

Vision por Computador

Practica de Clasificación

Héctor Felipe Mateo Romero
Sandra Gómez Gálvez



DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL
UNIVERSIDAD POLITÉCNICA DE MADRID

Profesores:

Luis Baumela Molina

15 de enero de 2021

Índice general

| | |
|--|-----------|
| 1. Introducción | 3 |
| 2. Descripción de los problemas | 4 |
| 2.1. Dataset CIFAR-10 | 4 |
| 2.2. Dataset Señales Tráfico | 5 |
| 3. Resultados finales | 7 |
| 3.1. Dataset CIFAR-10 | 7 |
| 3.1.1. Red alimentada hacia delante profunda | 7 |
| 3.1.2. Red convolucional | 8 |
| 3.2. Dataset Señales Tráfico | 10 |
| 3.2.1. Red alimentada hacia delante profunda | 10 |
| 3.2.2. Red convolucional | 11 |
| 4. Proceso de diseño de las redes | 14 |
| 4.1. Protocolo utilizado | 14 |
| 4.2. Dataset CIFAR-10 | 15 |
| 4.2.1. Red alimentada hacia delante profunda | 15 |
| 4.2.2. Red Convolucional | 16 |
| 4.3. Dataset Señales Trafico | 19 |
| 4.3.1. Red alimentada hacia delante profunda | 19 |
| 4.3.2. Red Convolucional | 27 |

| | |
|---|-----------|
| 5. Conclusiones | 39 |
| 5.1. Dataset CIFAR-10 | 39 |
| 5.2. Dataset Señales Tráfico | 39 |
| 5.3. Comparación ambos datasets | 40 |

Capítulo 1

Introducción

En el mundo de la visión artificial, uno de los problemas más importante es el reconocimiento de imágenes. Este problema se abordó de diversas formas, pero su implementación ha mejorado notablemente gracias a la llegada del Deep Learning. El Deep Learning ha supuesto una mejora en todos los campos de la visión y concretamente en el reconocimiento de imágenes gracias a un tipo de red neuronal, llamada *Red Convolutacional*, que funciona muy bien para esta tarea. Con esta red, con la arquitectura adecuada y un suficiente numero de imágenes es posible construir clasificadores muy precisos que actualmente llegan a % de acierto muy superiores a los de otros métodos.

El objetivo de esta práctica es reconocer objetos en dos datasets muy conocidos: *CIFAR-10* y *German Traffic Sign Detection Benchmark*.

El trabajo se distribuye de la siguiente manera. En la sección 2 se describirá el problema y los distintos datasets sobre los que trabajaremos. A continuación en la sección 3 se mostrarán los resultados obtenidos de las redes diseñadas que mejor rendimiento han aportado para los diferentes problemas. En esta sección se discutirán para ambos datasets tanto los resultados para la red alimentada hacia adelante profunda como para la red convolucional. En la siguiente sección 4, se hablará del proceso que se ha seguido para llegar a las distintas redes finales. Finalmente, en 5 se compararán los resultados obtenidos.

Capítulo 2

Descripción de los problemas

2.1. Dataset CIFAR-10

El dataset CIFAR-10 está compuesto por 60000 imágenes en color de un tamaño de 32 x 32 píxeles. Las imágenes están divididas en 10 clases, con 6000 imágenes por clase. En la figura 2.1 se muestran imágenes de cada una de las clases.

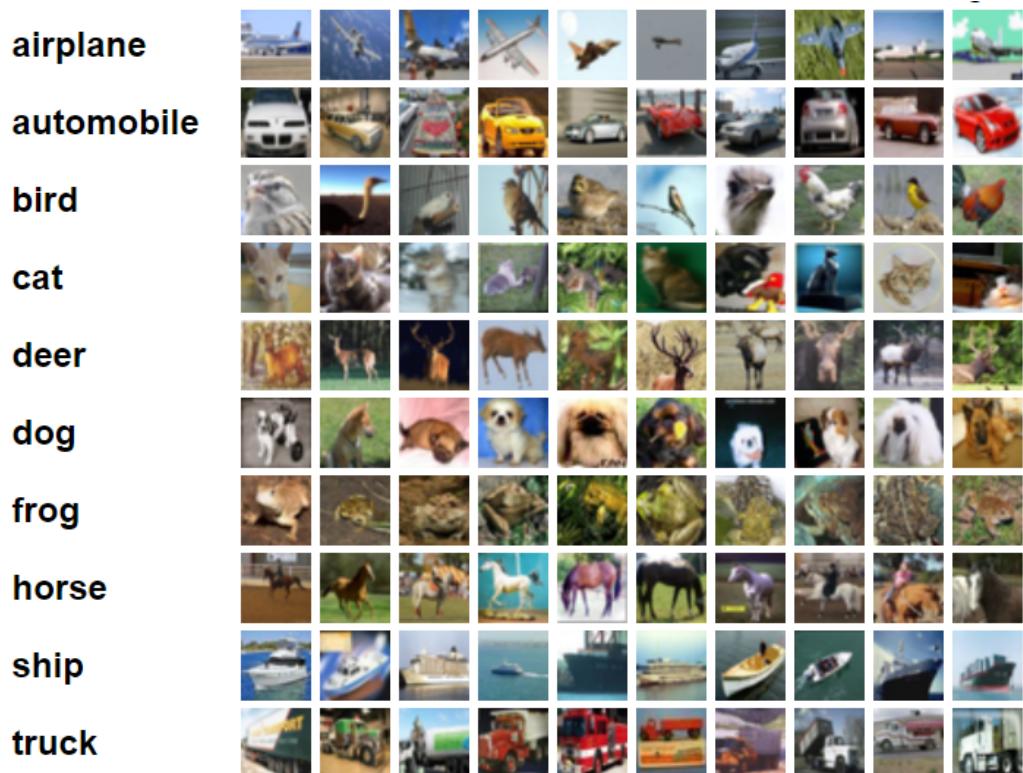


Figura 2.1: Dataset Cifar10 <https://www.cs.toronto.edu/~kriz/cifar.html>

Observando el dataset vemos que posee limitada cantidad de clases y además una alta variedad imágenes para cada una de las clases. Podemos observar también que las imágenes tienen un tamaño

pequeño, lo que simplifica el problema.

El objetivo será crear en primer lugar una red neuronal que no utilice ningún tipo de capa convolucional, para así observar cual es el límite en la potencia de este tipo de redes. En segundo lugar utilizaremos una arquitectura basada en las capas convolucionales intentando llegar a obtener el mayor porcentaje de acierto mientras seguimos una metodología adecuada.

2.2. Dataset Señales Tráfico

El dataset *German Traffic Sign Detection Benchmark(GTSDB)* está compuesto por 900 imágenes de señales de tráfico. Estas imágenes son en color, tienen un tamaño de 1360x800 y extensión .ppm (para visualizarlas recomendamos el software *GIMP*). Un ejemplo de estas imágenes se puede ver en la figura 2.2.



Figura 2.2: Imagen 00000.ppp

Además, el dataset está acompañado de un archivo *gt.txt* (figura 2.3) que indica para cada imagen de señal de tráfico:

- Primera columna: Nombre de la imagen. Por ejemplo *00878.ppm*.
- Columnas de la 2 a la 5: Región de interés de la imagen.
- Última columna: tipo de señal de tráfico.

```

00000.ppm;774;411;815;446;11
00001.ppm;983;388;1024;432;40
00001.ppm;386;494;442;552;38
00001.ppm;973;335;1031;390;13
00002.ppm;892;476;1006;592;39
00003.ppm;742;443;765;466;4

```

Figura 2.3: Fragmento del archivo *gt.txt*

Este archivo tiene una longitud de 1213. Como se ha explicado, hay 900 imágenes, y que este archivo tenga una longitud de 1213 se debe a que para algunas imágenes se indican varias regiones de interés.

Nosotros preprocesaremos las imágenes y para la Red Neuronal solo utilizaremos las regiones de interés. Por tanto, tendremos 1213 imágenes finalmente para utilizar en nuestra red neuronal. Una muestra de estas imágenes se puede ver en la figura 2.4.

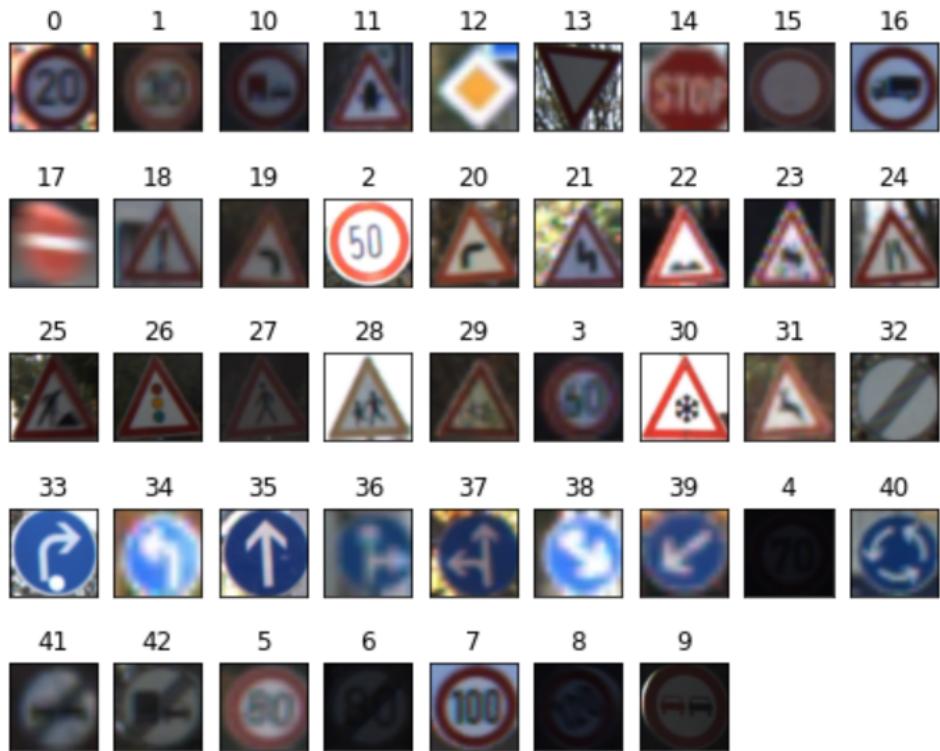


Figura 2.4: Imágenes usadas para entrenar y validar (dev y test final) la Red Neuronal

El objetivo será el mismo que para el dataset CIFAR-10:

1. Implementar una Red Neuronal alimentada hacia delante (*Feed-forward*) para observar la potencia de este tipo de redes en clasificación de imágenes.
2. Implementar una Red Neuronal Convolucional para tratar de obtener un mejor rendimiento.

Capítulo 3

Resultados finales

En esta sección se mostrarán y analizarán los resultados obtenidos de las redes diseñadas que mejor rendimiento han aportado para los diferentes problemas.

3.1. Dataset CIFAR-10

3.1.1. Red alimentada hacia delante profunda

La red profunda, como era de esperar, no consiguió resolver satisfactoriamente el problema. Tras realizar un proceso de ajustes de parámetros, la mejor red obtenida solo conseguía un 58.37 % en el conjunto de entrenamiento, un 48.640 % en el conjunto de validación y un 47.272 % en el conjunto de prueba. Se aplicó early-stopping para parar el entrenamiento cuando no se estén obteniendo redes mejores y se ha almacenado la mejor red hasta el momento.

Podemos ver en la figura 3.1 la evolución de la precisión y el error durante el proceso de entrenamiento y en la tabla 3.1 los hiper-parámetros que se han utilizado.

