Taller 6 - Intro a Redes Neuronales
Carlos Gómez – 202111593
Jerónimo Vargas – 202113305

1. Regresión con redes neuronales:

1.1

En este caso se utilizó un modelo que solo tiene variables continuas o tipo flotantes. Las variables que cuentan con esta característica son "Displacement", "Horsepower", "Weight" y "Acceleration".

En este caso, para el modelo base se tienen las siguientes especificaciones:

- Capa de normalización para las 4 variables explicativas.
- 2 capas densas de 64 neuronas cada una.
- Una capa de respuesta donde se retorna la variable respuesta.
- La función ReLu para la activación.

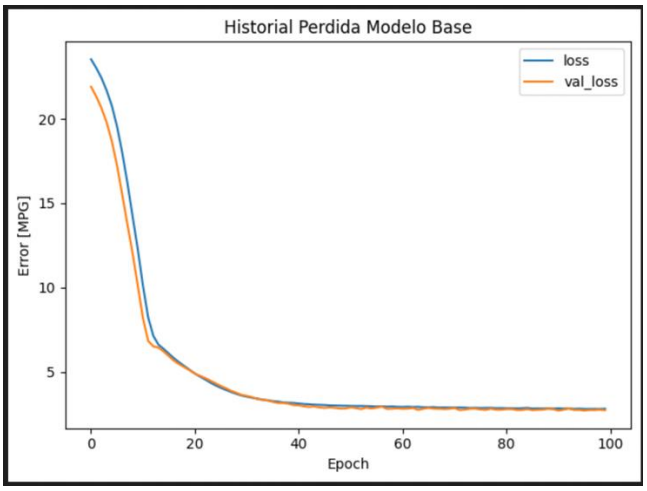
Aspectos para resaltar del modelo:

Layer (type)	Output Shape	Param #
normalization (Normalization)	(None, 4)	9
dense (Dense)	(None, 64)	320
dense_1 (Dense)	(None, 64)	4,160
dense_2 (Dense)	(None, 1)	65

Summary modelo Base.

- La capa de normalización tiene 9 parámetros que son 2 por variable (media y varianza) y el parámetro de intercepto o neurona ficticia.
- La primera capa densa tiene 320 parámetros que son 4 neuronas de normalización para cada neurona por separado, lo que sería 64 multiplicado por 4 y adicionalmente 64 parámetros debido al intercepto.
- La segunda capa densa tiene 4160 parámetros que son 64 neuronas de la capa 1 para cada neurona de la capa 2, lo cual sería 64 multiplicado por 64, mas 64 parámetros del intercepto.
- La capa de respuesta que tiene 65 parámetros que se distribuyen en 64 parámetros para cada neurona de la capa 2 más el intercepto.

1.2



En la gráfica anterior se puede observar cómo después de que se entrena el modelo base, el error absoluto medio (MAE) es bastante alto con valores superiores a 20. No obstante, una vez que las épocas aumentan el optimizador permite bajar el MAE hasta un valor cercano a 2.7. Asimismo, se puede ver como después de la época 50 el valor del MAE encuentra una asíntota sobre el valor antes mencionado.

1.3

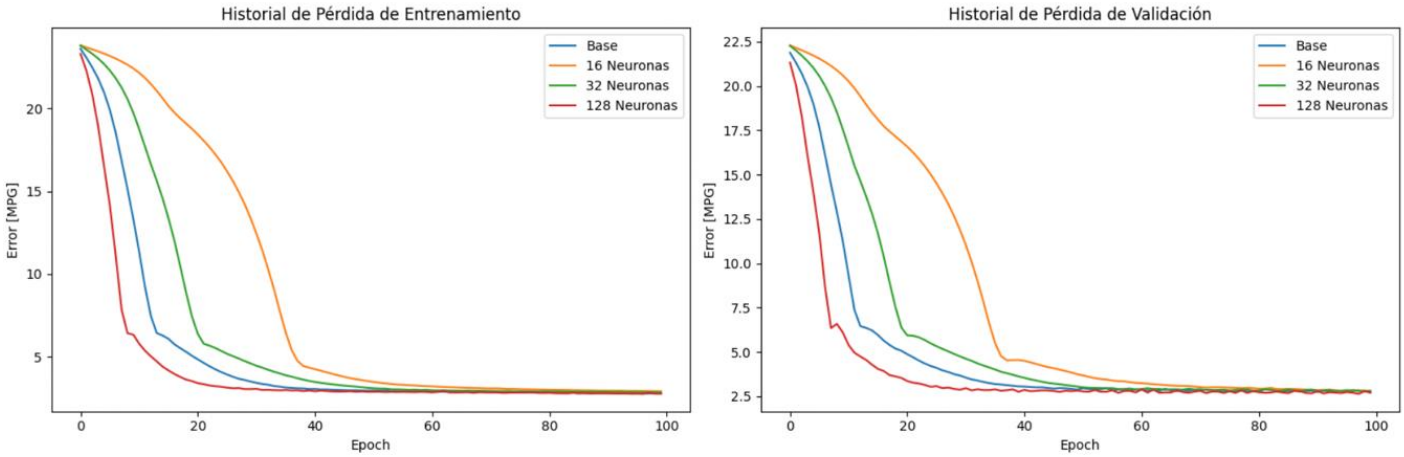
Como se mencionó anteriormente, el modelo base cuenta con dos capas de 64 neuronas. En esta sección del taller, se mantendrán estas dos capas densas, pero se modificará la cantidad de neuronas en cada una. Primero, se utilizarán 16 neuronas en cada capa, luego 32, y finalmente 128. Con estas configuraciones, se procederá a entrenar los modelos y se graficarán, por un lado, los historiales de la pérdida durante el entrenamiento y, por otro lado, los de la pérdida durante la validación para comparar los resultados. Además, se mostrará el tiempo necesario para entrenar cada modelo y la memoria requerida.

