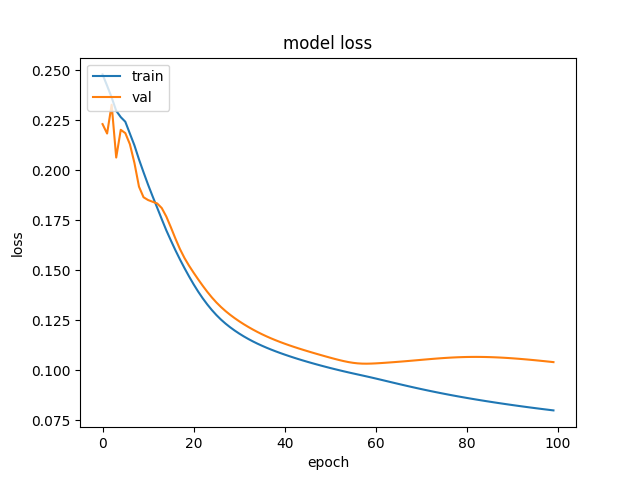
## Preprocesado de datos

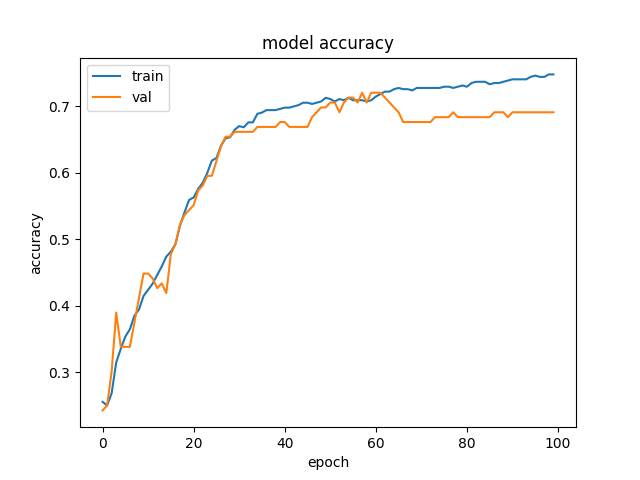
Para preprocesar los datos hemos hecho uso del script que está añadido en el zip “vehicleDataPrepare.py” el cual lee el fichero .csv y normaliza las primeras 17 columnas pasando los valores a un rango entre 0 y 1 para ello hace uso del máximo y mínimo de cada columna, posteriormente divide el dataset en dos a la vez que aleatoriza los valores, en nuestro caso hemos decidido una repartición 80% (train y validación) y un 20% (test).

## Forma de trabajo

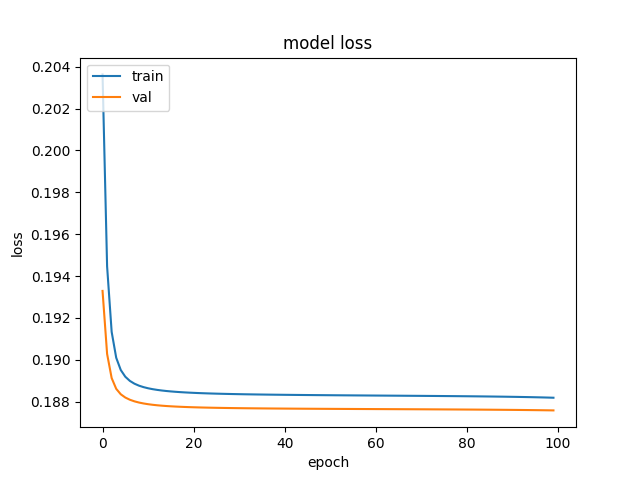
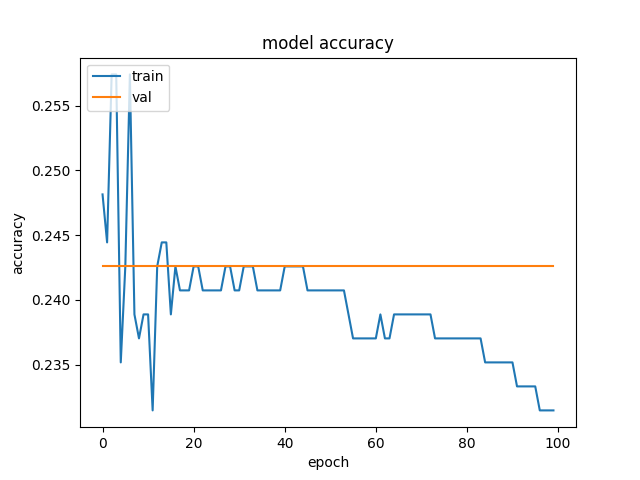
Para encontrar los mejores valores de los hiperparametros partimos de los valores que se nos dieron en el enunciado de la práctica, y desde ahí empezamos a modificar el número de capas, el número de neuronas por capa, el número de ciclos, y por último el learning rate. Hemos probado tanto a añadir más capas, más neuronas por capa y modificar el número de iteraciones.