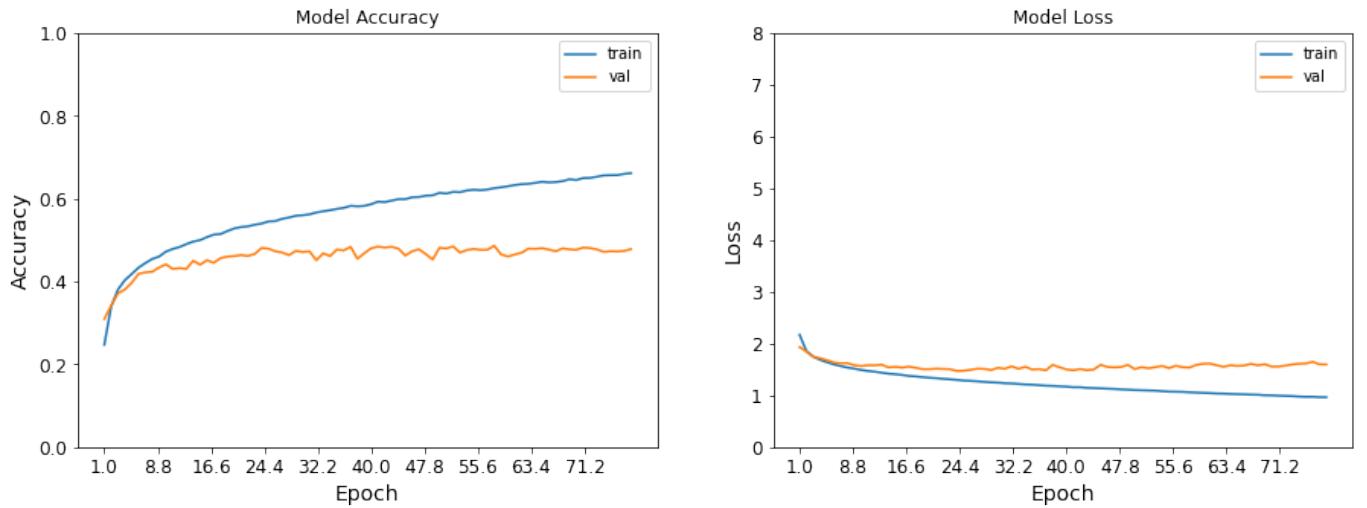


Figura 3.1: La evolución de la precisión y el error durante el proceso de entrenamiento

| Hiper-parámetros | Valor |
|---------------------------|--------------------------|
| Neuronas por capa | [75 neuronas x 10 capas] |
| Tasa de aprendizaje | 0.00005 |
| Nº de épocas | 100 |
| m_b (tamaño del batch) | 128 |
| Función de activación | ReLU |
| Función de inicialización | XAVIER NORMAL |
| Función de salida | softmax |
| Dropout | No |
| Optimizador | Nadam |
| Regularizador | No |
| Normalización | SI |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 3.1: Hiper-parámetros de la red final.

3.1.2. Red convolucional

La red convolucional conseguida si que resuelve el problema de forma satisfactoria, obteniendo una precisión del 84.80 % en el conjunto de entrenamiento, del 83.90 % en el conjunto de validación y del 83.61 % en el conjunto de test. Se ha aplicado early-stopping para parar el entrenamiento cuando no se estén obteniendo redes mejores, además de data-augmentation en el conjunto de entrenamiento y almacenar la mejor red hasta el momento.

Podemos ver en la figura 4.6 la evolución de la precisión y el error durante el proceso de entrenamiento

Se ha utilizado una red con la siguiente arquitectura 3.3 y en la tabla 3.2 podemos ver los hiperparámetros que se han utilizado. .

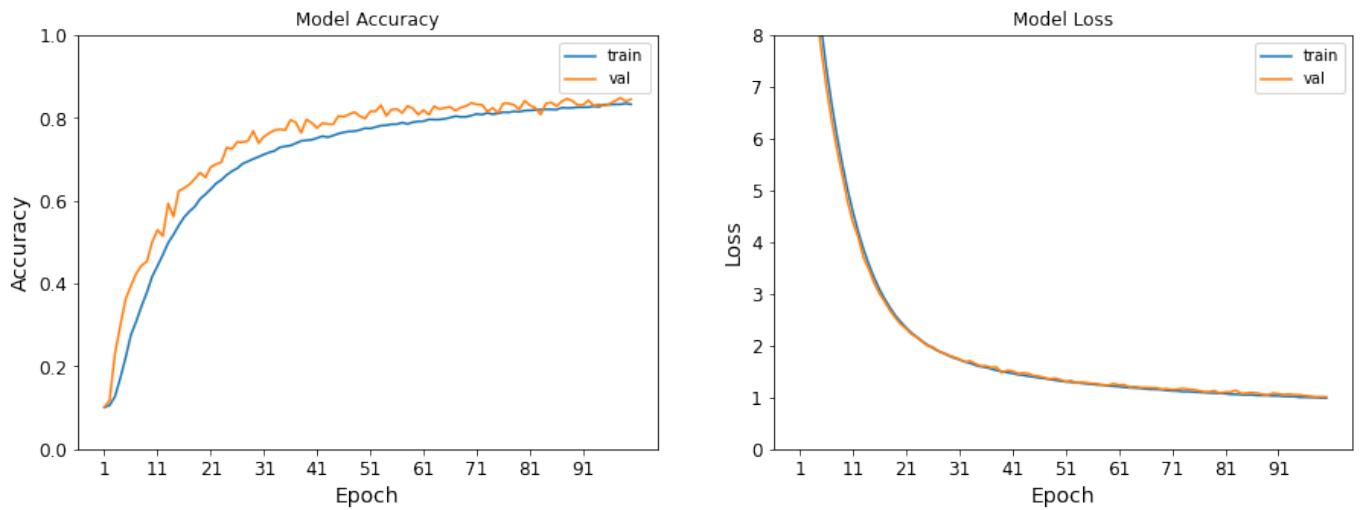


Figura 3.2: La evolución de la precisión y el error durante el proceso de entrenamiento

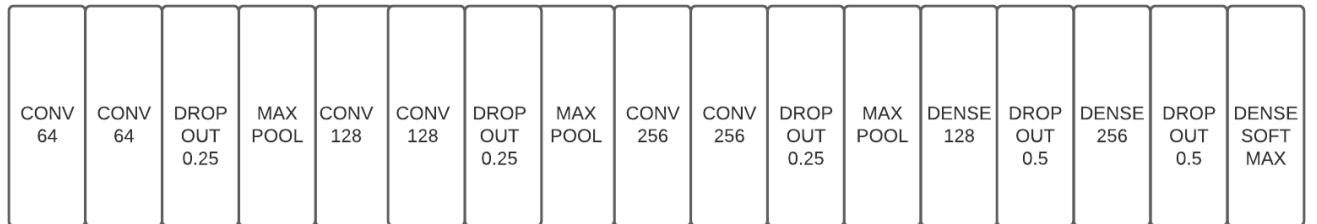


Figura 3.3: Estructura de la red convolucional

| Hiper-parámetros | Valor |
|---------------------------|--|
| Tasa de aprendizaje | 0.00005 |
| Nº de épocas | 100 |
| m_b (tamaño del batch) | 128 |
| Función de activación | ReLU |
| Función de inicialización | Xavier_normal |
| Función de salida | softmax |
| Optimizador | Nadam |
| Regularizador | Sí, -Data augmentation -Dropout -L2(weight_decay=0.01) |
| Normalización | Si |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 3.2: Hiper-parámetros de la red - Red final CNN

3.2. Dataset Señales Tráfico

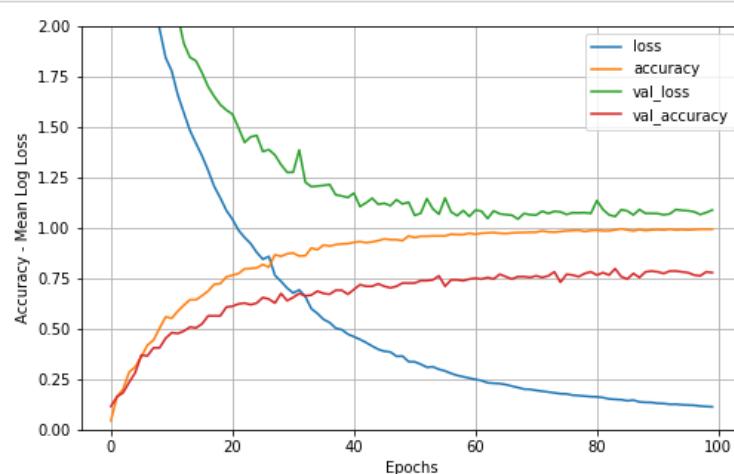
3.2.1. Red alimentada hacia delante profunda

La red alimentada hacia delante profunda que hemos considerado como red final con la configuración de la tabla 3.3. Nos ha aportado unos resultados (figura 3.4) que podríamos considerar como óptimos aunque es cierto que es una red que no llega a generalizar adecuadamente.

Al ser esta red considerada la final, se ha ejecutado el test final en ella y se ha obtenido una precisión (*Accuracy*) del 83 %. Resultados que son óptimos, aunque podrían ser mejores. Estos resultados se pueden ver en 3.5.

| Hiper-parámetros | Valor |
|---------------------------|---|
| Neuronas por capa | [(224, 244), (224, 224), 150, 100, 80, 43] |
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 100 |
| m_b (tamaño del batch) | 100 |
| Función de activación | ELU |
| Función de inicialización | he_normal |
| Función de salida | softmax |
| Dropout | [0.8,0.4,0.2] |
| Optimizador | SGD ($lr = 0.001$, $momentum = 0.9$, $nesterov = True$) |
| Regularizador | No |
| Normalización | BatchNormalization |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 3.3: Hiper-parámetros - Red Final ffNN



```
loss    accuracy   val_loss   val_accuracy
99  0.112725  0.993333  1.088261      0.777778
```

Figura 3.4: Resultados - Red Final ffNN

Test loss: 0.7868781089782715 - Accuracy: 0.8310249447822571

Figura 3.5: Resultados Test Final - Red Final ffNN

3.2.2. Red convolucional

En cuanto a la red convolucional que hemos considerado como red final, tiene la arquitectura de la figura 3.6 y la configuración de la tabla 4.2. Nos ha aportado unos resultados (figura 3.7) con los que no estamos muy conformes, ya que el bias se podría considerar como alto al igual que la varianza. Esto puede ser debido a los pocos ejemplos de entrenamiento de los que disponemos.

Al ser esta red considerada la final, se ha ejecutado el test final en ella y se ha obtenido una precisión (*Accuracy*) del 70 %. Estos resultados eran de esperar viendo la tendencia de la precisión del test de validación. Estos resultados se pueden ver en 3.8.