Modelo	CPU time (s)	Wall time (s)	Memoria (KB)
Base	6,08	10,5	17,79
16 neuronas	6,56	11	1,48
32 neuronas	7,23	13,1	4,92
128 neuronas	5,73	8,93	67,54

En primera instancia, se puede concluir que a medida que aumenta el número de neuronas aumenta el consumo de memoria (KB), no obstante, esto también afecta la capacidad del modelo para capturar patrones complejos en los datos, lo que puede incrementar el error.

En cuanto al CPU time y Wall time, se observa que el modelo con 128 neuronas es el más eficiente, con el menor tiempo de CPU (5,73 segundos) y el Wall time más bajo (8,93 segundos), superando incluso al modelo base. Aunque los modelos con 16 y 32 neuronas tienen menos capacidad, presentan tiempos más altos tanto en CPU (6,56 y 7,23 segundos)

como en Wall time (11 y 13,1 segundos), lo que sugiere una menor eficiencia en comparación. En resumen, el modelo con 128 neuronas logra optimizar tanto el uso del CPU como el tiempo total de ejecución, mientras que las configuraciones intermedias no parecen tan eficientes.



Graficas Perdida de entrenamiento y Perdida de validación respectivamente pata modelos que diferentes números de neuronas.

Las gráficas presentadas muestran el comportamiento del error durante el proceso de entrenamiento y validación de un modelo entrenado con diferentes configuraciones neuronales (base, 16, 32, y 128 neuronas). En ambas gráficas, se observa una rápida disminución del error durante las primeras 20 épocas, con una posterior estabilización, lo que indica que el modelo aprende rápidamente al inicio, para luego refinarse de forma más gradual.

En el gráfico de entrenamiento, se aprecia que los modelos con mayor cantidad de neuronas, especialmente el de 128 neuronas, logran una reducción más acelerada del error en comparación con los modelos de menos neuronas. Esto sugiere que un modelo con mayor capacidad (más neuronas) es más eficaz para ajustar los datos de entrenamiento, reduciendo el error de manera más eficiente. Sin embargo, todos los modelos tienden a estabilizarse a partir de las 60 épocas, con errores muy bajos al final del entrenamiento.

Por otro lado, en la gráfica de validación se observa un comportamiento similar, aunque el error de validación es inicialmente mayor que el de entrenamiento. El modelo con 128 neuronas también logra el mejor rendimiento en validación, alcanzando el menor error en menor cantidad de épocas. Sin embargo, la diferencia entre los modelos tiende a disminuir con el tiempo, convergiendo todos ellos a un error final bastante similar tras las 60 épocas.

1.4 Modificación en los números de capa en la red neuronal.

Modelo 1 capa:

Layer (type)	Output Shape	Param #
normalization_1 (Normalization)	(None, 4)	9
dense_52 (Dense)	(None, 64)	320
dense_53 (Dense)	(None, 1)	65

Summary modelo 1 Capa.

