Universidad de O'Higgins

Tarea N°2:
Redes Neuronales

Cristopher Acevedo Guajardo

COM4402 – Introducción a Inteligencia Artificial

Escuela de Ingeniería, Universidad de O'Higgins

28, Octubre, 2023

***Resumen*— En este informe sobre la tarea número 2 del curso se presenta la aplicación de distintos modelos de redes neuronales en el conjunto de datos "*Optical Recognition of Handwritten Digits Data Set*", para analizar y comparar el rendimiento en un problema de clasificación de dígitos de distintas estructuras o modelos de redes neuronales. Se utilizan las métricas de *Validation Loss* y *Accuracy* para controlar el ajuste de los modelos y que estos no se sobreajusten y se construyen las matrices de confusión para cada modelo, los cuales se utilizan para evaluar el rendimiento de los distintos modelos y funciones de activación en los conjuntos de entrenamiento, validación y de prueba. Las pruebas realizadas indican que el modelo más eficiente de los que fueron utilizados para este problema de clasificación es una red neuronal con una capa oculta de 40 unidades, con función de activación ReLU (Rectified Linear Unit).**

I. Introducción

Las redes neuronales son un tipo de algoritmo de Machine Learning que están inspiradas en la estructura y en la función del cerebro humano, capaces de aprender y resolver una amplia cantidad de tareas y problemas.

El objetivo principal de esta tarea es evaluar el rendimiento de distintos modelos de redes neuronales, con distinta cantidad de capas ocultas, distinta cantidad de unidades ocultas, y diferentes funciones de activación, en este caso, las funciones de activación ReLU (Rectified Lineal Unit) y Tanh (Tangente Hiperbólica).

Se utilizará la base de datos "*Optical Recognition of Handwritten Digits Data Set*", otorgada por el equipo docente del curso, que contiene 64 características, 10 clases y 5620 muestras.

Se evaluará el rendimiento de cada modelo de red neuronal en el conjunto de entrenamiento, en el conjunto de validación y en el conjunto de prueba.

Las métricas que usaremos para evaluar y comparar el rendimiento de los modelos serán el *validation loss*, *accuracy* y el *tiempo de ejecución*.

II. Marco teórico

Las redes neuronales son un tipo de algoritmo ampliamente utilizado en machine learning y en deep learning. Están formadas por nodos interconectados, llamados *neuronas*, que procesan información y se comunican entre sí. Las redes neuronales pueden entrenarse para realizar una amplia variedad de tareas, como el reconocimiento de imágenes, problemas de clasificación, el procesamiento del lenguaje natural, traducción automática, entre muchas otras más.

Los componentes de una red neuronal son los siguientes:

A. Capa de entrada (Input layer)

La capa de entrada recibe los datos de entrada (inputs) a la red. Por ejemplo, si la red se utiliza para el reconocimiento de imágenes, la capa de entrada podría recibir una representación en pixeles de una imagen.

B. Capas ocultas (Hidden layers)

Las capas ocultas se encargan de aprender patrones complejos en los datos. Al aumentar el número de capas ocultas de la red, será capaz de aprender patrones cada vez más complejos, pero aumentando también el costo computacional.

C. Capa de salida (Output layer)

La capa de salida produce los resultados de salida de la red (outputs). Por ejemplo, si la red se utiliza para el reconocimiento de imágenes, la capa de salida podría producir una distribución de probabilidades para las distintas clases de objetos en una imagen.

Las redes neuronales funcionan mediante la propagación de la información de la capa de entrada a la capa de salida y el entrenamiento de la red según el error en la salida entregada en comparación a la salida real.

Al entrenar una red neuronal, en cada capa las neuronas realizan un cálculo simple a partir de las entradas que reciben de la capa anterior. En los modelos más comunes, este cálculo consiste en la multiplicación de los valores de las entradas por un peso, y, para la mayoría de los casos, sumando un término correspondiente al sesgo de la red. Al recibir y ponderar estas entradas, se utiliza una función de activación para evaluar el resultado de esa neurona para las entradas que recibe, y se propaga por la red. Los valores de la capa de salida (el output, esto puede ser, una predicción) de la red se comparan con los valores reales de los datos de entrada, se calcula el error, y se ajustan los pesos de la red mediante el método llamado "*backpropagation*" para reducir el error. Este proceso se repite hasta que se la red es capaz de predecir con precisión los resultados del conjunto de entrenamiento. Cada vez que se repite este proceso, es decir, que se entrena la red, se calcula el error y se reajustan los pesos, se llama "*época*" (*epoch*). La función de pérdida que se utilizará es *Cross-Entropy Loss*, y se utilizará el optimizador *Adam*.

Las funciones de activación que se utilizarán son: ReLU (Rectified Lineal Unit), que es comúnmente utilizada en capas ocultas para introducir no-linealidad en el modelo, producir salidas positivas (y 0 para valores negativos) y permitir que la red aprenda representaciones más complejas de los datos, y la función Tanh (Tangente Hiperbólica), que también introduce no-linealidad a la red, pero la diferencia con ReLU es que su salida está en una escala diferente (-1 a 1).