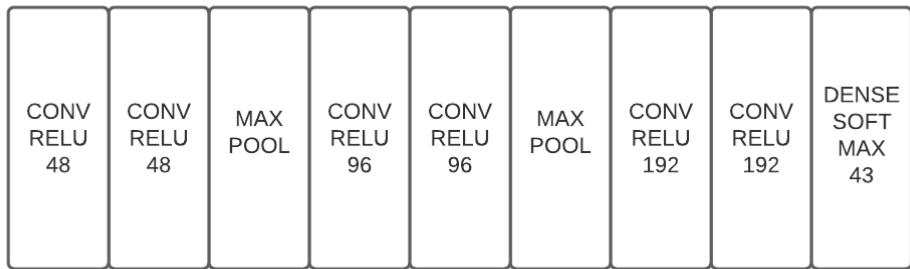


Figura 3.6: Estructura - Red Final CNN

| Hiper-parámetros | Valor |
|---------------------------|--|
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 100 |
| m_b (tamaño del batch) | 32 |
| Función de activación | ReLU |
| Función de inicialización | Sí, He_normal |
| Función de salida | softmax |
| Dropout | Sí [0.1,0.2, 0.2] |
| Optimizador | Adam ($lr = 0,001, beta_1 = 0,9, beta_2 = 0,999$) |
| Regularizador | <p>Sí,</p> <p>-Data augmentation:</p> <p><i>featurewise_center = False,</i> <i>featurewise_std_normalization = False,</i> <i>rotation_range = 10.,</i> <i>width_shift_range = 0,1,</i> <i>height_shift_range = 0,1,</i> <i>zoom_range = 0,3,</i> <i>shear_range = 0,1,</i> <i>horizontal_flip = True</i>)</p> <p>-Dropout</p> <p>-L2($weight_decay=1e-4$)</p> |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 3.4: Hiper-parámetros de la red - Red final CNN

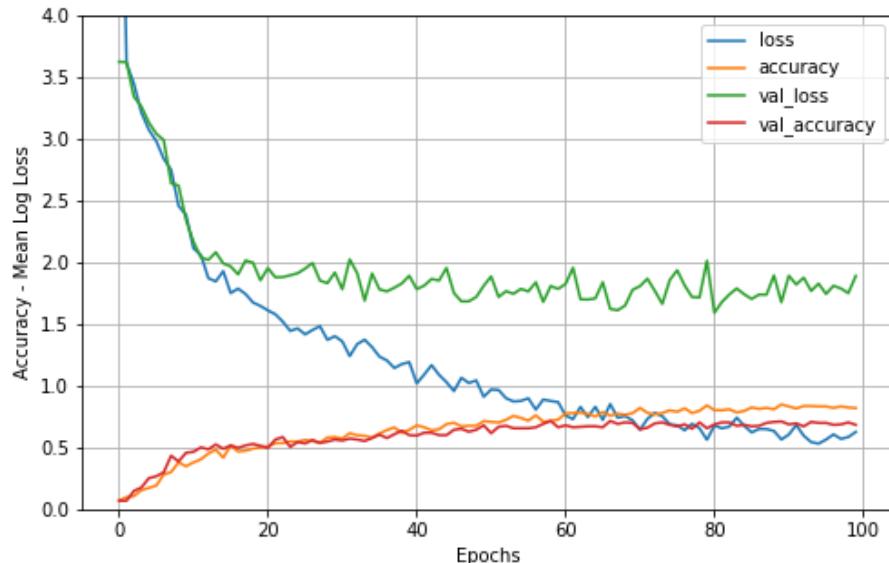


Figura 3.7: Resultados de la red - Red Final CNN

```
Test loss: 1.4171252250671387 - Accuracy: 0.7036011219024658
```

Figura 3.8: Resultados Test Final - Red Final CNN

Capítulo 4

Proceso de diseño de las redes

En esta sección se mostrará el proceso que se ha seguido para llegar a las distintas redes finales que mejor rendimiento han aportado.

4.1. Protocolo utilizado

Para la creación de las redes hemos utilizado el siguiente protocolo:

1. Ajuste inicial de los parámetros: En primer lugar retocaremos parámetros básicos como el learning rate, función de activación y optimizador para obtener una red que si aprenda aunque posea una precisión pobre.
2. Ajuste de bias y varianza : En esta parte lo que haremos ir tomando decisiones para reducir bias y varianza basándonos el diagrama 4.1 hasta que no podamos mejorar mas la red

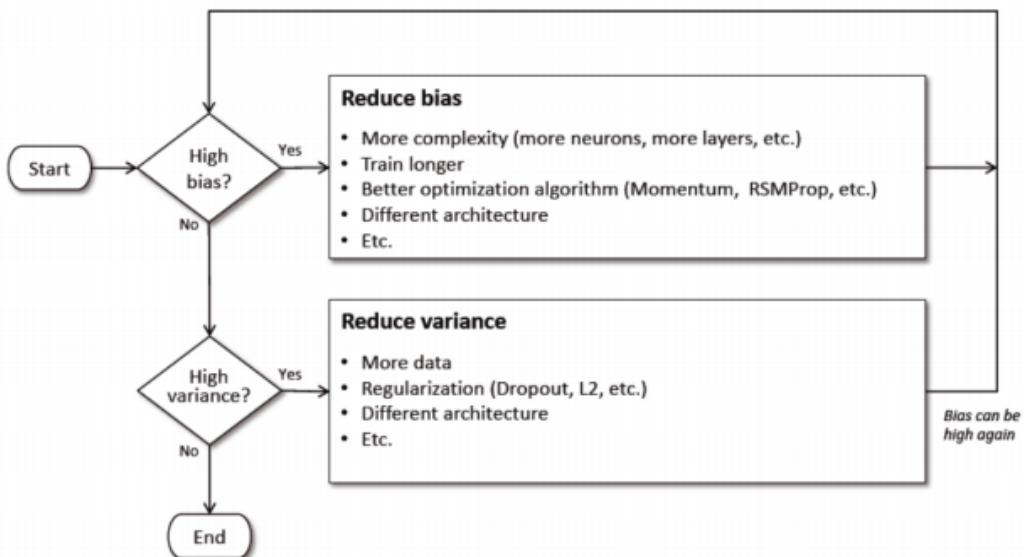


Figura 4.1: La evolución de la precisión y el error durante el proceso de entrenamiento de la red inicial

4.2. Dataset CIFAR-10

4.2.1. Red alimentada hacia delante profunda

Utilizaremos como modelo inicial la red de la tabla ??.

| Hiper-parámetros | Valor |
|---------------------------|--------------------------|
| Neuronas por capa | [50 neuronas x 10 capas] |
| Tasa de aprendizaje | 0.01 |
| Nº de épocas | 100 |
| m_b (tamaño del batch) | 126 |
| Función de activación | SeLU |
| Función de inicialización | XAVIER NORMAL |
| Función de salida | softmax |
| Dropout | No |
| Optimizador | Nadam |
| Regularizador | No |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.1: Hiper-parámetros de la red inicial.

Los resultados de la red podemos verlos en la figura 4.2

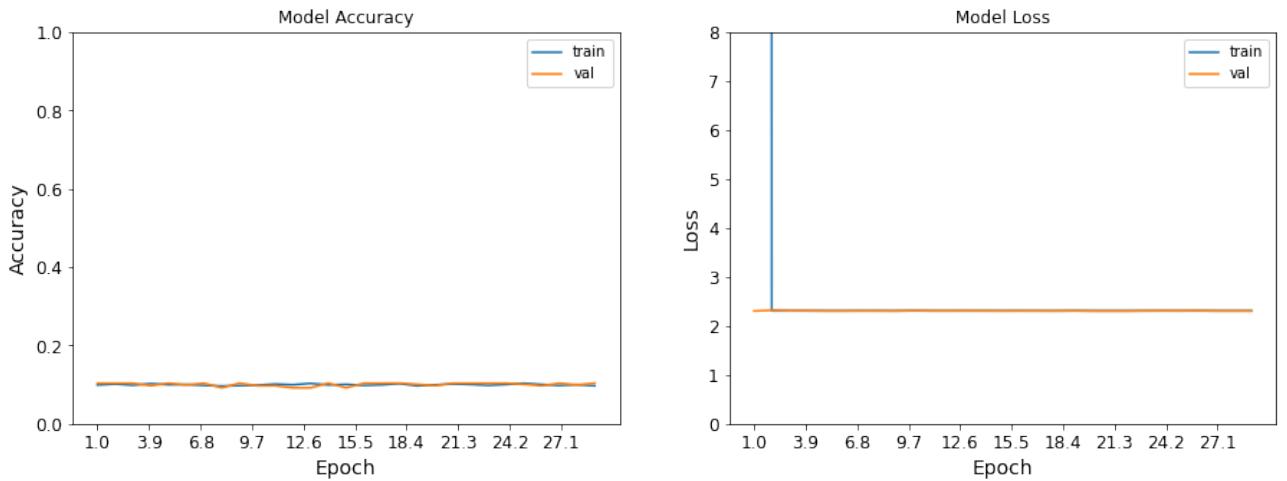


Figura 4.2: La evolución de la precisión y el error durante el proceso de entrenamiento de la red inicial

Podemos observar que esta red no realiza ningún tipo de aprendizaje, posee aproximadamente un 10 % de precisión, correspondiente a decir continuamente una de las posibles clases para todas las instancias independientemente del tipo.

Comenzamos probando añadiendo normalización de batches a la red, arreglando el problema que poseíamos y obteniendo un 55 % en el conjunto de entrenamiento y un 44 % en el conjunto de validación. Como son resultados bajos procedemos a hacer mas compleja la red con la esperanza que esto le de mas

capacidad de aprendizaje. Probamos varios casos: duplicando el numero de capas, añadiendo neuronas a cada una y combinando ambas. Los mejores resultados se obtuvieron al utilizar 10 capas de 75 neuronas. Con un 65% en entrenamiento y un 46% en validación.

Como se empezaba a ver un overfitting se probaron diversas técnicas de regularización como L1, L2 o dropout pero los resultados fueron inferiores a los anteriores.

Tras fracasar en reducir el overfit sin perjudicar la red se decidió probar con otras de funciones de activación. Se observó que la función *Selu* producía resultados ligeramente mejores que otras como Relu, sigmoide, elu,...

Siguiendo esta linea se decidió modificar el learning rate, se probaron valores desde $1 * 10^{-1}$ hasta $1 * 10^{-6}$, consiguiendo el mejor resultado (55% y 47.9%) con $5 * 10^{-5}$.

Otro intento que se siguió para reducir el bias fue cambiar el algoritmo de optimización, se probaron SGD, Adam y Nadam y se observó que Nadam daba unos resultados mejores (58.37% en el conjunto de entrenamiento, un 48.640% en el conjunto de validación).

Finalmente se realizaremos otras pruebas como volver a modificar la arquitectura o cambiar el inicializador pero ninguna consiguió mejorar la red e incluso perjudicaban su rendimiento enormemente. Como no se encontraba ninguna red mejor se decidió aceptar esta red como la red final.

4.2.2. Red Convolucional

Utilizaremos como modelo inicial una red con la estructura de la figura 4.3 y los hiper-parámetros de la tabla ??, inspirados en los utilizados en la anterior red.

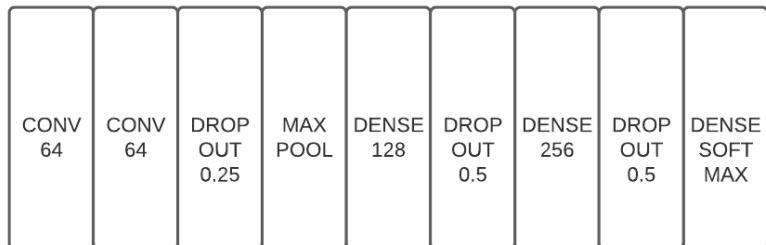


Figura 4.3: Estructura de la red utilizada

En la figura 4.4 podemos ver la evolución de la precisión y el error a lo largo del entrenamiento de la red. Esta red obtiene un 72% de aciertos en el conjunto de entrenamiento y un 76% en el de validación. Podemos observar como una red convolucional simple resuelve el problema de forma mucho mas satisfactoria que la profunda pero esta precisión puede ser mejorada aún mas.