Modelo 2 capa:

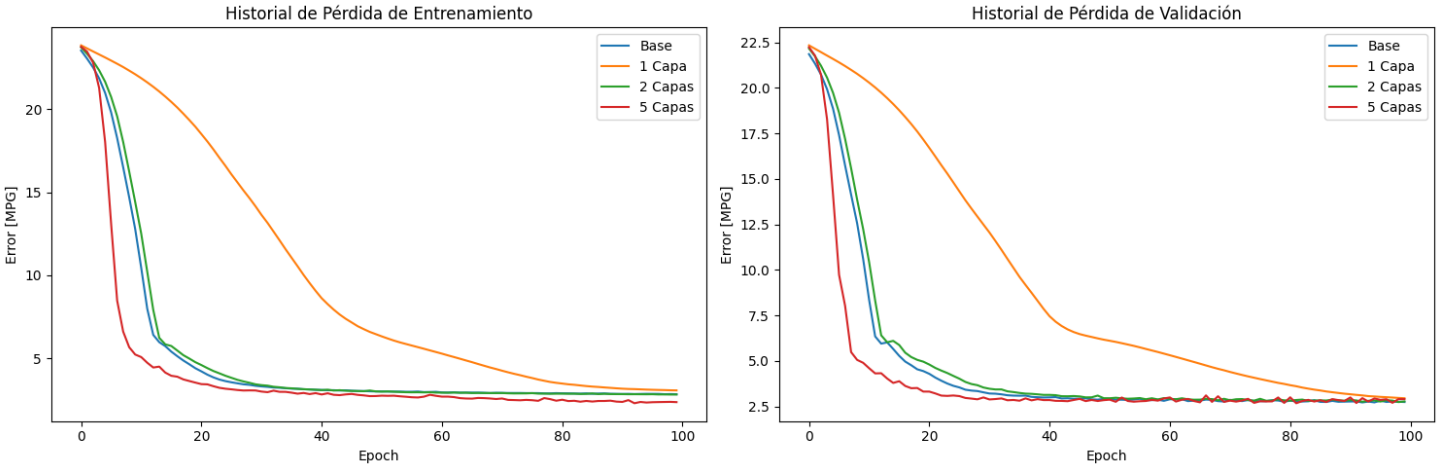
Layer (type)	Output Shape	Param #
normalization_1 (Normalization)	(None, 4)	9
dense_54 (Dense)	(None, 64)	320
dense_55 (Dense)	(None, 64)	4,160
dense_56 (Dense)	(None, 2)	130

Summary modelo 2 Capas.

Modelo 5 capa:

Layer (type)	Output Shape	Param #
normalization_1 (Normalization)	(None, 4)	9
dense_57 (Dense)	(None, 64)	320
dense_58 (Dense)	(None, 64)	4,160
dense_59 (Dense)	(None, 64)	4,160
dense_60 (Dense)	(None, 64)	4,160
dense_61 (Dense)	(None, 5)	325

Summary modelo 5 Capas.



Graficas Perdida de entrenamiento y Perdida de validación respectivamente de modelos de diferentes capas.

Curva de pérdida de entrenamiento

Los modelos con 1 capa y 2 capas convergen rápidamente, con el modelo de 2 capas alcanzando una pérdida ligeramente menor. El modelo de 5 capas tiene una curva de entrenamiento más lenta y presenta una pérdida más alta en las primeras épocas, lo que sugiere que le toma más tiempo ajustarse a los datos.

Curva de pérdida de validación

Las curvas de validación siguen un patrón similar al de las curvas de entrenamiento. El modelo de 2 capas es el que mejor generaliza en el conjunto de validación, mostrando una caída rápida de la pérdida y una buena estabilización. El modelo de 5 capas muestra un rendimiento peor en validación, lo que indica un posible sobreajuste. Aunque tiene una mayor capacidad de aprendizaje, no logra generalizar bien a datos no vistos.

Conclusión:

El modelo de 2 capas es el más equilibrado, ofreciendo un buen rendimiento tanto en entrenamiento como en validación, sin la complejidad innecesaria del modelo de 5 capas.
El modelo de 1 capa es una opción sólida y eficiente si se busca simplicidad y menor costo computacional, aunque puede no capturar completamente la complejidad de los datos.
El modelo de 5 capas, a pesar de su alta capacidad de representación, no mejora significativamente el rendimiento y sufre de sobreajuste, lo que sugiere que agregar más capas no siempre conduce a una mejor generalización.

1.5 .

TanH: tangente hiperbólica

Layer (type)	Output Shape	Param #
normalization_1 (Normalization)	(None, 4)	9
dense_68 (Dense)	(None, 64)	320
dense_69 (Dense)	(None, 64)	4,160
dense_70 (Dense)	(None, 1)	65

Summary modelo con función de activación TanH.