Se evaluará el rendimiento de seis modelos distintos, utilizando distinto número de capas ocultas y con una distinta cantidad de unidades ocultas, así como también la diferencia de rendimiento al utilizar cada una de las funciones de activaciones, ReLU y Tanh.

Las métricas que se utilizarán para evaluar los modelos serán:

- *accuracy:* Representa la cantidad de predicciones correctas realizadas por el modelo, dividido en el número total de muestras (o labels).
- *validation loss*: Es el valor de la función de pérdida (loss) en el conjunto de validación. Un valor más bajo significa un mejor rendimiento del modelo. También se usará para evitar el sobreajuste (*overfitting*) en el modelo.
- *time*: El tiempo de ejecución del modelo, utilizando como tipo de Runtime el Acelerador de Hardware T4 GPU de Google Colab.

En conjunto, se construirá la matriz de confusión para cada modelo, con el fin de evaluar y analizar visualmente los resultados de las predicciones de los modelos.

III. Metodología

Esta sección describe en detalle el proceso y la metodología que se aplicó para evaluar el rendimiento de los modelos de red neuronal en la base de datos "*Optical Recognition of Handwritten Digits Data Set*".

A. Recopilación y carga de los datos

La base de datos utilizada en esta tarea es el "*Optical Recognition of Handwritten Digits Data Set*", extraída del repositorio entregado por el equipo docente del curso.

TABLA I
CARGA DE LA BASE DE DATOS AL PROGRAMA

```
!wget =
https://raw.githubusercontent.com/Felipe1401/Mine
ria/main/dataset_digits/1_digits_train.txt
!wget =
https://raw.githubusercontent.com/Felipe1401/Mine
ria/main/dataset_digits/1_digits_test.txt
```

B. Preprocesamiento de los datos

En el código base otorgado por el equipo docente para la realización de esta tarea se transformó cada conjunto de datos a un dataframe. Se separó el conjunto de entrenamiento original en dos conjuntos, un conjunto de entrenamiento nuevo con el 70% de los datos del conjunto de datos original, y un conjunto de validación con el otro 30% de los datos.

Luego, con el método de escalado de sklearn, se estandarizaron los tres conjuntos de datos (el de entrenamiento, validación y el de prueba) en función del conjunto de entrenamiento.

Utilizando la librería PyTorch, se crearon los Dataloaders con un *batch_size* de 128, para los tres conjuntos de datos (entrenamiento, validación y prueba).

C. Creación del modelo

En el código base otorgado por el equipo docente para la realización de esta tarea, utilizando la librería PyTorch, se estableció el modelo inicial de la red neuronal que se entrenó, utilizando la librería PyTorch. Se indicaron el número de capas de entrada, ocultas, de salida, la función de activación, y el criterio de pérdida (*Cross Entropy Loss*) junto con el optimizador *Adam*.

TABLA II
MODELO DE RED NEURONAL CON 1 CAPA OCULTA Y 10 UNIDADES OCULTAS, ACTIVACIÓN RELU

```
# ReLU, 1 capa oculta, 10 neuronas ocultas
model = nn.Sequential(
          nn.Linear(64,10),
          nn.ReLU(),
          nn.Linear(10,10)

device = torch.device("cuda")
model = model.to(device)
criterion = nn.CrossEntropyLoss
optimizer =
torch.optim.Adam(model.parameters(), lr = 1e.3)
```

El primer modelo que se evaluó fue una red neuronal con 64 neuronas en la capa de entrada correspondiente a las características del conjunto de datos, 10 neuronas en una capa oculta, y 10 neuronas de salida correspondiente a las diferentes clases dentro del conjunto de datos, en este caso, números del 0 al 9.

Se implementó también dentro del entrenamiento una forma de evitar el sobreajuste en la red. Esta implementación consistió en comparar, luego de un número determinado de épocas (25, en este caso), el valor de la función de pérdida en la época actual (en el conjunto de validación) con los valores de la función de pérdida hace 25 épocas. Si los 25 valores anteriores son menos que el actual, y además (para hacer más robusta la implementación) los últimos 10 valores que tuvo el *accuracy* son menores o iguales que el actual, significa que la red está comenzando a aumentar constantemente el valor de la función de pérdida, y empeorando el *accuracy*, por lo tanto, se detendrá el entrenamiento bajo esas condicionales.