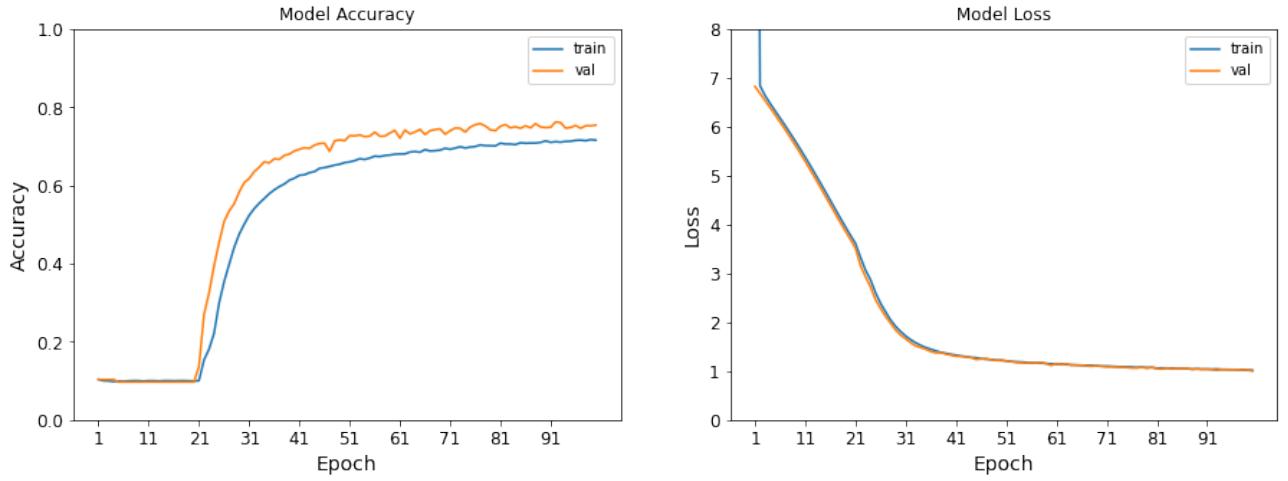


Figura 4.4: Estructura de la red utilizada

| Hiper-parámetros | Valor |
|---------------------------|--------------------------|
| Tasa de aprendizaje | 0.1 |
| Nº de épocas | 100 |
| m_b (tamaño del batch) | 128 |
| Función de activación | ReLU |
| Función de inicialización | Xavier_normal |
| Función de salida | softmax |
| Optimizador | Adam |
| Regularizador | No |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.2: Hiper-parámetros de la red - Red inicial CNN

En primer lugar se procedió a modificar la estructura de la red, añadiendo 2 capas convolucionales de 128 filtros y una de maxpool consiguiendo resultados similares a la anterior pero con un mayor overfitting. En segundo lugar , de forma similar se añadieron 2 capas de 256 filtros. Los resultados de esta red si fueron mejores, llegando a un 78 % en el conjunto de validación y un 99 % en el de entrenamiento en un menor número de epochs. A las vistas de que aumentar la estructura estaba aumentando los aciertos pero también incrementando enormemente el overfitting se decidió aplicar técnicas de regularización.

Se comenzó aplicando la técnica L1 con el parámetro por defecto: 0.01. Los resultados fueron muy malos, ya que provocaba que la red realizase ningún tipo de aprendizaje. En su lugar se optó por la técnica L2. Esta técnica reportó unos resultados muy interesantes : En primer lugar se consiguió una reducción del overfit con un 90.48 % en el conjunto de entrenamiento y además un incremento considerable en el conjunto de validación hasta llegar al 80.81 %. Se probó con otros valores como 0.1 o 0.03 pero ninguno reportó mejores resultados que el inicial. Además la combinación de L1 Y L2 provocaba resultados similares al caso de utilizar L1 solamente.

A continuación se realizaron cambios en otros hiper-parámetros: En primer lugar se probó utilizando

otras funciones de activación diferente a la ya utilizada selu , concretamente relu, elu y leaky relu. Los mejores resultados se obtuvieron con la función relu con un 91.32 % en el conjunto de entrenamiento y un 83.45 % en el de validación.

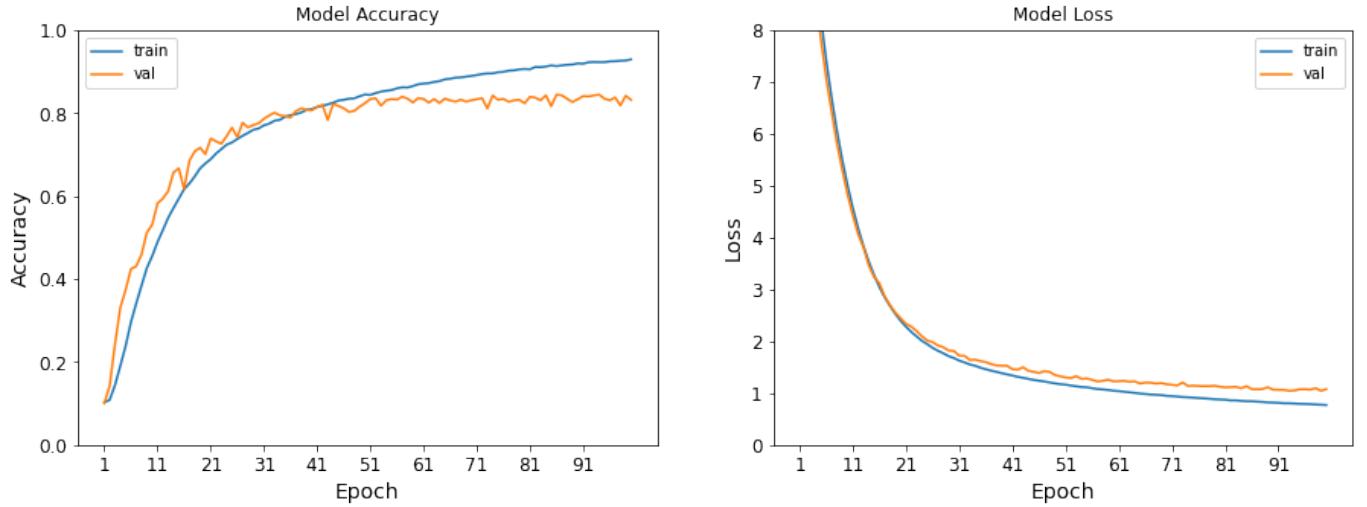


Figura 4.5: La evolución de la precisión y el error durante el proceso de entrenamiento

Como se había reducido el overfitting se probó a añadir mayor complejidad a la red, añadiendo capas de 512 filtros pero los resultados fueron inferiores a los que ya se habían obtenido. A continuación, se probó a modificar el learning rate con diferentes valores entre 0.1 y 0.0000001 , obteniendo que el mejor valor era el que ya se estaba utilizando : 0.00005.

Finalmente se aplicó la técnica de data-augmentation al conjunto de entrenamiento. Esta técnica aportó unos resultados muy interesantes, ya que eliminó completamente el overfitting. Posteriormente se intentó realizar cambios en la arquitectura para intentar reducir el bias pero no se obtuvieron resultados mejores, por lo que se tomó esta red como la final.

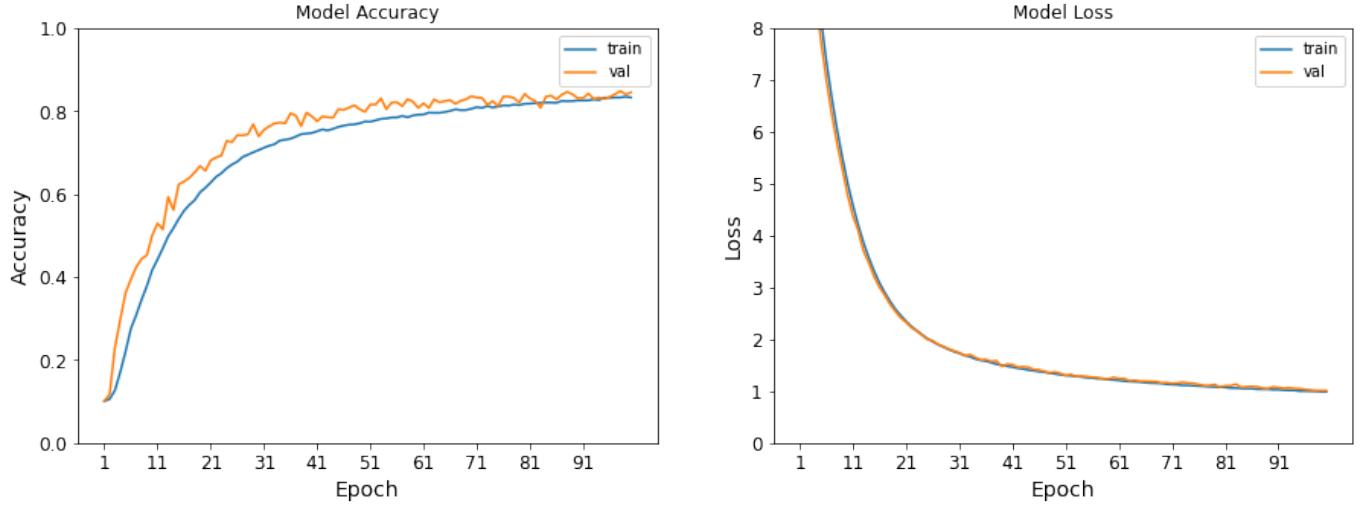


Figura 4.6: La evolución de la precisión y el error durante el proceso de entrenamiento

4.3. Dataset Señales Trafico

Una vez descargado el dataset, se han leído todas las imágenes por código y se han obtenido las imágenes con solo la parte relevante de la imagen donde se muestra la señal de tráfico. Solo trabajaremos con estas imágenes de partes relevantes. Después se han preprocesado los datos y se han creado 3 subconjuntos:

- **Training:** Subconjunto utilizado para entrenar el modelo neuronal. Estará compuesto por un total de 600 imágenes.
- **Validation:** Subconjunto utilizado para realizar el test de validación. Compuesto por 252 imágenes.
- **Test:** Subconjunto utilizado para realizar el test final. Compuesto por 361 imágenes.

A continuación se implementará una red neuronal alimentada hacia delante profunda y otra convolucional para ver el rendimiento de ambas redes con este dataset para clasificación de imágenes.

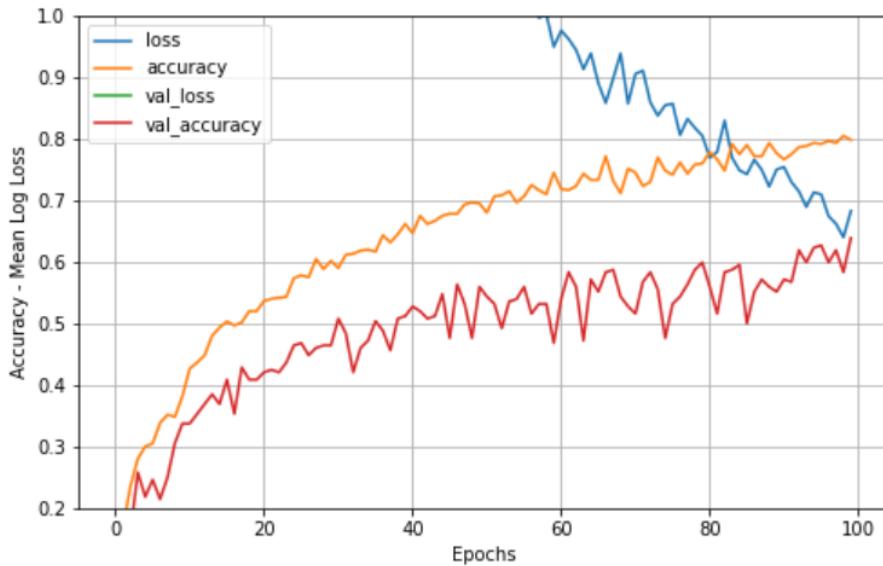
4.3.1. Red alimentada hacia delante profunda

Utilizaremos como modelo inicial la red proporcionada en el enunciado de la práctica. Los hiperparámetros de esta red se muestran en la tabla 4.3.

| Hiper-parámetros | Valor |
|---------------------------|---|
| Neuronas por capa | $[(224, 244), (224, 224), 80, 43]$ |
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 100 |
| m_b (tamaño del batch) | 16 |
| Función de activación | ReLU |
| Función de inicialización | No |
| Función de salida | softmax |
| Dropout | No |
| Optimizador | SGD ($lr = 0,001, momentum = 0,9, nesterov = True$) |
| Regularizador | No |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.3: Hiper-parámetros de la red inicial.

Los resultados de la red podemos verlos en la figura 4.7



```

loss    accuracy   val_loss   val_accuracy
99    0.682661  0.798333  1.84598      0.638889

```

Figura 4.7: Resultados de la red inicial

En los resultados se observa un alto bias y una alta varianza. Por tanto, lo primero que debemos hacer siguiendo el protocolo utilizado es reducir el bias.