Sigmoide

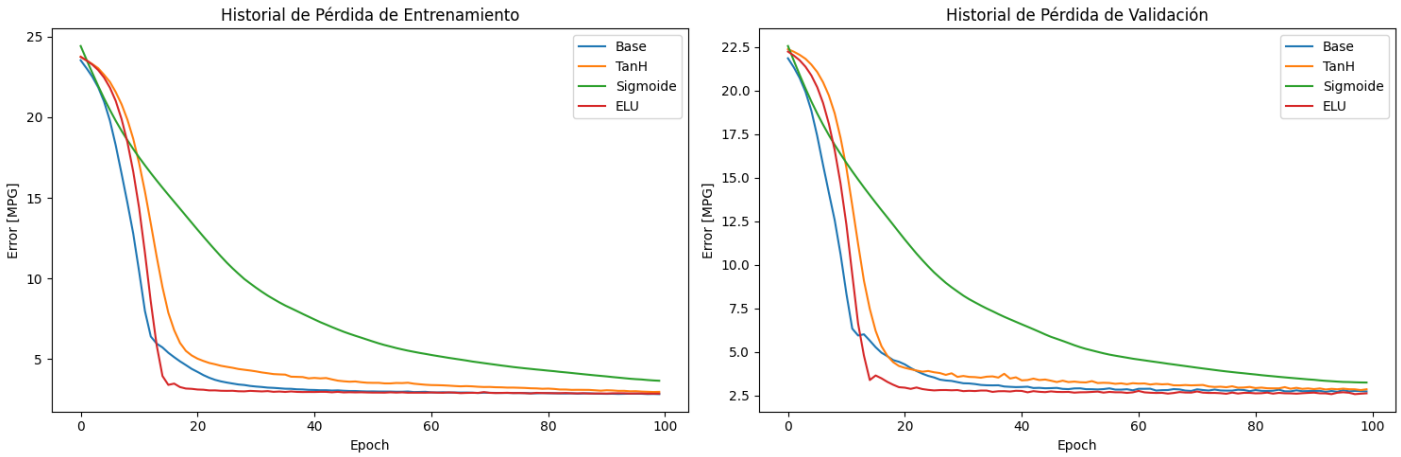
Layer (type)	Output Shape	Param #
normalization_1 (Normalization)	(None, 4)	9
dense_71 (Dense)	(None, 64)	320
dense_72 (Dense)	(None, 64)	4,160
dense_73 (Dense)	(None, 1)	65

Summary modelo con función de activación Sigmoide.

ELU: (Exponential Linear Unit)

Layer (type)	Output Shape	Param #
normalization_1 (Normalization)	(None, 4)	9
dense_80 (Dense)	(None, 64)	320
dense_81 (Dense)	(None, 64)	4,160
dense_82 (Dense)	(None, 1)	65

Summary modelo con función de activación ELU.



Graficas Perdida de entrenamiento y Perdida de validación respectivamente de modelos con diferente función de activación.

En el experimento, se entrenaron y compararon tres modelos modificando las funciones de activación: TanH, Sigmoide, y ELU, obteniendo lo siguientes resultados.

TanH: Tangente hiperbólica
La pérdida de entrenamiento decrece más rápidamente que la Sigmoide, pero más lentamente que ELU. La convergencia es razonable, aunque no tan eficiente como en el modelo con ELU. Además, el modelo muestra un rendimiento decente, pero con una curva de pérdida ligeramente más alta comparada con ELU.

Sigmoide
Entrenamiento**:** La curva de pérdida es la más lenta en descender. El modelo con Sigmoide tarda más en alcanzar un valor bajo de pérdida, lo que sugiere que la red tiene dificultades para aprender rápidamente. Además, el modelo muestra una alta pérdida en validación, lo que indica que su capacidad de generalización es la más débil entre los modelos probados.

ELU: Exponential Linear Unit
ELU muestra la mejor curva de entrenamiento, con una rápida disminución de la pérdida desde el principio, lo que indica una mayor capacidad para ajustar los datos de entrenamiento. El modelo con ELU generaliza mejor, con una pérdida de validación más baja que los otros modelos, demostrando una capacidad de aprendizaje más eficiente.

En conclusión, se puede decir qie el modelo con ELU es el más efectivo, proporcionando la mejor convergencia tanto en entrenamiento como en validación. TanH muestra un buen rendimiento, pero no supera a ELU. Sigmoide es la menos eficiente, mostrando dificultades para aprender debido al problema del desvanecimiento del gradiente.

1.6

Teniendo en cuenta los resultados anteriores se propusieron y evaluaron dos modelos diferentes al modelo base: uno con regularización L2 y otro con Dropout (20%).

La regularización L2 penaliza los pesos grandes, reduciendo el sobreajuste y mejorando la generalización. Aunque la pérdida de entrenamiento disminuye más lentamente, el modelo muestra una menor pérdida en validación.

