TRABAJO FINAL REDES NEURONALES

ENTREGA 2: ESTEBAN MONTOYA BETANCUR

RESPUESTA A LAS PREGUNTAS

2. Una vez analizado el código paso a paso y dada su completa comprensión, se procede a realizar modificaciones a cada uno de los hiperparámetros establecidos para el entrenamiento del modelo. Para esto, se tuvieron en cuenta las siguientes variables:

- **Cantidad de neuronas**
- **Tipo de optimizador**
- **Funciones de activación**
- **Funciones de pérdida**
- **Tamaños de entrada de la imagen**

En la Tabla 1 se muestran los parámetros con los cuales se realizó la variación en el modelo. El primer caso corresponde a la condición inicial, es decir, los parámetros con los cuales se presentó el problema inicialmente. En la parte derecha de la tabla se presentan los resultados obtenidos al variar dichos parámetros, en función de las métricas de función de pérdida, exactitud y precisión.

Adicionalmente, se calculó la diferencia de estas métricas con respecto a la condición inicial, con el fin de comparar si las condiciones variadas presentaban métricas iguales, superiores o inferiores, para finalmente considerar si tales variaciones proporcionaban mejores resultados en el entrenamiento del modelo. Específicamente, los colores rojo, verde y amarillo resaltan, respectivamente, si las métricas reflejan condiciones desfavorables, favorables o iguales a la condición inicial, y las barras ilustran la proporción en que esto sucede. En el punto 4 se analizarán los resultados obtenidos.

3.a Los labels se pasan a valores categóricos con el fin de poder ingresar a la capa de entrada de la red neuronal un vector de 0 y 1 en vez de un número entero, con el fin de proporcionar el input adecuado que necesita la red neuronal.

3.b La normalización a la hora de entrenar los datos ayuda en la manipulación de los valores, ya que se pasa de una matriz de dos dimensiones a un vector y los valores de este oscilarán entre 0 y 1.

Tabla 1. Tabla de parámetros aplicados al modelo y sus resultados en términos de función de pérdida, exactitud y precisión.

	Cantidad Neuronas	Tipo Optimizador	Función de Activación	Funciones de pérdida	Tamaños de entrada	Función de pérdida	Exactitud	Precisión	Diferencia Pérdida	diferencia accuracy	diferencia precision
1	512	rmsprop	relu	categorical_crossentropy	28*28	0,0683	98,38%	98,47%	0,0000	0,00%	0,00%
2	512	Adam	relu	categorical_crossentropy	28*28	0,0909	97,97%	98,07%	0,0226	-0,41%	-0,40%
3	512	adadelta	relu	categorical_crossentropy	28*28	0,7472	85,40%	98,01%	0,6789	-12,98%	-0,46%
4	512	Nadam	relu	categorical_crossentropy	28*28	0,0667	98,36%	98,45%	-0,0016	-0,02%	-0,02%
5	512	SGD	relu	categorical_crossentropy	28*28	0,2056	94,16%	95,80%	0,1373	-4,22%	-2,67%
6	1	rmsprop	relu	categorical_crossentropy	28*28	1,5672	41,29%	68,96%	1,4989	-57,09%	-29,51%
7	10	rmsprop	relu	categorical_crossentropy	28*28	0,2279	93,49%	94,62%	0,1596	-4,89%	-3,85%
8	100	rmsprop	relu	categorical_crossentropy	28*28	0,0842	97,78%	97,89%	0,0159	-0,60%	-0,58%
9	1000	rmsprop	relu	categorical_crossentropy	28*28	0,0625	98,47%	98,51%	-0,0058	0,09%	0,04%
10	5000	rmsprop	relu	categorical_crossentropy	28*28	0,0600	98,50%	98,55%	-0,0083	0,12%	0,08%
11	5000	rmsprop	sigmoid	categorical_crossentropy	28*28	0,0731	97,78%	97,97%	0,0048	-0,60%	-0,50%
12	512	rmsprop	softmax	categorical_crossentropy	28*28	1,3115	43,95%	93,13%	1,2432	-54,43%	-5,34%
13	512	rmsprop	softplus	categorical_crossentropy	28*28	0,0721	98,10%	98,20%	0,0038	-0,28%	-0,27%
14	512	rmsprop	softsign	categorical_crossentropy	28*28	0,0573	98,40%	98,47%	-0,0110	0,02%	0,00%
15	512	rmsprop	tanh	categorical_crossentropy	28*28	0,0658	98,21%	98,30%	-0,0025	-0,17%	-0,17%
16	512	rmsprop	selu	categorical_crossentropy	28*28	0,0773	97,89%	98,00%	0,0090	-0,49%	-0,47%
17	512	rmsprop	elu	categorical_crossentropy	28*28	0,0725	98,25%	98,30%	0,0042	-0,13%	-0,17%
18	512	rmsprop	exponential	categorical_crossentropy	28*28	4,3411	97,06%	97,06%	4,2728	-1,32%	-1,41%
19	512	rmsprop	relu	BinaryCrossentropy	28*28	0,0124	98,34%	98,39%	-0,0559	-0,04%	-0,08%
20	512	rmsprop	relu	Poisson	28*28	0,1061	98,29%	98,43%	0,0378	-0,09%	-0,04%
21	512	rmsprop	relu	KLDivergence	28*28	0,0721	98,27%	98,32%	0,0038	-0,11%	-0,15%
22	512	rmsprop	relu	MeanSquaredError	28*28	0,0031	97,96%	98,35%	-0,0652	-0,42%	-0,12%
23	512	rmsprop	relu	categorical_crossentropy	16*16	0,0571	98,42%	98,53%	-0,0112	0,04%	0,06%
24	512	rmsprop	relu	categorical_crossentropy	10*10	0,0915	97,24%	97,40%	0,0232	-1,14%	-1,07%
25	512	rmsprop	relu	categorical_crossentropy	32*32	0,0676	98,42%	98,48%	-0,0007	0,04%	0,01%
26	512	rmsprop	relu	categorical_crossentropy	64*64	0,0738	98,49%	98,52%	0,0055	0,11%	0,05%