Comenzaremos cambiando la función *ReLU* por la *ELU*. Seguidamente, instauraremos una normalización *He* para la inicialización de los pesos. Además, para reducir el bias añadiremos más capas y más neuronas, también se aumentará el tamaño del batch, y se aumentará el nº de épocas ya que en la gráfica se observa que el *Accuracy* puede mejorar con más entrenamiento. Esta configuración se puede ver en la siguiente tabla 4.4.

| Hiper-parámetros | Valor |
|---------------------------|---|
| Neuronas por capa | [(224, 244), (224, 224), 150, 100, 80, 43] |
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 500 |
| m_b (tamaño del batch) | 50 |
| Función de activación | ELU |
| Función de inicialización | he_normal |
| Función de salida | softmax |
| Dropout | No |
| Optimizador | SGD ($lr = 0.001, momentum = 0.9, nesterov = True$) |
| Regularizador | No |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.4: Hiper-parámetros - Cambio de parámetros 1

Los resultados obtenidos se pueden observar en la figura 4.8.

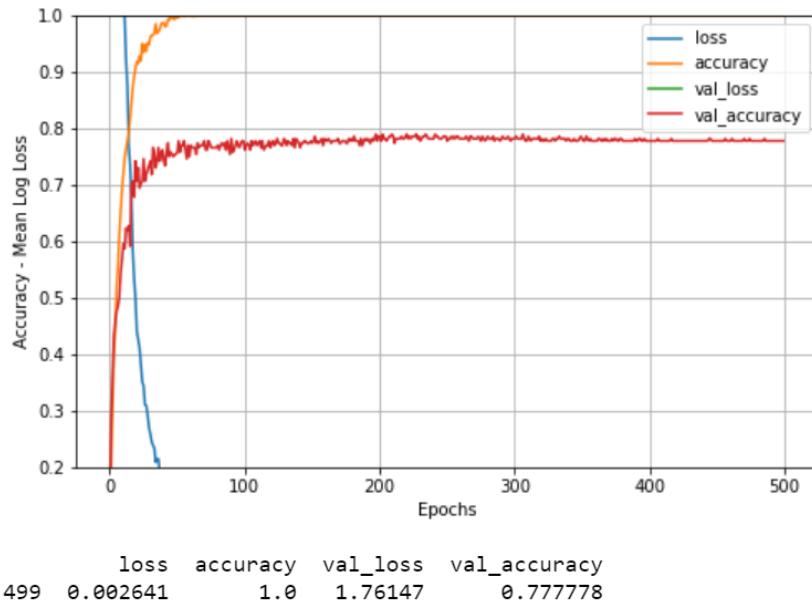


Figura 4.8: Resultados - Cambio de parámetros 1

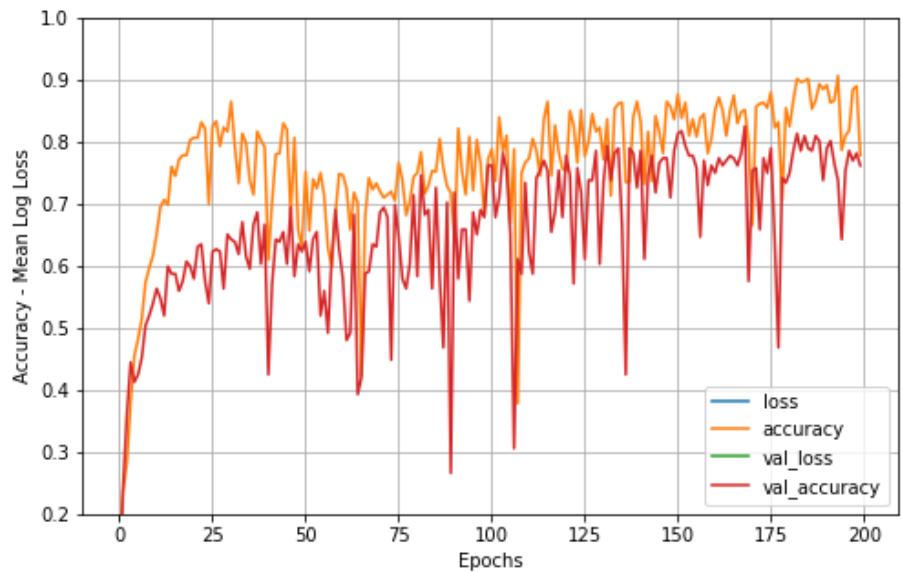
Observamos como el bias ahora es bajo y como la varianza se encuentra alta. Además, también se observa un *overfitting* que tendremos que reducir. Por tanto, trataremos de reducir la varianza y con ello el *overfitting*.

Para ello, vamos a introducir una regularización L1L2 para tratar de mantener los pesos pequeños. Además, reduciremos el número de épocas a 200. Esta configuración se puede ver en la siguiente tabla 4.5.

| Hiper-parámetros | Valor |
|---------------------------|---|
| Neuronas por capa | [(224, 244), (224, 224), 150, 100, 80, 43] |
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 200 |
| m_b (tamaño del batch) | 50 |
| Función de activación | ELU |
| Función de inicialización | he_normal |
| Función de salida | softmax |
| Dropout | No |
| Optimizador | SGD ($lr = 0.001, momentum = 0.9, nesterov = True$) |
| Regularizador | L1_L2 |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.5: Hiper-parámetros - Cambio de parámetros 2

Los resultados obtenidos se pueden observar en la figura 4.9.



```

loss      accuracy   val_loss    val_accuracy
199  4.496962  0.778333  4.692275      0.761905

```

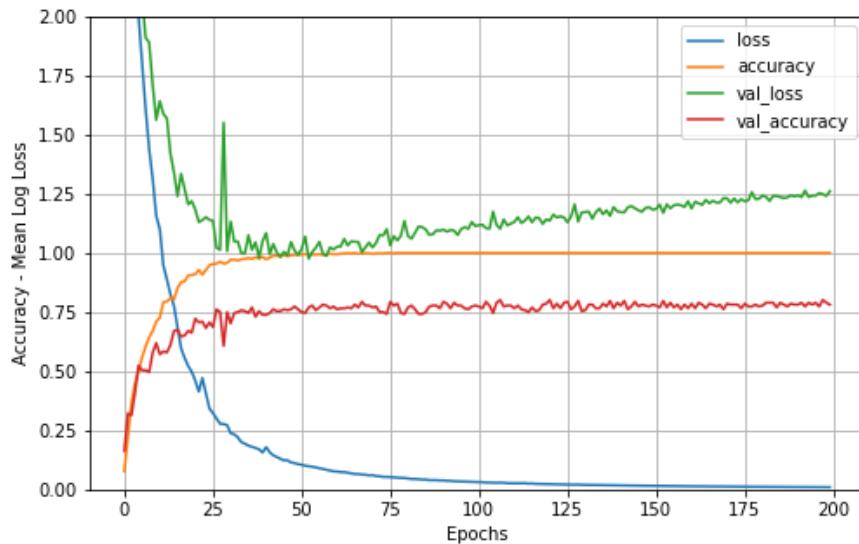
Figura 4.9: Resultados - Cambio de parámetros 3

Los resultados obtenidos son desastrosos. Así que decimos cambiar el regularizador por un dropout. Esta configuración se puede ver en la siguiente tabla 4.6.

| Hiper-parámetros | Valor |
|---------------------------|---|
| Neuronas por capa | [(224, 244), (224, 224), 150, 100, 80, 43] |
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 200 |
| m_b (tamaño del batch) | 50 |
| Función de activación | ELU |
| Función de inicialización | he_normal |
| Función de salida | softmax |
| Dropout | [0.8,0.4,0.2] |
| Optimizador | SGD ($lr = 0.001, momentum = 0.9, nesterov = True$) |
| Regularizador | No |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.6: Hiper-parámetros - Cambio de parámetros 3

Los resultados obtenidos se pueden observar en la figura 4.10.



```

loss  accuracy  val_loss  val_accuracy
199  0.009395      1.0  1.261652      0.781746

```

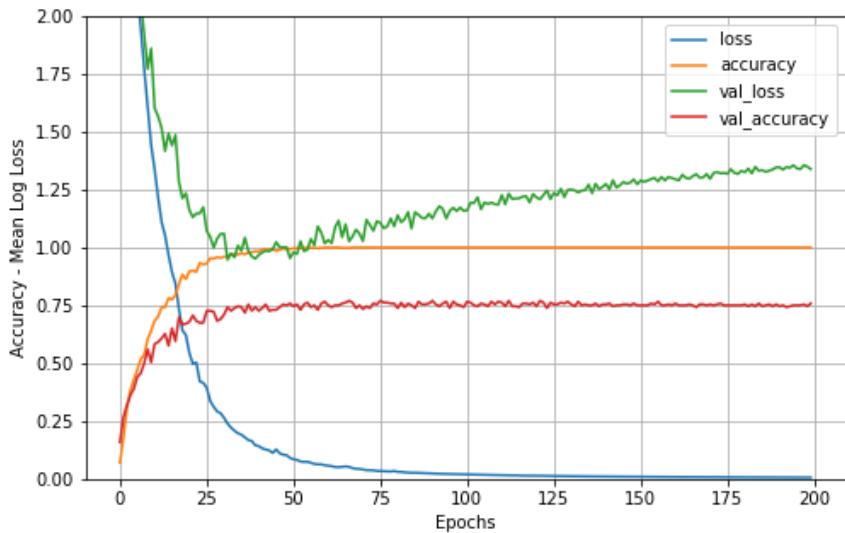
Figura 4.10: Resultados - Cambio de parámetros 3

La varianza sigue siendo alta y además se observa *overfitting* aproximadamente a partir de la época 75. Para reducir esta varianza se añadirá una capa más para tratar de que generalice mejor. Esta configuración se puede ver en la siguiente tabla 4.7.

| Hiper-parámetros | Valor |
|---------------------------|---|
| Neuronas por capa | [(224, 244), (224, 224), 150, 100, 80, 80, 43] |
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 200 |
| m_b (tamaño del batch) | 50 |
| Función de activación | ELU |
| Función de inicialización | he_normal |
| Función de salida | softmax |
| Dropout | [0.8, 0.4, 0.2, 0] |
| Optimizador | SGD ($lr = 0.001, momentum = 0.9, nesterov = True$) |
| Regularizador | No |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.7: Hiper-parámetros - Cambio de parámetros 4

Los resultados obtenidos se pueden observar en la figura 4.11.



```

loss    accuracy   val_loss   val_accuracy
199  0.005637      1.0  1.338921     0.757937

```

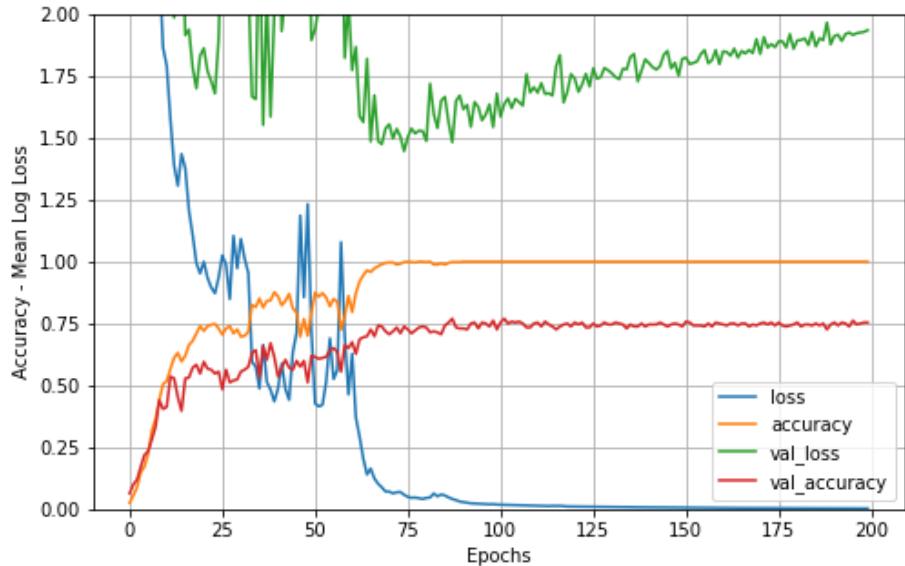
Figura 4.11: Resultados - Cambio de parámetros 4