TABLA III
CONDICIÓN DE DETENCIÓN DEL ENTRENAMIENTO PARA EVITAR EL SOBREAJUSTE DE LA RED

```
# ReLU, 1 capa oculta, 10 neuronas ocultas
if len(loss_val)>25:
  last_losses = loss_val[-25:]
  last_acc = accuracy_val[-10:]
  if all(loss < last_losses[-1] for loss in
last_losses[:-1]) and accuracy <=
min(last_acc):
    print("Entrenamiento finalizado debido a
que la pérdida de validación comenzó a
incrementar (evitando el sobreajuste).")
     ...
     break
```

Luego, duranta cada epoch (o época) en el entrenamiento, calculamos, guardamos y mostramos el número del epoch, el valor de la función de pérdida en el conjunto de entrenamiento en esa epoch, el valor de la función de pérdida en el conjunto de validación en esa epoch, el accuracy (utilizando el conjunto de validación) en la epoch, y, al final, el tiempo total de entrenamiento del modelo o de ejecución.

Esto se realizó para 6 modelos diferentes, todos con 64 neuronas/unidades en la capa de entrada, que recibe las características del dataset, y 10 neuronas/unidades en la capa de salida, que nos entrega la predicción de la clase de una muestra (dígitos del 0 al 9) y un máximo de 1000 epochs, pero variando la cantidad de capas ocultas, el número de neuronas en las capas ocultas, y la función de activación, utilizando los siguientes modelos:

(a) El mencionado anteriormente en la Tabla II, con 1 capa oculta y 10 neuronas ocultas, con función de activación ReLU
(b) 1 capa oculta y 40 neuronas en la capa oculta, con función de activación ReLU
(c) 1 capa oculta y 10 neuronas en la capa oculta y función de activación Tanh.
(d) 1 capa oculta y 40 neurona en la capa oculta, con función de activación Tanh
(e) 2 capas ocultas con 10 neuronas ocultas en cada capa, con función de activación ReLU
(f) 2 capas ocultas con 40 neuronas ocultas en cada capa, con función de activación ReLU

Finalmente, en base a los resultados de los desempeños de los modelos en las métricas evaluadas, se definió el mejor modelo para este conjunto de datos, y utilizando la función *model.state_dict()* se recuperó el estado de la red en donde el se obtuvo el mayor *accuracy*, y se utilizó para calcular el *accuracy* en el conjunto de prueba.

D. Evaluación y visualización de los resultados

Luego de crear cada modelo y entrenar la red con el conjunto de entrenamiento, las métricas de evaluación que se guardaron y que se mostraron durante el proceso de entrenamiento fueron:

- *validation loss*
- *accuracy*
- *time (segundos)*

para cada época en el entrenamiento. El accuracy se calculó según los resultados de las predicciones utilizando el conjunto de validación.

Una vez terminado el entrenamiento, se mostró el gráfico de la función de pérdida en el conjunto de entrenamiento y en el de validación durante cada época, y se graficaron las matrices de transición para ambos conjuntos, el de entrenamiento y de validación, en una escala de grises para visualizar el rendimiento del modelo en clasificar correctamente las muestras, y utilizando la red con mejor desempeño, que fue la red de una capa oculta y 40 neuronas ocultas con función de activación ReLU, se calculó su *accuracy* y se generó su matriz de confusión.

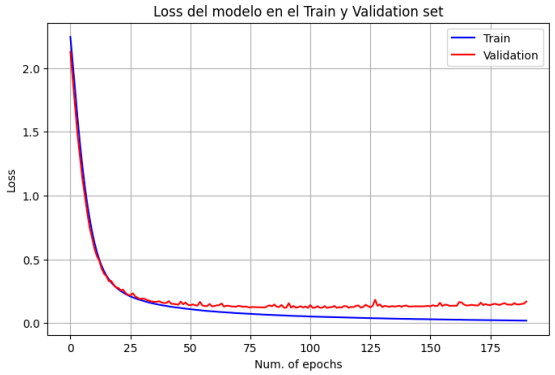
IV. Resultados

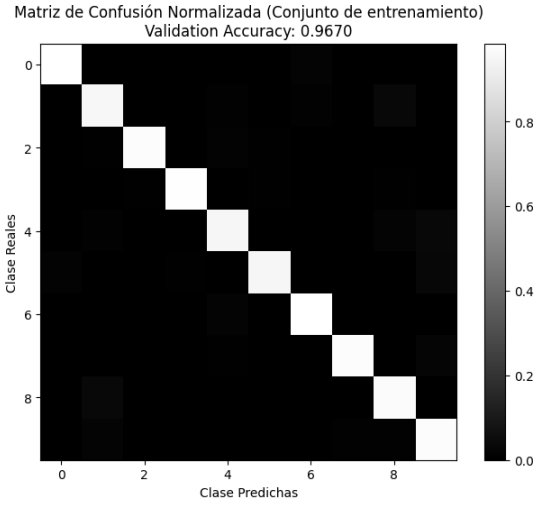
Los resultados obtenidos luego de haber realizado el entrenamiento y la evaluación de los seis modelos están resumidos en las siguientes tablas:

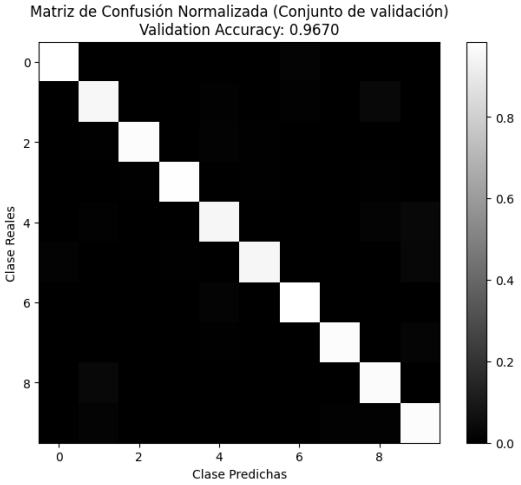
A. *1 capa oculta, 10 neuronas, función de activación ReLU*

TABLA IV
Modelo 1 capa oculta, 10 neuronas, f. activación ReLU

# of epochs	val loss	accuracy	time (s)
190	0,11929	0.9685	13.094



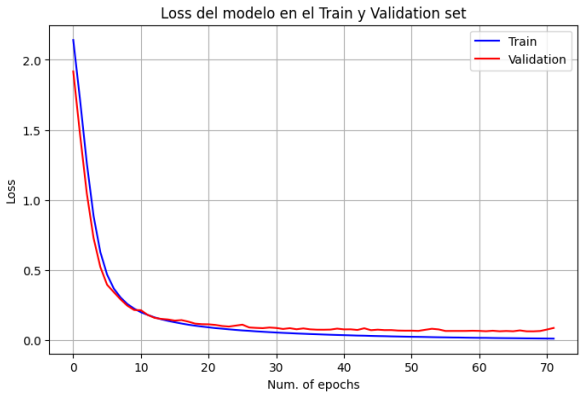


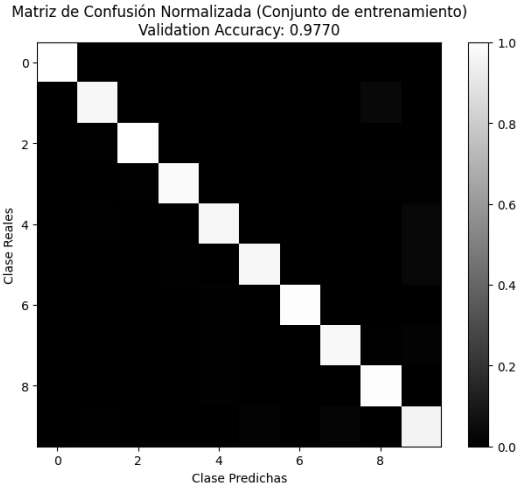


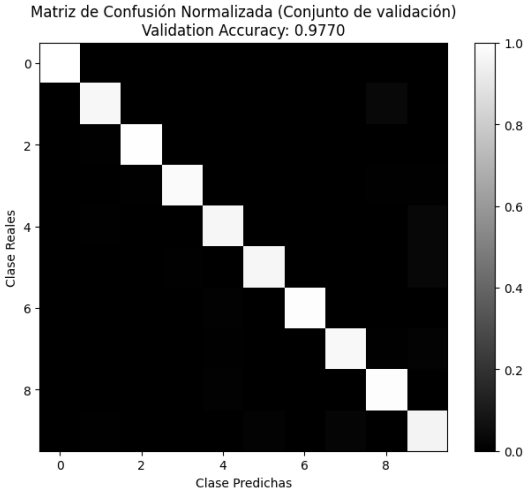
B. *1 capa oculta, 40 neuronas, función de activación ReLU*