Layer (type)	Output Shape	Param #
normalization (Normalization)	(None, 4)	9
dense_31 (Dense)	(None, 64)	320
dense_32 (Dense)	(None, 64)	4,160
dense_33 (Dense)	(None, 1)	65

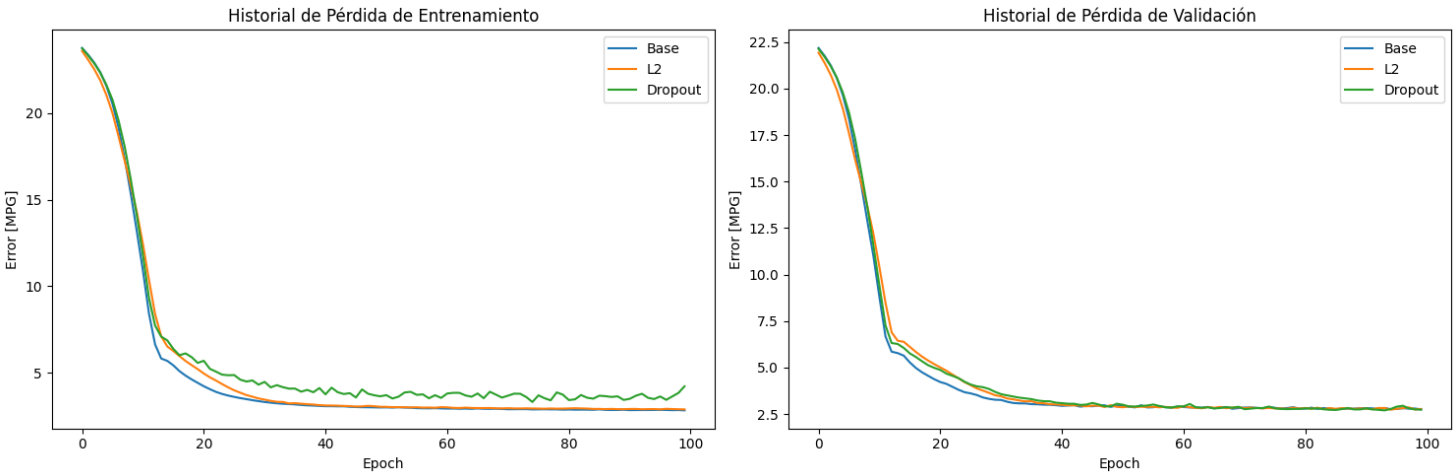
Summary del modelo regularización L2.

El modelo con Dropout desconecta aleatoriamente neuronas durante el entrenamiento, lo que evita la dependencia excesiva de ciertas neuronas y mejora la capacidad de generalización. En ambos modelos, la curva de pérdida de entrenamiento es más lenta, pero se observa una clara mejora en la pérdida de validación en comparación con el modelo base, que mostraba tendencia al sobreajuste.

Layer (type)	Output Shape	Param #
normalization (Normalization)	(None, 4)	9
dense_34 (Dense)	(None, 64)	320
dropout (Dropout)	(None, 64)	0
dense_35 (Dense)	(None, 64)	4,160
dropout_1 (Dropout)	(None, 64)	0
dense_36 (Dense)	(None, 1)	65

Summary del modelo Dropout.

En conclusión, tanto la regularización L2 como el Dropout mejoran la generalización, haciendo que los modelos sean más robustos y menos propensos a sobre ajustarse.



Graficas Perdida de entrenamiento y Perdida de validación respectivamente de modelos que mejores al base.

En la gráfica comparativa de pérdida de entrenamiento y validación, el modelo base muestra un rápido descenso de la pérdida, pero presenta un leve sobreajuste. El modelo con L2 generaliza mejor, con una pérdida de validación más baja, aunque su entrenamiento es más lento debido a la penalización de pesos grandes. El modelo con Dropout tiene una curva de entrenamiento más errática, pero alcanza la mejor pérdida en validación, indicando una mayor capacidad de generalización. En resumen, tanto L2 como Dropout mejoran el rendimiento, siendo Dropout la técnica más efectiva para evitar el sobreajuste.

Referencias:

Interactive Chaos. (s.f.). Regularización L2. InteractiveChaos. Recuperado de https://interactivechaos.com/es/manual/tutorial-de-machine-learning/regularizacion-l2

García Ferreira, F. (s.f.). Conoce todo sobre el Dropout. García Ferreira. Recuperado de https://www.garcia-ferreira.es/conoce-todo-sobre-el-dropout/