Los resultados tampoco son mucho mejores al modelo anterior. Aunque el bias esté bajo, vamos a tratar de probar con un optimizador mejor, con Adam para observar si tenemos mejores resultados. Esta configuración se puede ver en la siguiente tabla 4.8.

| Hiper-parámetros | Valor |
|---------------------------|--|
| Neuronas por capa | [(224, 244), (224, 224), 150, 100, 80, 43] |
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 200 |
| m_b (tamaño del batch) | 50 |
| Función de activación | ELU |
| Función de inicialización | he_normal |
| Función de salida | softmax |
| Dropout | [0.8,0.4,0.2,0] |
| Optimizador | Adam($lr = 0.001, beta_1 = 0.9, beta_2 = 0.999$) |
| Regularizador | No |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.8: Hiper-parámetros - Cambio de parámetros 5

Los resultados obtenidos se pueden observar en la figura 4.12.



```

loss  accuracy  val_loss  val_accuracy
199  0.002444      1.0  1.935678      0.753968

```

Figura 4.12: Resultados - Cambio de parámetros 5

Los resultados obtenidos empeoran el modelo, así que deshechamos esta idea. Trataremos de normalizar el batch para tratar de disminuir la varianza, esto lo haremos sobre la red obtenida con el cambio de parámetros 3 de la tabla 4.6. Por tanto, esta nueva configuración se puede ver en la siguiente tabla 4.9.

| Hiper-parámetros | Valor |
|---------------------------|---|
| Neuronas por capa | [(224, 244), (224, 224), 150, 100, 80, 43] |
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 200 |
| m_b (tamaño del batch) | 50 |
| Función de activación | ELU |
| Función de inicialización | he_normal |
| Función de salida | softmax |
| Dropout | [0.8,0.4,0.2] |
| Optimizador | SGD ($lr = 0.001, momentum = 0.9, nesterov = True$) |
| Regularizador | No |
| Normalización | BatchNormalization |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.9: Hiper-parámetros - Cambio de parámetros 6

Los resultados obtenidos se pueden observar en la figura 4.13.

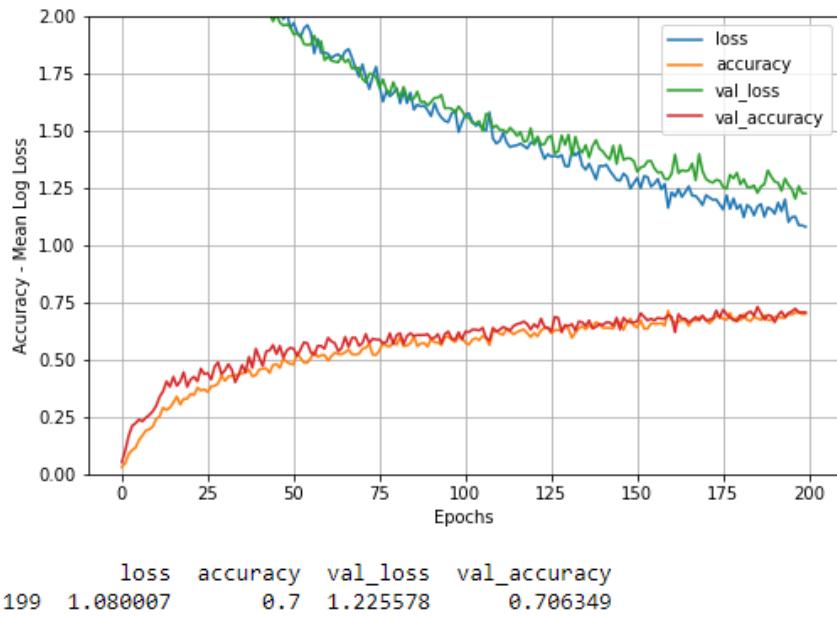


Figura 4.13: Resultados - Cambio de parámetros 6

Empeora también los resultados, por tanto también deshechamos esta configuración.

Después de varios modelos, concluimos que el mejor modelo que hemos alcanzado es el modelo 3. Para ver si podemos mejorar este modelo reduciendo el *overfitting*, vamos a reducir el número de épocas a 100. Además, también aumentamos el tamaño del batch a 100 para ver si así reducimos la varianza. Esta nueva configuración se puede ver en la siguiente tabla 4.10.

| Hiper-parámetros | Valor |
|---------------------------|---|
| Neuronas por capa | [(224, 244), (224, 224), 150, 100, 80, 43] |
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 100 |
| m_b (tamaño del batch) | 100 |
| Función de activación | ELU |
| Función de inicialización | he_normal |
| Función de salida | softmax |
| Dropout | [0.8,0.4,0.2] |
| Optimizador | SGD ($lr = 0.001, momentum = 0.9, nesterov = True$) |
| Regularizador | No |
| Normalización | BatchNormalization |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.10: Hiper-parámetros - Cambio de parámetros 7

Los resultados obtenidos se pueden observar en la figura 4.14.

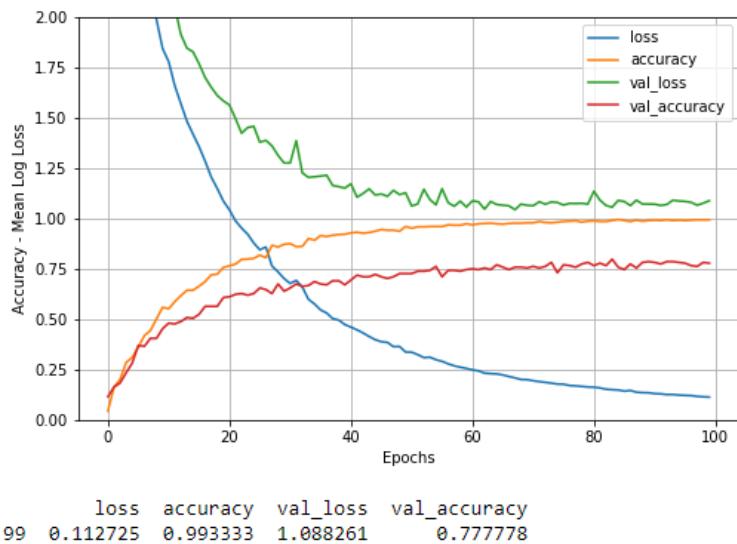


Figura 4.14: Resultados - Cambio de parámetros 7

Los resultados obtenidos se adecuan más a lo que buscamos aunque la varianza sigue siendo alta. Esto puede deberse a la poca cantidad de datos de los que disponemos, ya que si tuviéramos más datos el modelo tendría más capacidad de generalizar. Aún así, tenemos un bias bajo y prácticamente no tenemos *overfitting*. También, en este modelo hemos conseguido la pérdida de validación más baja, aunque esta sigue siendo alta, pero al menos se ha reducido. Por tanto, se ha decidido tomar esta última red como la red final.

4.3.2. Red Convolucional

Utilizaremos como modelo inicial la red convolucional proporcionada al final del tema *Convolutional Neural Networks (I): Fundamentals*. La estructura de esta red se pueden observar en la figura 4.15 y los hiper-parámetros utilizados en 4.11.

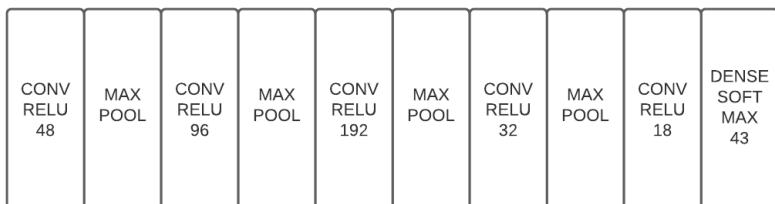


Figura 4.15: Estructura de la red inicial

| Hiper-parámetros | Valor |
|---------------------------|---|
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 30 |
| m_b (tamaño del batch) | 32 |
| Función de activación | ReLU |
| Función de inicialización | No |
| Función de salida | softmax |
| Dropout | No |
| Optimizador | SGD ($lr = 0.001, momentum = 0.9, nesterov = True$) |
| Regularizador | No |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.11: Hiper-parámetros de la red inicial.

Los resultados de la red podemos verlos en la figura 4.16

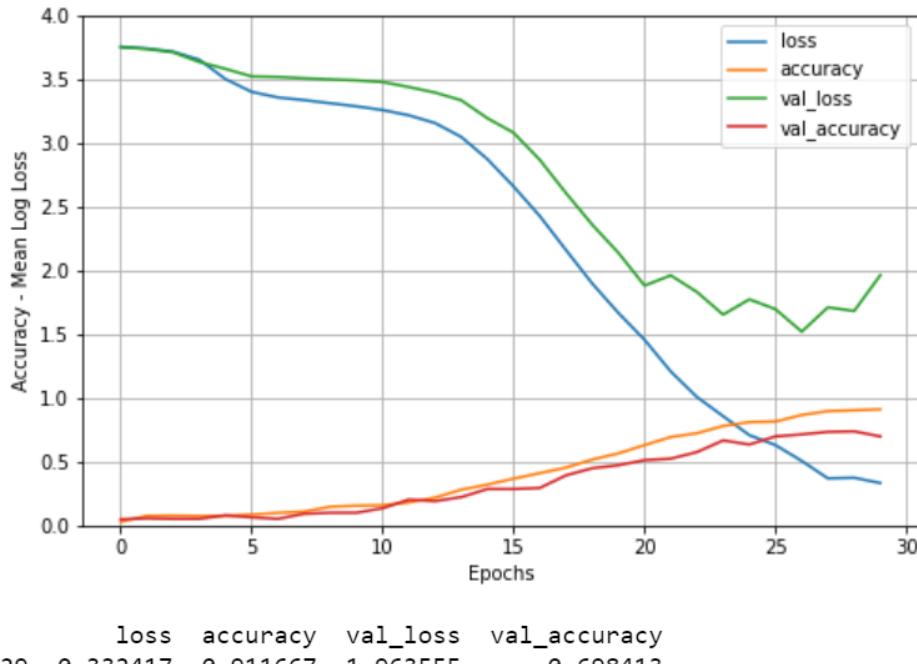


Figura 4.16: Resultados de la red inicial

Lo primero que vamos a tratar es de reducir el bias. Para ello, vamos a añadir más capas. Vamos a duplicar cada capa.

Esta nueva estructura se puede ver en la figura 4.17 y los hiper-parámetros son los mismos que en la tabla 4.11.

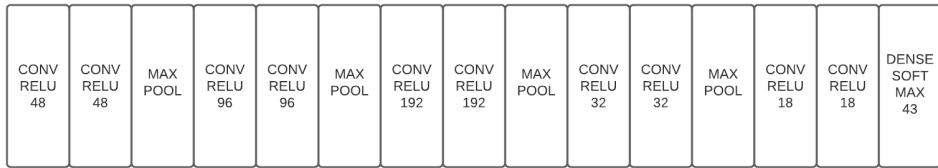


Figura 4.17: Estructura de la red - Cambio 1

Los resultados de la red podemos verlos en la figura 4.18

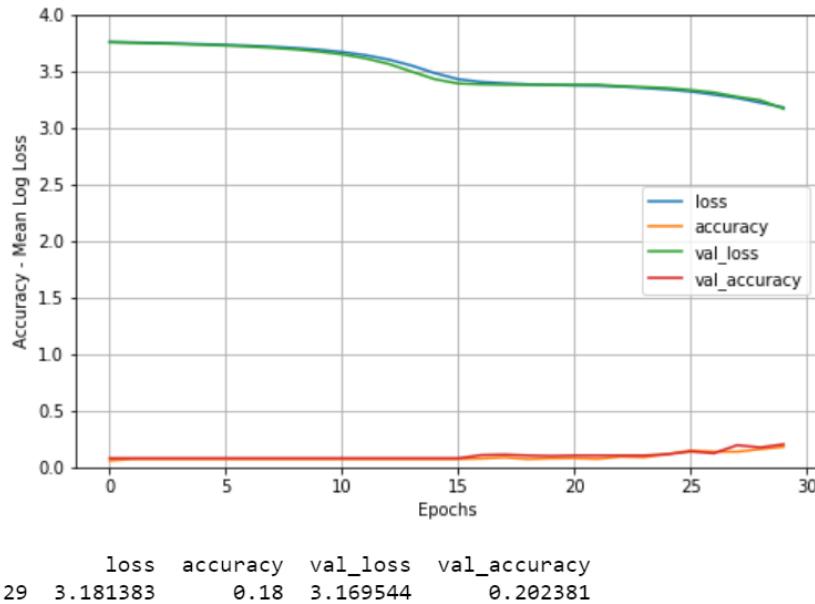


Figura 4.18: Resultados de la red - Cambio 1

Los resultados obtenidos no son nada buenos. El bias empeora drásticamente y se vuelve muy alto. Por tanto, seguiremos con la estrategia de duplicar, pero reduciremos capas finales de convolución. La estructura que utilizaremos a continuación se puede observar en la figura 4.19 y los hiper-parámetros seguirán siendo los indicados en la tabla 4.11.