TABLA V
MODELO 1 CAPA OCULTA, 40 NEURONAS, F. ACTIVACIÓN RELU

# of epochs	val loss	accuracy	time (s)
71	0.06432	0.9800	4.431



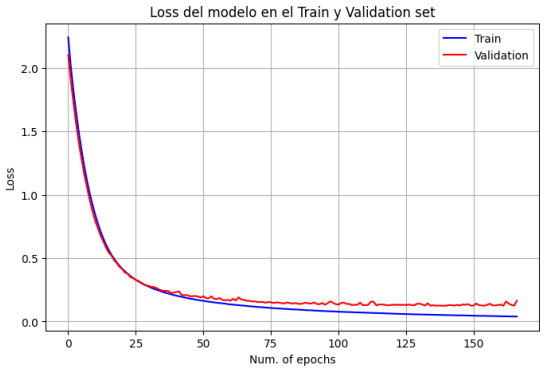


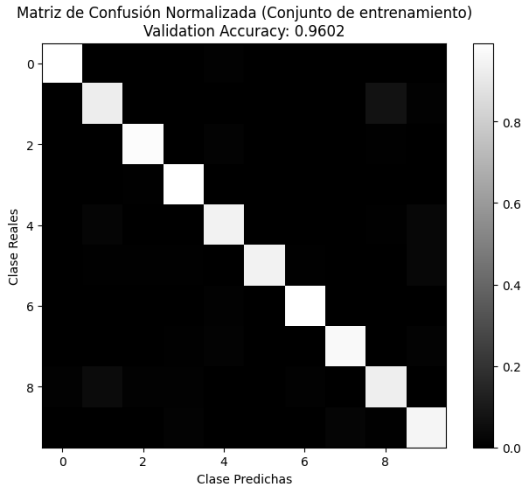


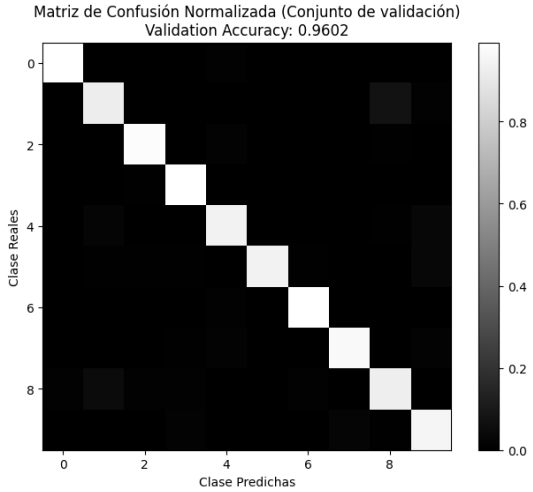
C. *1 capa oculta, 10 neuronas, función de activación Tanh*