# Experimentación

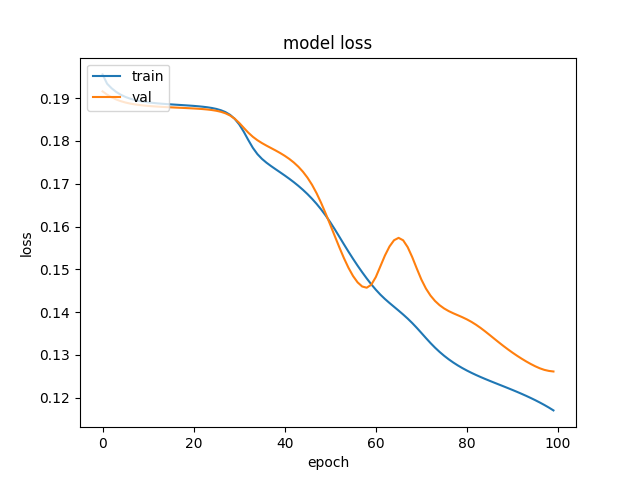
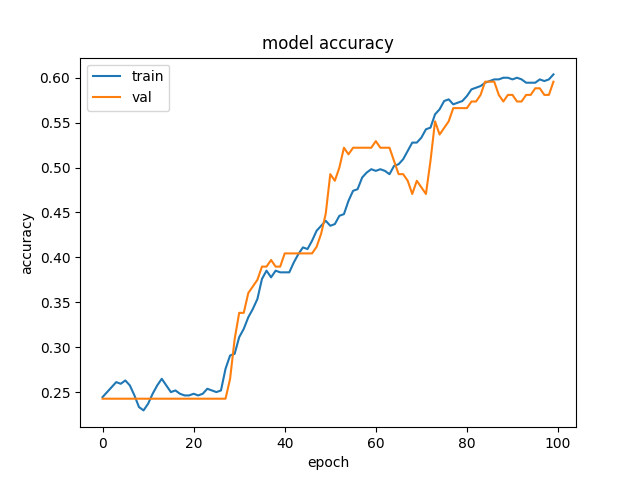
A continuación podemos observar las dos gráficas iniciales:



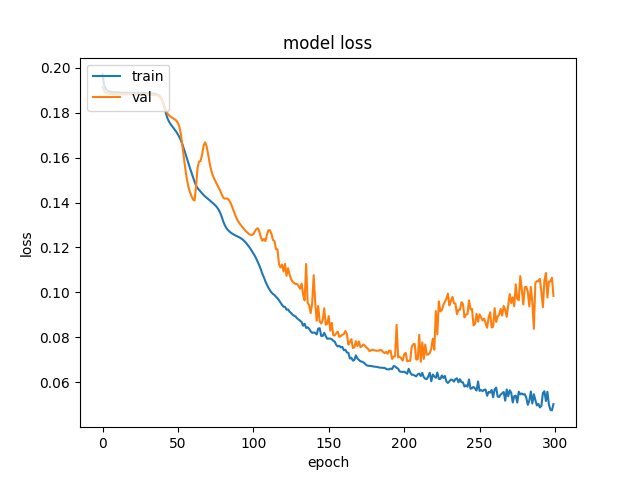
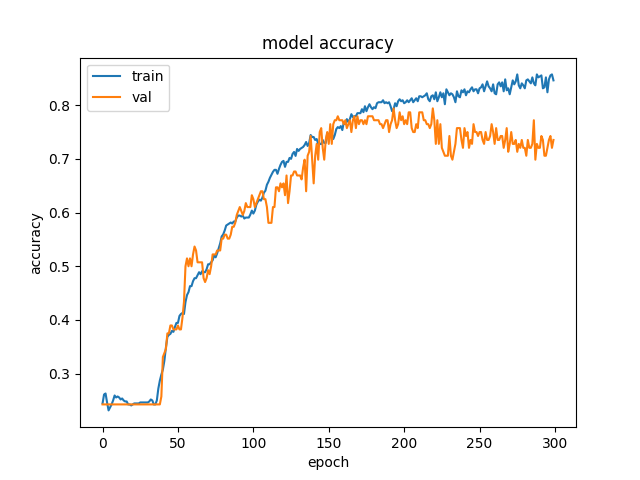
Lo primero que probamos fue a aumentar el número de capas, pasando a tener tres capas con 350,200,100 neuronas y un learning rate de 0.2 y el resultado fue el siguiente:



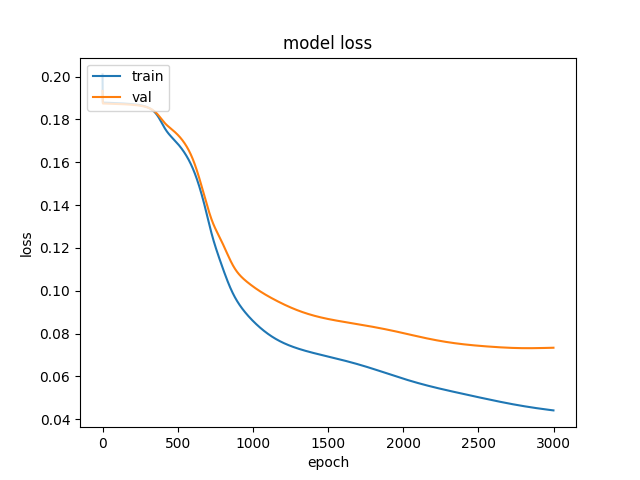
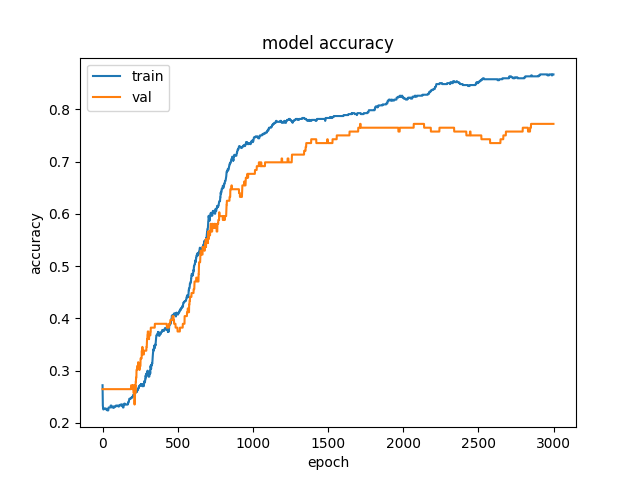
El resultado no fue el esperado, ya que se redujo en gran medida el accuracy del modelo, por lo que probamos la misma configuración de capas y neuronas pero con diferentes learning rates(0.02 y 0.6), y no dio resultado, seguimos con los mismos resultados. Por lo que procedimos a cambiar el número el número de neuronas por capa, y pasamos a una configuración de 100, 50, 35 y un lr de 0.2 y obtuvimos mejores resultados que con la anterior configuración:



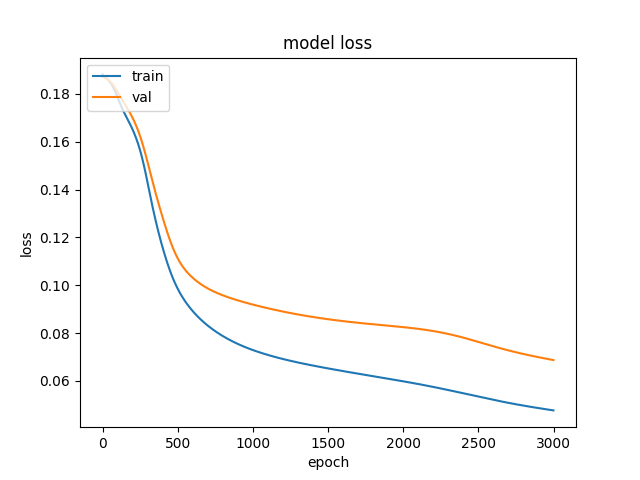
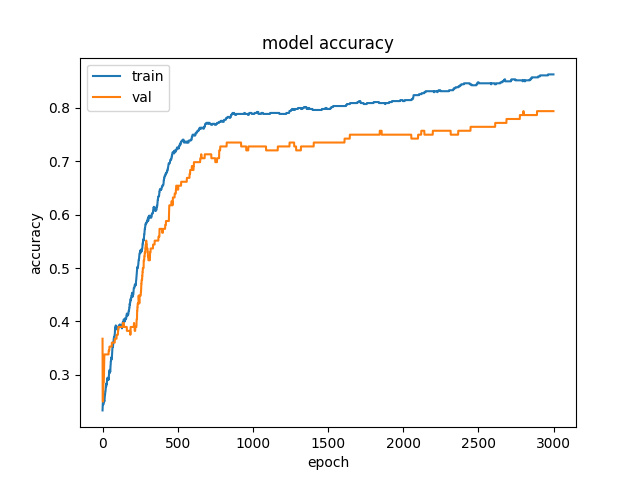
Probamos a modificar el learning rate y bajarlo a 0.02 pero conseguimos peores resultados, por lo que probamos a aumentarlo hasta 0.35 y aumentar el número de ciclos hasta 300 consiguiendo mejores resultados:



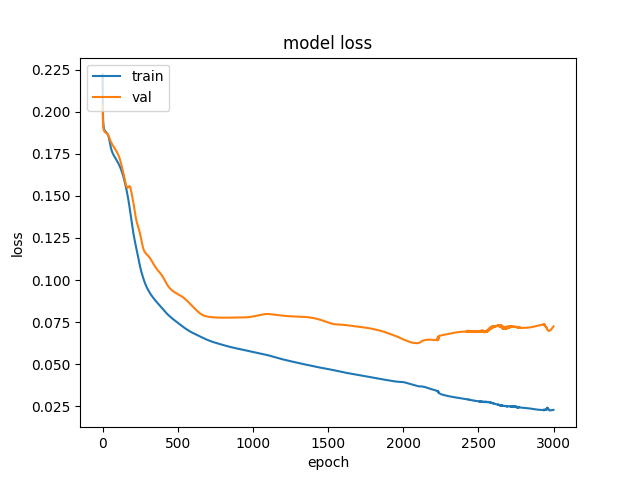
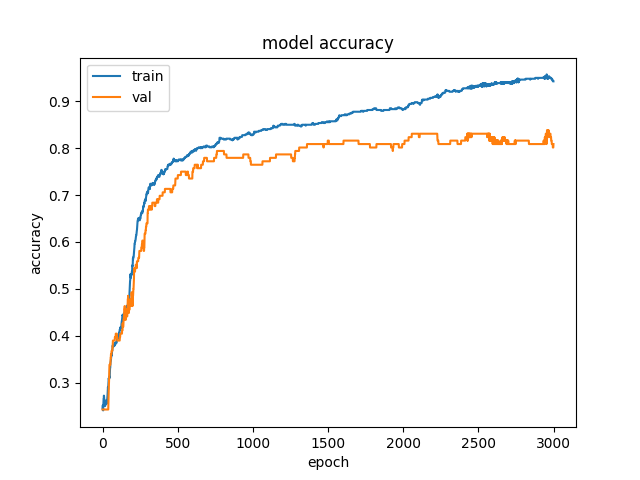
En este punto se aprecia que aumentando el número de ejecuciones se produce un overfitting por lo que probamos a aumentar más aún el número de ciclos pero reduciendo el learning rate en dos pasos (0.15 , 0.01) obteniendo con lr = 0.01 el siguiente resultado:



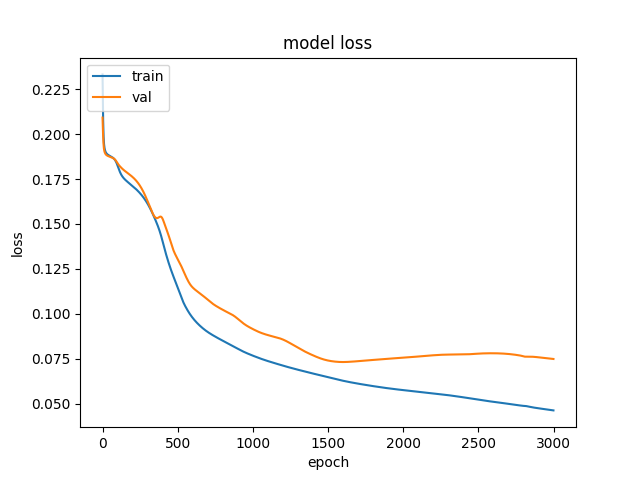
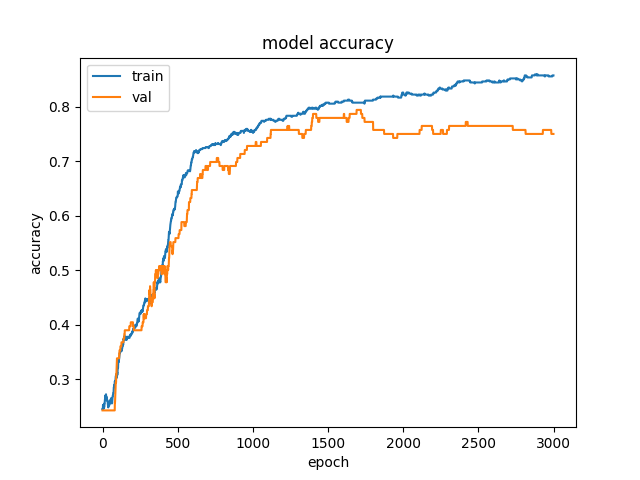
En este punto decidimos cambiar el número de capas que forman la red, pasando a una red de 100 neuronas en la primera capa y sesenta en la segunda obteniendo buenos resultados pero estancandose:



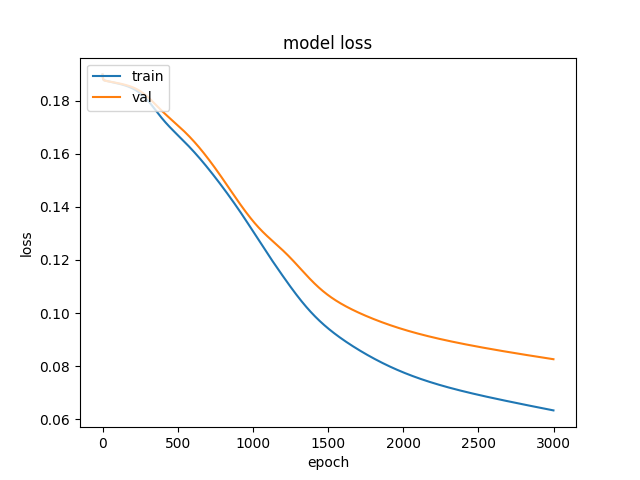
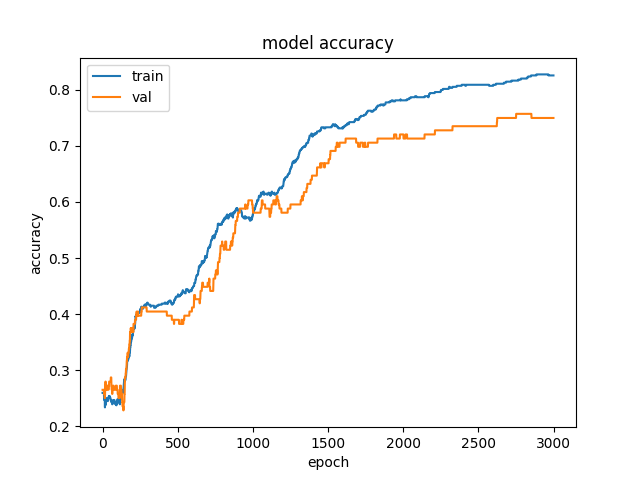
En este punto cambiamos dramáticamente el número de capas pasando a una configuración de 1200 neuronas en la primera capa y 600 en la segunda, con un lr=0.06 obteniendo los siguientes resultados:



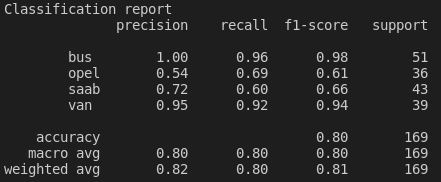
El resultado anterior ha sido el mejor que hemos obtenido, a parte de esas pruebas hemos realizado muchas más pruebas tanto con un alto número de neuronas como el que se muestra a continuación que tiene 2500 neuronas en la primera capa y 1500 neuronas en la segunda capa con un lr=0.06:



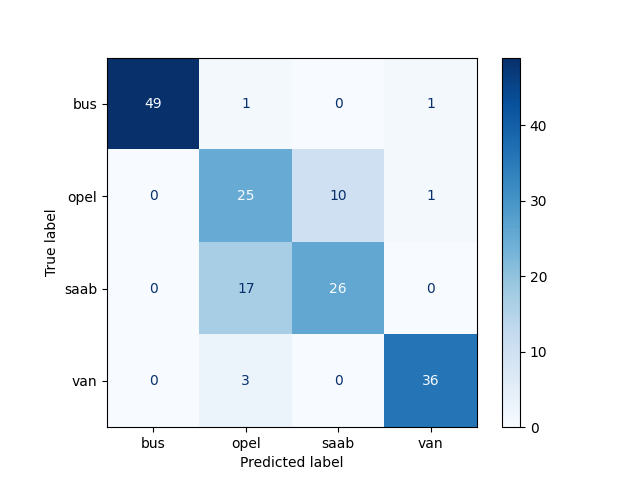
Obtenemos buenos resultados pero son inferiores a los anteriores, por lo que pasamos al otro extremo que es reducir drásticamente el número de neuronas en cada capa a una configuración de 12 en la primera capa y 4 en la segunda con un lr=0.005:



Igual que con el caso anterior se obtienen buenos resultados pero siguen siendo inferiores a la configuración 1200-600, por lo que pasamos a explicar los resultados obtenidos con la mejor configuración:



Ahí podemos observar el resultado de dicho modelo con el 100% de los datos de train, el accuracy disminuye y aunque por pruebas sabemos que usando el 100% de train con diferentes configuraciones se obtiene un mayor accuracy nos hemos fijado únicamente en el accuracy de validation ya que sería el proceso a seguir en un caso real, es decir que no se conozca el dataset de test.



Si observamos su matriz de confusión vemos que tanto para van como para bus el modelo funciona a la perfección, alcanzando una precisión de 1 en bus, pero para opel y saab no es del todo preciso el modelo.

El resto de pruebas realizadas se pueden encontrar en la carpeta screenshot, y el formato seguido para nombrar el fichero es el siguiente:

*nºneuronas1ªCapa\_nºneuronas2ªCapa\_nºneuronas3ªCapa\_learningRate\_tipo.png*

en caso de ser solo dos capas el tercer valor sería el learning rate, y el tipo si es una “a” quiere decir que es la gráfica de accuracy, y en caso de ser “p” es que es el de pérdidas o loss.

# Conclusiones

La práctica nos ha servido para trabajar con redes de neuronas para la clasificar datos, nos ha parecido muy interesante ya que nos muestra de una forma práctica lo aprendido en las clases de teoría, además de un enfoque práctico que es lo de verdad útil para un futuro. El hecho de contar con un pdf con los pasos detallados de cómo funciona el código, nos parece un gran acierto.

Nos hubiera gustado que hubiese habido algún tipo de competición o concurso para que se valorará el número de entrenamientos, pruebas y resultados obtenidos.