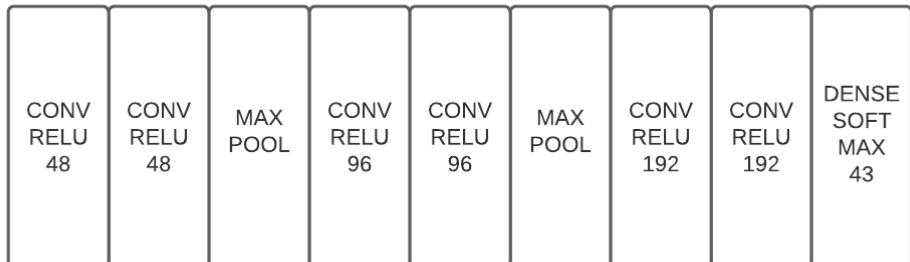


Figura 4.19: Estructura de la red - Cambio 2

Los resultados de la red se pueden ver en la figura 4.20.

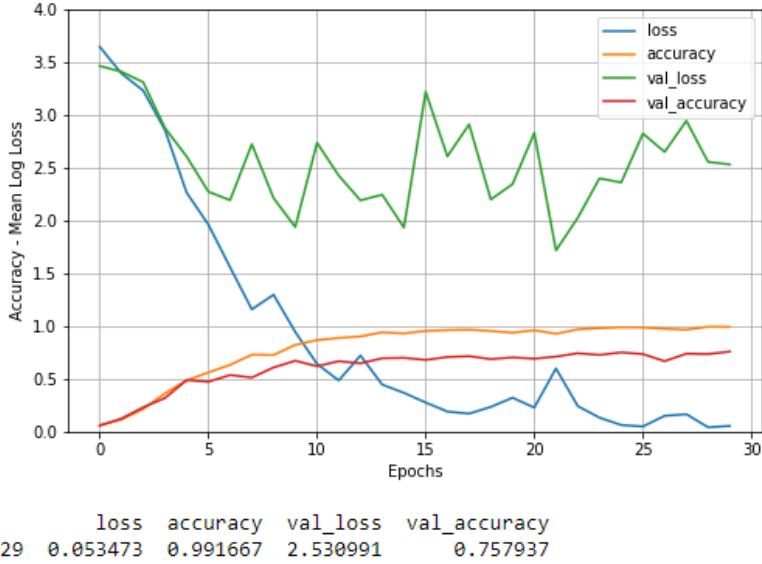


Figura 4.20: Resultados de la red - Cambio 2

Hemos obtenido una mejora notable en el bias, ya no se encuentra alto. Pero la varianza sigue siendo alta, por ello, vamos a hacer regularización. Además, se observa que se ha dado *overfitting* (sobreentrenamiento). El primer método de regularización que vamos a utilizar es aumento de los datos (*Data augmentation*) para que la red generalice mejor. Por tanto, la estructura será la misma que en 4.19 y los hiper-parámetros se pueden ver en la tabla 4.12.

| Hiper-parámetros | Valor |
|---------------------------|--|
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 30 |
| m_b (tamaño del batch) | 32 |
| Función de activación | ReLU |
| Función de inicialización | No |
| Función de salida | softmax |
| Dropout | No |
| Optimizador | SGD ($lr = 0,001, momentum = 0,9, nesterov = True$) |
| Regularizador | Sí, <i>Data augmentation</i> : <i>featurewise_center = True</i> , <i>featurewise_std_normalization = True</i> , <i>rotation_range = 20</i> , <i>width_shift_range = 0,2</i> , <i>height_shift_range = 0,2</i> , <i>zoom_range = 0,2</i> , <i>shear_range = 0,1</i> , <i>horizontal_flip = True</i>) |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.12: Hiper-parámetros de la red - Cambio 3

Los resultados de la red se pueden ver en la figura 4.21.

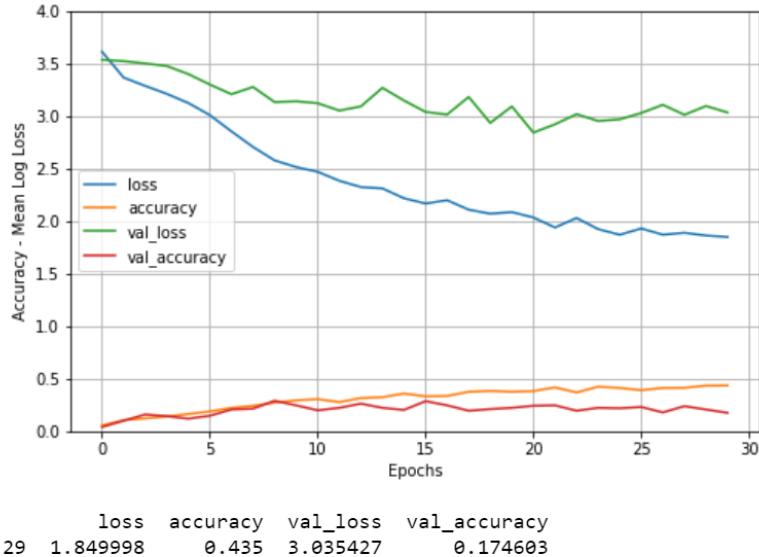


Figura 4.21: Resultados de la red - Cambio 3

Observamos como ahora obtenemos un alto bias. Para reducirlo vamos a cambiar SDG por Adam y además entrenaremos más tiempo. Además, se probarán diversas configuraciones de los hiper-parámetros del aumento de datos.

Después de probar con varias configuraciones para los hiper-parámetros del aumento de datos se ha seleccionado la configuración que mejor resultado aporta.

Por tanto, la estructura será la misma que en 4.19 y los hiper-parámetros se pueden ver en la tabla 4.13.

| Hiper-parámetros | Valor |
|---------------------------|---|
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 50 |
| m_b (tamaño del batch) | 32 |
| Función de activación | ReLU |
| Función de inicialización | No |
| Función de salida | softmax |
| Dropout | No |
| Optimizador | Adam ($lr = 0,001, beta_1 = 0,9, beta_2 = 0,999$) |
| Regularizador | Sí, <i>Data augmentation:</i> <i>featurewise_center = False,</i> <i>featurewise_std_normalization = False,</i> <i>rotation_range = 10.,</i> <i>width_shift_range = 0,1,</i> <i>height_shift_range = 0,1,</i> <i>zoom_range = 0,3,</i> <i>shear_range = 0,1,</i> <i>horizontal_flip = True</i>) |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.13: Hiper-parámetros de la red - Cambio 4

Los resultados de la red se pueden ver en la figura 4.22.

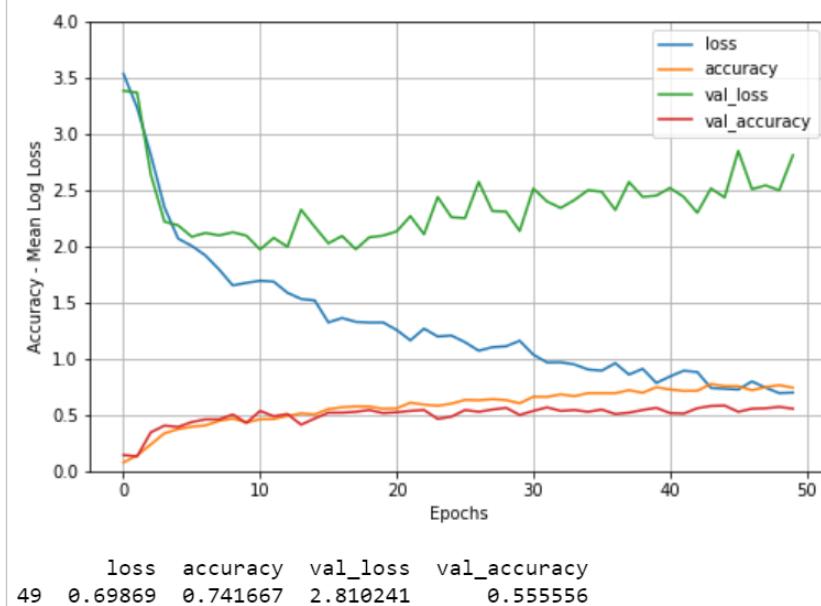


Figura 4.22: Resultados de la red - Cambio 4

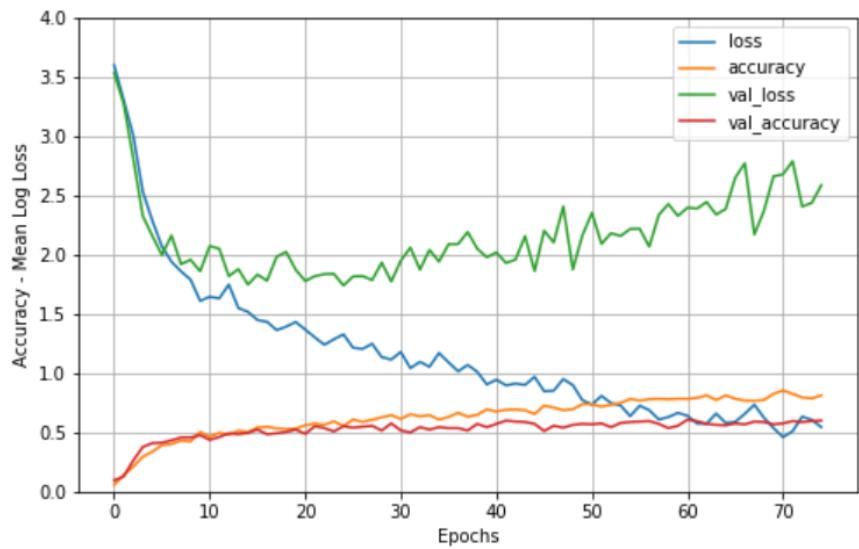
Se observa como, en comparación con la anterior red, la precisión aumenta e incluso que en la época 30 la precisión también es mejor. La tendencia de la precisión es hacia arriba, así que en el siguiente

paso entrenaremos más épocas, además, para lidiar con el sobreentrenamiento introduciremos Dropout. La estructura de esta nueva red será la misma que en 4.19 y los hiper-parámetros se pueden ver en la tabla 4.14.

| Hiper-parámetros | Valor |
|---------------------------|---|
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 75 |
| m_b (tamaño del batch) | 32 |
| Función de activación | ReLU |
| Función de inicialización | No |
| Función de salida | softmax |
| Dropout | Sí [0.1,0.2, 0.2] |
| Optimizador | Adam ($lr = 0,001, beta_1 = 0,9, beta_2 = 0,999$) |
| | Sí, <i>-Data augmentation:</i> $featurewise_center = False,$ $featurewise_std_normalization = False,$ $rotation_range = 10.,$ $width_shift_range = 0,1,$ $height_shift_range = 0,1,$ $zoom_range = 0,3,$ $shear_range = 0,1,$ $horizontal_flip = True$) <i>-Dropout</i> |
| Regularizador | No |
| Normalización | categorical_crossentropy |
| Función loss | |
| Métrica | accuracy |

Tabla 4.14: Hiper-parámetros de la red - Cambio 5

Los resultados de la red se pueden ver en la figura 4.23.



| | loss | accuracy | val_loss | val_accuracy |
|----|----------|----------|----------|--------------|
| 74 | 0.544066 | 0.811667 | 2.586539 | 0.599206 |

Figura 4.23: Resultados de la red - Cambio 5

Se observa como la precisión ha aumentado, y cómo el sobreentrenamiento se ha reducido. Como el bias sigue siendo alto, para reducirlo, se realizarán más épocas ya que se ve una tendencia hacia arriba en la precisión. Por otro lado, para reducir la varianza y el sobreentrenamiento se añadirá un regularizador l2.