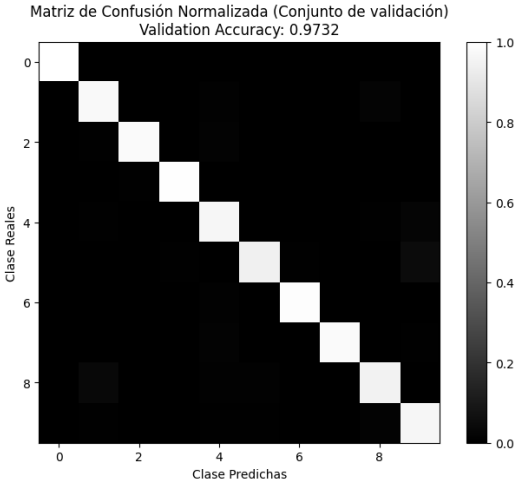
TABLA VI
MODELO 1 CAPA OCULTA, 10 NEURONAS, F. ACTIVACIÓN TANH

# of epochs	val loss	accuracy	time (s)
166	0.12428	0.9639	10.855





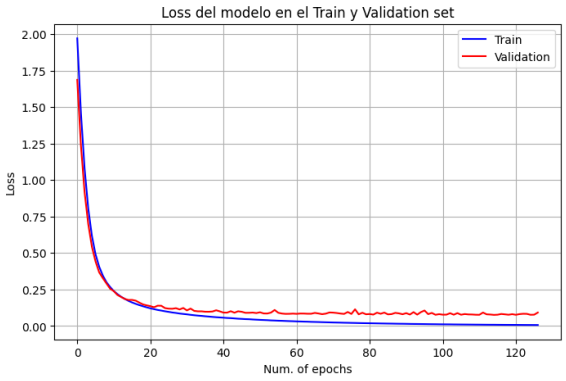


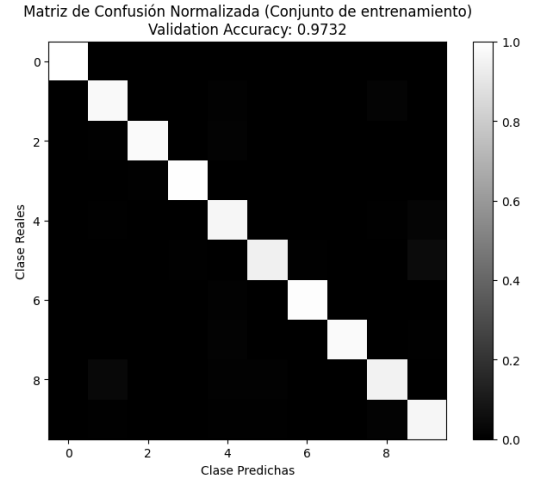


D. *1 capa oculta, 40 neuronas, función de activación Tanh*

TABLA VII
MODELO 1 CAPA OCULTA, 40 NEURONAS, F. ACTIVACIÓN TANH

# of epochs	val loss	accuracy	time (s)
126	0.0760	0.9739	8.023

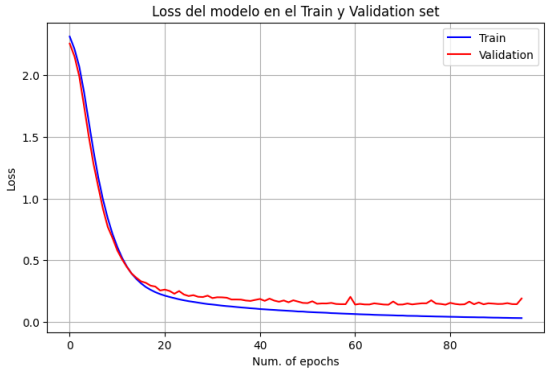


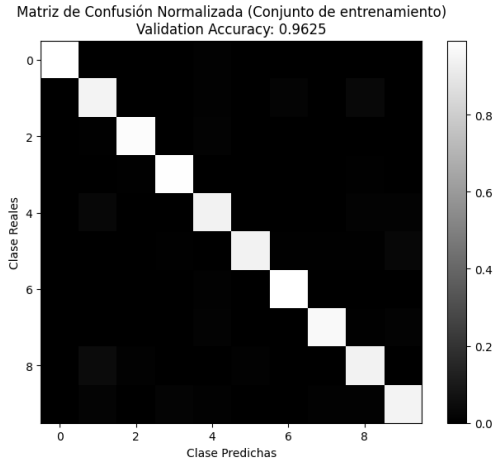


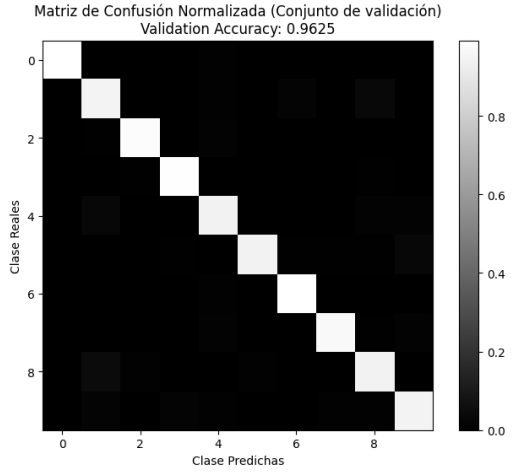
E. *2 capas ocultas, 10 neuronas en cada capa oculta, función de activación ReLU*

TABLA VIII
MODELO 1 CAPA OCULTA, 40 NEURONAS, F. ACTIVACIÓN RELU

# of epochs	val loss	accuracy	time (s)
95	0.1400	0.9678	7.046



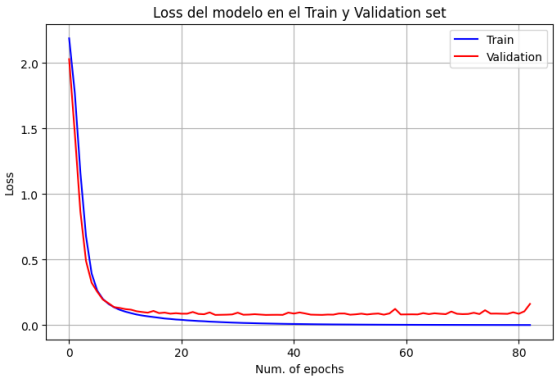


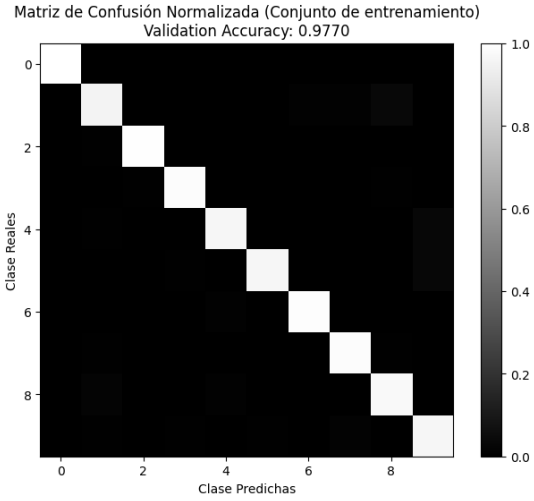


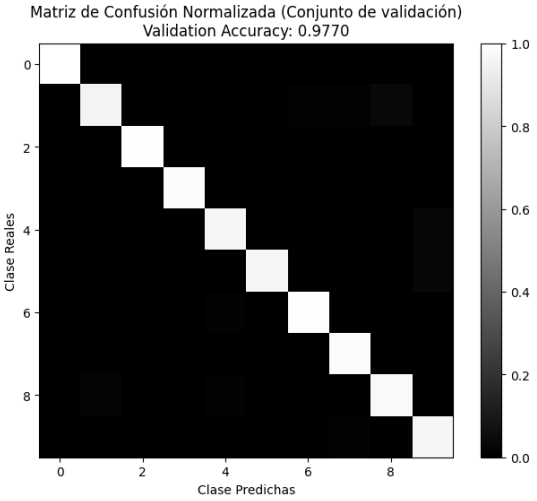
F. *2 capas ocultas, 40 neuronas en cada capa oculta, función de activación ReLU*