4. A continuación, se presentará un análisis basado en las variaciones realizadas para el numeral 2 y cómo éstos se reflejan en las métricas implementadas para la evaluación del modelo.

4.1. Neuronas

Inicialmente, se analizan los resultados obtenidos al variar la cantidad de neuronas en el modelo. Para esto, se debe especificar que las neuronas presentadas en el modelo inicial corresponden a 512, lo cual permite resultados muy aceptables.

Para este caso, se analizaron varios escenarios, explorando inicialmente con 1 neurona, siendo este un caso extremo y que intuitivamente no presentaría buenos resultados, pero se quiso observar este escenario. Posteriormente, se analizaron los resultados con 10, 100, 1000 y 5000 neuronas.

Ante estas variaciones, como lo decía la intuición, el modelo que implementó 1 y 10 neuronas fue bastante pobre y las métricas obtuvieron porcentajes muy bajos de exactitud y precisión, a su vez que la función de pérdida aumentó. En el caso de 100 neuronas, los resultados fueron más cercanos al 100%, pero aún así, menos significativos que al implementar 512 neuronas. Mientras que el uso de 5000 neuronas consecuentemente reflejó unas mejores métricas, la mejora no se consideró lo suficientemente sustancial como para compensar el importante aumento de tiempo de ejecución que implicó.

En conclusión, la cantidad de neuronas empleadas en la condición inicial, representan unas métricas bastante buenas, a su vez que su tiempo de ejecución se mantiene en justas proporciones. En la Figura 1 se presentan las gráficas de función de pérdida para las diferentes variaciones de neuronas.