Así, la estructura de esta nueva red será la misma que en 4.19 y los hiper-parámetros se pueden ver en la tabla 4.15.

| Hiper-parámetros | Valor |
|---------------------------|--|
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 100 |
| m_b (tamaño del batch) | 32 |
| Función de activación | ReLU |
| Función de inicialización | No |
| Función de salida | softmax |
| Dropout | Sí [0.1,0.2, 0.2] |
| Optimizador | Adam ($lr = 0,001, beta_1 = 0,9, beta_2 = 0,999$) |
| | Sí, <i>-Data augmentation:</i> $featurewise_center = False,$ $featurewise_std_normalization = False,$ $rotation_range = 10.,$ $width_shift_range = 0,1,$ $height_shift_range = 0,1,$ $zoom_range = 0,3,$ $shear_range = 0,1,$ $horizontal_flip = True$) |
| Regularizador | <i>-Dropout</i> <i>-L2(weight_decay=1e-4)</i> |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.15: Hiper-parámetros de la red - Cambio 6

Los resultados de la red se pueden ver en la figura 4.24.

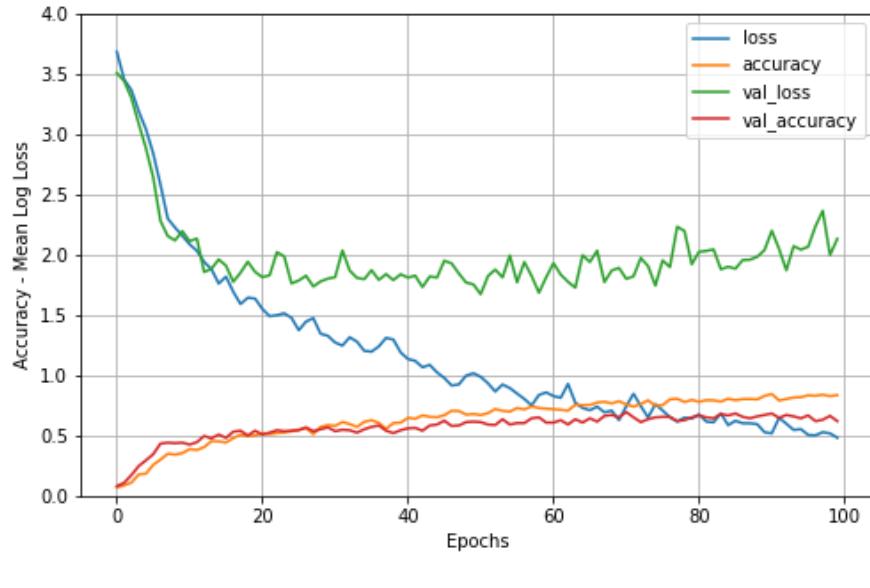


Figura 4.24: Resultados de la red - Cambio 6

Se ha reducido el bias y la varianza. Además, el sobreentrenamiento se ha vuelto más estable.

Por ahora, este es el modelo que más nos convence.

Para intentar mejorar el modelo se ha introducido BatchNormalization (en los hiperparámetros 4.15 con la arquitectura 4.19) , pero no se han obtenido mejores resultados (figura 4.25).

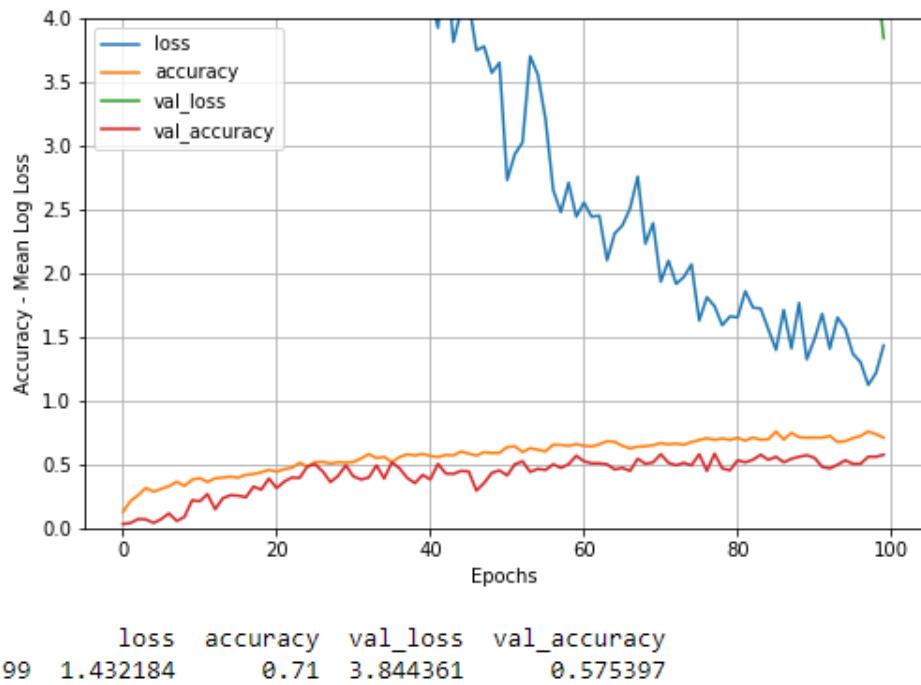


Figura 4.25: Resultados de la red - Cambio 7

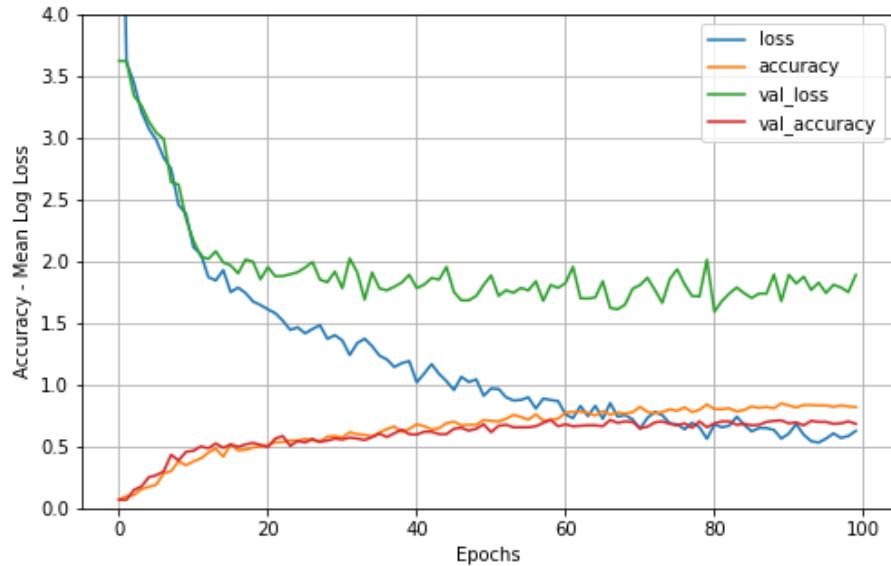
Por tanto, se ha probado finalmente a añadir inicialización He, al modelo del cambio 6 4.15.

Así, la estructura de esta nueva red será la misma que en 4.19 y los hiper-parámetros se pueden ver en la tabla 4.16.

| Hiper-parámetros | Valor |
|---------------------------|---|
| Tasa de aprendizaje | 0.001 |
| Nº de épocas | 100 |
| m_b (tamaño del batch) | 32 |
| Función de activación | ReLU |
| Función de inicialización | Sí, He_normal |
| Función de salida | softmax |
| Dropout | Sí [0.1,0.2, 0.2] |
| Optimizador | Adam ($lr = 0,001, beta_1 = 0,9, beta_2 = 0,999$) |
| Regularizador | Sí, <i>-Data augmentation:</i> $featurewise_center = False,$ $featurewise_std_normalization = False,$ $rotation_range = 10.,$ $width_shift_range = 0,1,$ $height_shift_range = 0,1,$ $zoom_range = 0,3,$ $shear_range = 0,1,$ $horizontal_flip = True)$ <i>-Dropout</i> <i>-L2(weight_decay=1e-4)</i> |
| Normalización | No |
| Función loss | categorical_crossentropy |
| Métrica | accuracy |

Tabla 4.16: Hiper-parámetros de la red - Cambio 8

Los resultados de la red se pueden ver en la figura 4.26.



```
loss    accuracy   val_loss   val_accuracy
99  0.625724  0.818333  1.88994      0.68254
```

Figura 4.26: Resultados de la red - Cambio 8

Observamos como obtenemos unos resultados muy parecidos a los del modelo del cambio 6 (figura 4.24). Obtenemos precisiones muy parecidas. Pero mucho menos sobreentrenamiento a lo largo de todo el entrenamiento, y además reducimos los *loss*. Por tanto, se ha decidido tomar esta última red como red final.

Capítulo 5

Conclusiones

5.1. Dataset CIFAR-10

Hemos observado como una FFNN no consigue resolver el problema de forma aceptable, mientras que una CNN que consigue unos resultados mucho más confiables.

La FNN obtenida, además de poseer un gran bias posee una varianza considerable, mientras que la CNN aunque posee un ligero bias, su varianza es prácticamente nula.

Los motivos de estos resultados se deben principalmente a que es un problema con una cantidad baja de clases pero con un alto número de imágenes de cada una de las clases, que además es aumentado gracias al *data augmentation*. Concluimos que una CNN es más adecuada para la resolución de este problema.

5.2. Dataset Señales Tráfico

En el dataset de señales de tráfico se han obtenido mejores resultados en la red alimentada hacia delante profunda que en la red convolucional. Esto ha podido ser debido a la estructura elegida para crear la red convolucional, aunque se ha probado con varias estructuras y no ha habido mejora. Por otro lado, el entrenamiento de la red convolucional ha sido muy tedioso ya que tardaba muchísimo tiempo en entrenar. En cuanto a los picos de sobreentramiento de la red convolucional, se deben a las distintas imágenes que muestra a la red el *data augmentation*.

Ambas redes finales ffNN y CNN tienen altas varianzas porque es un problema que tiene que clasificar en 43 clases diferentes, eso son muchas y de ahí su dificultad.

Concluimos que para el dataset de señales de tráfico la mejor red es la ffNN ya que produce resultados mejores(en los 3 test) e incluso realiza el entrenamiento en mucho menor tiempo.

5.3. Comparación ambos datasets

Los resultados de la ffNN del dataset CIFAR-10 son bastante malos en comparación a los resultados del dataset de las Señales de Tráfico, ya que en el primero se ha encontrado precisiones cercanas al 50 % y las del segundo sobrepasaban el 70 %. La ffNN en el dataset de Señales de Tráfico ha proporcionado un rendimiento bastante bueno, aunque es cierto que le cuesta generalizar.

En cuanto a las redes convolucionales, el dataset CIFAR-10 ha obtenido mejores resultados que el dataset de las Señales de Tráfico. En el primero la precisión más baja encontrada ha sido un 83.61 %, mientras que en el segundo ha sido de 68.25 %. Esto puede deberse a la dificultad del dataset de Señales de Tráfico ya que tiene un gran número de clases que clasificar.

Pero, ¿a qué se debe que la ffNN haya funcionado mejor en el dataset de Señales de tráfico y que la CNN en el CIFAR-10? Esto debe a los tamaños de los datasets. Las ffNN funcionan mejor para tamaños pequeños, como lo es de señales de tráfico (900 imágenes). En comparación a las CNN que funcionan mucho mejor para datasets grandes como el CIFAR-10 (60000 imágenes).