TABLA IX
MODELO 1 CAPA OCULTA, 40 NEURONAS, F. ACTIVACIÓN RELU

# of epochs	val loss	accuracy	time (s)
82	0.07839	0.9785	6.249



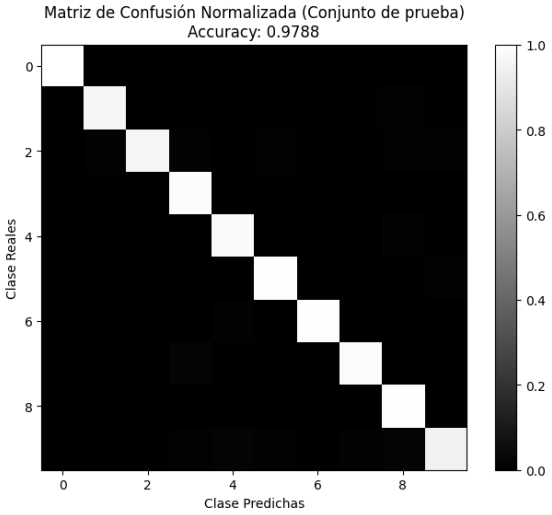




G. *Modelo con el estado (recuperado/checkpoint) de mayor accuracy durante el entrenamiento, con 1 capa oculta, 40 neuronas ocultas y función de activación ReLU*

TABLA X
MODELO 1 CAPA OCULTA, 40 NEURONAS, F. ACTIVACIÓN RELU, EN EL ESTADO CON MAYOR ACCURACY

accuracy en el conjunto de prueba
0.9788



V. ANÁLISIS

A. *Efectos de variar la cantidad de neuronas en la capa oculta*

El efecto de aumentar de 10 neuronas ocultas a 40 neuronas ocultas en la capa oculta, para el caso de tener una capa oculta y activación ReLU, redució el tiempo de entrenamiento de

aproximadamente 13 a 4.5 segundos, el validation loss de 0.1193 a 0.0643, y aumentó el accuracy de un 96.85% a un 98%, siendo este el mejor accuracy obtenido en los distintos modelos o arquitecturas probadas. Haciendo la misma variación en la cantidad de neuronas, pero utilizando la función de activación Tanh, se redujo el tiempo de 10.85 segundos a 8 segundos aproximadamente, el validation loss de 0.124 a 0.076, y el accuracy aumentó de un 96.4% a un 97.4%.

B. *Efectos de variar la cantidad de capas ocultas*

Al variar la cantidad de capas ocultas, utilizando la función de activación ReLU en los dos modelos, los cuales se evaluaron con 2 capas ocultas y con una variación de 10 neuronas ocultas en cada capa en un modelo, y 20 neuronas ocultas en cada capa en el otro modelo, se pudo notar, entre estos dos modelos, que al utilizar 2 capas ocultas con 20 neuronas ocultas en cada capa se disminuyó el tiempo de entrenamiento de 7 segundos a 6.25 segundos, el validation loss de 0.14 a 0.0784, y el accuracy aumentó de un 96.8% a un 97.9%.

Si hacemos la comparación con los efectos de variar la cantidad de neuronas al utilizar solamente una capa oculta, el efecto en el desempeño de la red mejora en ambos casos, al aumentar las neuronas y al aumentar las capas, sin embargo, al utilizar dos capas de 20 neuronas ocultas (con 40 neuronas en total) se obtuvo un accuracy de un 97.9%, y utilizando una capa oculta con 40 neuronas ocultas tuvo como resultado un accurac de un 98%, así como también una diferencia en el tiempo siendo casi 2 segundos y requiriendo menos épocas el modelo con una sola capa oculta y una mayor cantidad de neuronas en la capa.

C. *Efectos de la función de activación*

Al utilizar la función de activación ReLU, se obtuvieron tiempos de entrenamiento de aproximadamente 13 y 4.5 segundos para los modelos de 10 y 40 neuronas ocultas, respectivamente, y unos tiempos de entrenamiento de aproximadamente 10.8 y 8 segundos para los modelos de 10 y 40 neuronas ocultas, respectivamente, utilizando la función Tanh.

El accuracy también aumento al cambiar la función de activación. Al utilizar la función de activación ReLU, el accuracy para los modelos de 10 y 40 neuronas en la capa oculta fue de un 96.9% y 98%, respectivamente, a diferencia de la función de activación Tanh, obtuvo un accuracy de 96.4% y 97.4% respectivamente para cada modelo.

D. *Matriz de confusión y accuracy en el conjunto de validación*

Las matrices de confusiones que se obtuvieron para cada uno de los modelos o arquitecturas, utilizando el conjunto de validación, están en una escala de grises, y se pudo observar para todos los modelos que se tiene una diagonal de color blanco en la matriz. Esto indicó que, para todas las redes entrenadas, la clase predicha o la clasificación de una muestra de la red fue correctamente clasificada para casi todos (casi el 100%) de los datos/muestras. Todas las demás casillas (es decir, los demás pares de clases) están casi totalmente en negro, lo que significa que las muestras que fueron clasificadas incorrectamente por la red fueron muy pocas (o cerca del 0%).