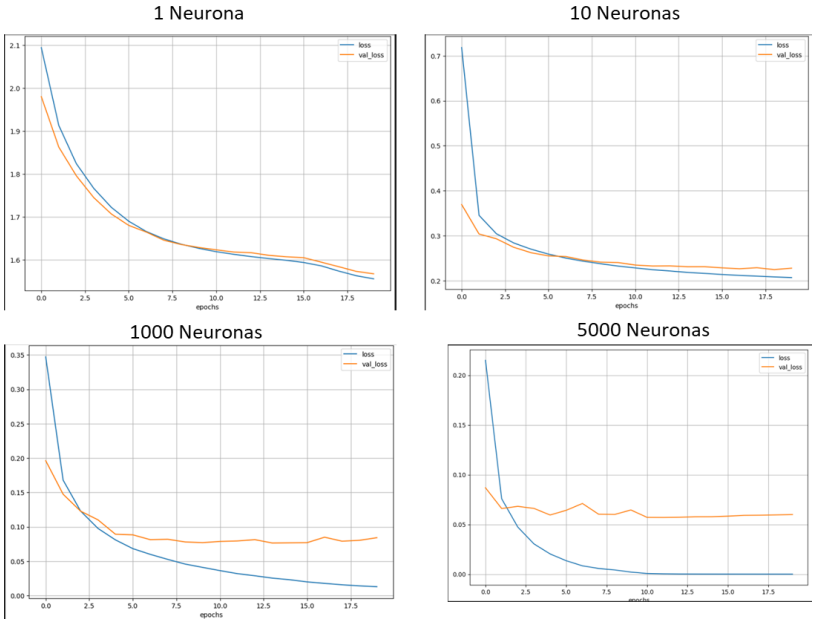

Figura 1. Gráficas de función de pérdida entrenamiento vs prueba para diferente cantidad de neuronas.

4.2. Función de Activación

Otro parámetro que se varió fue la función de activación para la capa oculta, donde se probaron diferentes funciones correspondientes a **softmax, sigmoide, softplus, softsign, tanh, selu, elu y exponencial**. Para estas variaciones se destaca el hecho de que solamente la función softsign presentó mejores métricas que la condición inicial, sin embargo, la diferencia es mínima y se podría considerar bastante similar. Las demás funciones obtuvieron rendimientos más bajos que la condición inicial, pero a nivel general, bastante cercanos a la función relu, lo que podría indicar que todas estas funciones pueden llegar a ser útiles y su comportamiento es similar a la función relu, a excepción de las funciones **softmax** y **exponencial**, las cuales muestran un rendimiento muchísimo más bajo y, en el caso de la exponencial, su gráfica es opuesta a las demás, incrementando su función de pérdida a medida que avanza el entrenamiento. Por lo cual, se entiende que ambas funciones tienen comportamientos diferentes al esperado para este tipo de ejercicio.

En la Figura 2 se presentan las gráficas de función de pérdida para las diferentes funciones de activación experimentadas.

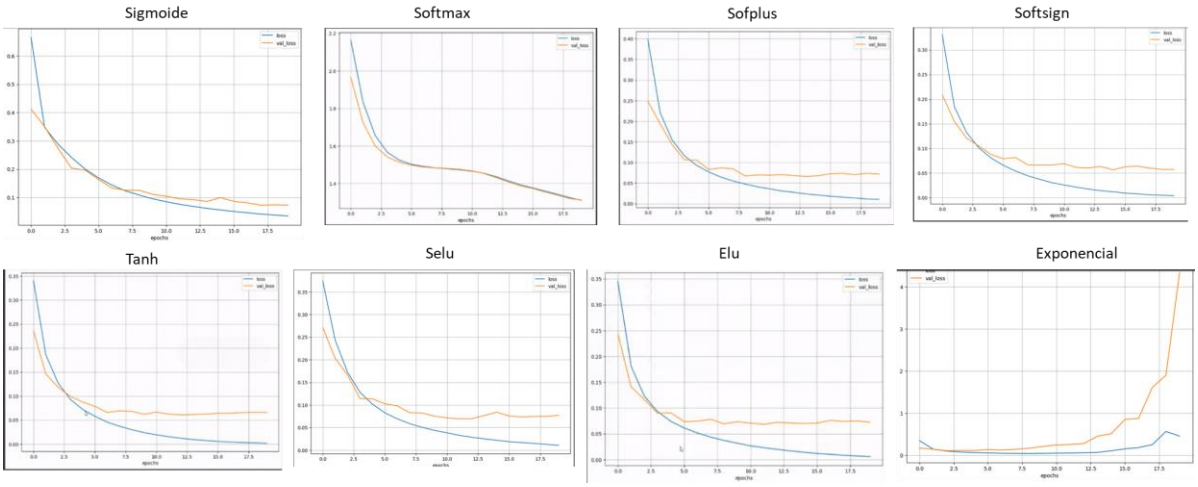

Figura 2. Gráficas de función de pérdida entrenamiento vs prueba para diferentes funciones de activación

4.3. Optimizadores

Para los optimizadores se exploraron parámetros asociados a los más relevantes, donde según algunas referencias solían ser los que mejor funcionan el 99% de las ocasiones. Estos parámetros corresponden a los optimizadores **Adam, Adadelta, Nadam** y **SGD.** De acuerdo con los entrenamientos evaluados, en todos los casos las métricas arrojaron resultados inferiores a la condición inicial, sin embargo, los parámetros Adam y Nadam fueron los más cercanos, los otros se alejaron un poco más considerablemente de los resultados esperados.