Si se ajusta correctamente el contraste del monitor que se esté usando, es posible observar que se destaca un pequeño tono de gris (no negro completamente) para todos los modelos los casos caso donde: la clase predicha es un 1 y la clase real es un 8; la clase predicha es 9 y la clase real es 5; la clase predicha es 8 y la clase real es 1, siendo este último caso (predicha 8 y real 1) más notorio en la matriz de confusión del modelo de 1 capa ocultas con 10 neuronas ocultas y función de activación Tanh, que justamente es el modelo o estructura con

el accuracy más bajo dentro de los que fueron evaluados. En estos casos en particular, es probable que el 5 sea “dibujado” o escrito con forma de 9, y lo mismo para los demás casos.

Según el accuracy obtenido por cada modelo o estructura de red, aparte de los resultados mencionados en cuanto a la variación de esta métrica según las capas, la cantidad de neuronas y la función de activación, se pudo observar que en general el mejor accuracy se obtiene al utilizar un modelo de una capa oculta, con 40 neuronas ocultas y utilizando la función ReLU. Además, los valores del accuracy obtenidos entre las distintas variaciones de los modelos están entre un 96% y un 98%, por lo que, en general, se obtuvo un buen desempeño de los modelos utilizados, observando el accuracy y las matrices de confusión obtenidos.

E. *Matriz de confusión y accuracy en el conjunto de prueba*

El resultado del *accuracy* en el conjunto de prueba para el mejor modelo (de 1 capa oculta y 40 neuronas ocultas, con función de activación ReLU) fue de 0.9788, es decir, clasificó correctamente un 97.9% (aproximadamente) de las muestras del conjunto de pruebas. En el conjunto de validación, hasta antes que el modelo se empezara a sobreajustar, habíamos obtenido un accuracy de un 98% para este mismo modelo, por lo que la diferencia entre el accuracy de la red en el conjunto de validación y el conjunto de prueba es mínima, dejando en prueba otra vez que el modelo de 1 capa oculta y 40 neuronas ocultas es un buen modelo/estructura (y el mejor dentro de los que se probaron) para clasificar las muestras del conjunto de datos utilizado.

En cuanto a la matriz de confusión, se obtuvo una matriz con las mismas características que los demás modelos: una diagonal en blanco (para las clases predichas correctamente) y lo demás en negro, sin embargo, la presencia de los casos con cierto grado de ”gris” al ajustar el contraste del monitor son menos visibles para este modelo, y también lo son en el conjunto de prueba.

VI. Conclusiones generales

Se pudo observar que, para el problema/tarea de clasificación de dígitos desde el 0 al 9, con el conjunto de datos utilizado “*Optical Recognition of Handwritten Digits Data Set*”, utilizar un mayor número de unidades o neuronas ocultas en las capas ocultas mejoró notablemente el desempeño de la red, que se pudo observar en el *accuracy* y en la matriz de confusión de cada modelo. El modelo de una capa oculta y con 40 unidades ocultas y con función de activación ReLU fue el modelo que obtuvo el mejor desempeño en el conjunto de validación, con una disminución del tiempo de entrenamiento de aproximadamente un 70% en comparación que el modelo de 10 neuronas ocultas, así como también un aumento en accuracy de casi un 1.5%, y teniendo un resultado similar en el conjunto de prueba.

Variar la función de activación en el modelo también mostró una gran diferencia en el desempeño de la red, pero se obtuvo siempre mejores resultados al utilizar los modelos con función de activación ReLU.

Por otro lado, al utilizar dos capas ocultas en vez de una también hizo que aumentara considerablemente el desempeño de la red, sin embargo, aún utilizando 2 capas ocultas, el aumento de desempeño más significativo estuvo dado al aumentar el número de neuronas ocultas en las dos capas ocultas.

BIBLIOGRAFÍA Y REFERENCIAS

[1] Ignacio Bugeño, Alfonso Ehijo, Felipe Gómez, Camila Rapu, Código base Tarea 2, Escuela de Ingeniería, Universidad de O'Higgins. https://colab.research.google.com/drive/1btpLNbRF05LCiPt7cfUGKYOxZfbvihNI?usp=sharing#scrollTo=SVqQYb9wcb27

[2] Ignacio Bugeño, Alfonso Ehijo, Felipe Gómez, Camila Rapu, "Ayudantía 2 - Clustering", Escuela de Ingeniería, Universidad de O'Higgins. Reference: https://colab.research.google.com/drive/1LG3xC0BM6hPoOe-KLfncd0Rvf7__SgxA?usp=sharing#scrollTo=n00kastddkac

[3] PyTorch "torch.max", PyTorch torch documentation. Reference: https://pytorch.org/docs/stable/generated/torch.max.html