En la Figura 3 se presentan las gráficas de función de pérdida asociadas a los diferentes optimizadores empleados.

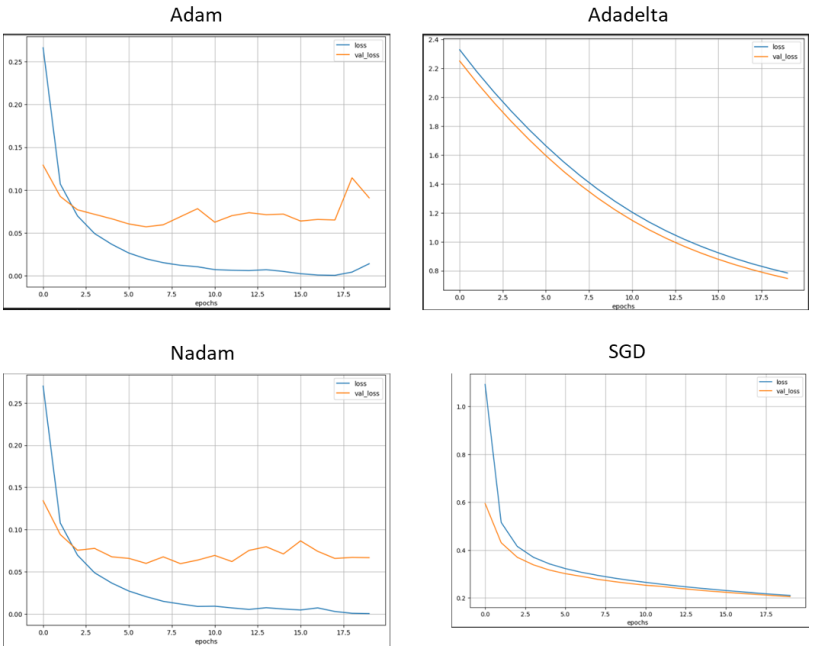

Figura 3. Gráficas de función de pérdida entrenamiento vs prueba para diferentes optimizadores

4.2. Tamaño de Entrada de Imagen

Para evaluar diferentes tamaños de entrada de las imágenes, se utilizó la librería CV2 para su redimensionamiento y obtener imágenes para la capa de entrada con menos o más pixeles. En este caso, se realizaron diferentes entrenamientos utilizando dimensiones de 16*16, 10*10, 32*32 y 64*64. De todas estas dimensiones, generalmente se obtuvieron resultados bastante similares, siendo todos ligeramente mejores que la condición inicial, a excepción de la resolución de 10*10, que fue un poco menos eficiente. A pesar de no ser muy considerable la diferencia de los resultados, podría ser una condición favorable reducir la dimensión de las imágenes a 16*16, ya que los datos de entrada serán menores y por lo tanto su procesamiento podría ser más ágil. En el caso de las resoluciones superiores, quizás no tendría mucho sentido implementarlas, debido a que el redimensionamiento está añadiendo nuevos pixeles basados en una interpolación de los existentes y no directamente información más detallada, por lo cual, además de añadir más datos a procesar, éstos no son más que obtenidos a partir de cálculos y no de observaciones.

De cualquier manera, la intuición sugiere que una resolución intermedia, la cual es de 28*28, proporciona una cantidad adecuada de información para la capa de entrada, y además no requiere realizar un redimensionamiento, que necesita un procesamiento adicional y que proporciona resultados muy similares.

Finalmente, como conclusión, se sugiere que la configuración inicial para el entrenamiento del modelo es bastante acertada en cuanto a los resultados obtenidos con cualquiera de las variaciones implementadas. Si bien, existen algunos parámetros que cumplen con condiciones muy similares o incluso mejores que la condición inicial, éstas en ocasiones implican mayor tiempo de procesamiento o pasos adicionales para la conversión de los insumos de entrada. De cualquier manera, se requeriría

un conocimiento más profundo de cada una de las variables ingresadas para poder determinar si vale la pena cambiar estos parámetros para refinar un poco el modelo, y quizás al analizar una cantidad más voluminosa de datos, podría reflejarse más considerablemente en los resultados.