Tarea N° 2

Redes Neuronales

Javiera Ahumada Moreno
Escuela de Ingeniería, Universidad de O'Higgins
21, oct, 2023

***Abstract*—Este documento tiene como objetivo evaluar, analizar y visualizar redes neuronales en la problemática de clasificación de dígitos utilizando el conjunto de datos Optical Recognition of Handwritten Digits. Este conjunto consta de 64 características, 10 clases y un total de 5620 muestras. Se emplearán redes neuronales con una estructura que incluye una capa de entrada de dimensionalidad 64, una o dos capas ocultas y una capa de salida con 10 neuronas con función de activación softmax, se utiliza la entropía cruzada como función de perdida, aplicando optimizador Adam y utilizando PyTorch para entrenar y validar la red neuronal que implementa el clasificador de dígitos, además se implementarán funcionalidades específicas para evitar el sobreajuste de la red, incluyendo un control cuidadoso del proceso de entrenamiento y la monitorización de las pérdidas de entrenamiento y validación.// Con la finalidad de investigar los efectos de variar la estructura de la red para comprender cómo estos factores afectan su rendimiento.**

I. Introducción

Las redes neuronales han emergido como una gran herramienta en el análisis de datos y la clasificación de información. Las redes neuronales en programación se relacionan a un enfoque de modelado computacional inspirado en el funcionamiento del cerebro humano. Estas redes neuronales artificiales son un elemento esencial en el campo del aprendizaje automático y la inteligencia artificial. Tienen la capacidad de aprender y adaptarse a partir de los datos. son capaces de identificar patrones, realizar predicciones y mejorar su rendimiento a medida que se les proporciona más información. pueden manejar datos altamente complejos y no estructurados, lo que las hace adecuadas para tareas como el procesamiento de clasificación de dígitos.

A lo largo de este análisis, se examina cómo las redes neuronales se aplican en distintos escenarios, se investigan cómo diferentes configuraciones de redes neuronales, como el número de capas ocultas, el tamaño de las neuronas y las funciones de activación, pueden influir en su rendimiento, abordando un desafío crítico en el entrenamiento de redes neuronales: evitar el sobreajuste. Se analizarán estrategias para detener el entrenamiento en el momento adecuado, lo que asegura que las redes sean efectivas en la generalización de los datos no vistos, comprendiendo por qué las redes neuronales son buenas herramientas para resolver problemas complejos y cómo estas están transformando industrias desde la visión por computadora hasta la atención médica.

La tarea de clasificar dígitos ha servido como un escenario práctico, pues se ha observado cómo las redes neuronales se aplican a este desafío y cómo su rendimiento varía según la configuración.

II. Marco Teórico

Las redes neuronales, también conocidas como redes neuronales artificiales o redes neuronales simuladas, son un subconjunto de machine learning y forman la columna vertebral de los algoritmos de deep learning. Su nombre y estructura están inspirados en el cerebro humano e imitan la forma en que las neuronas biológicas se envían señales entre sí.

Consisten en capas de nodos que contienen una capa de entrada, una o más capas ocultas y una capa de salida. Cada nodo o neurona artificial está conectado a otro nodo y tiene un peso y un umbral asociado. Si la salida de un nodo excede un cierto umbral, el nodo se activa y envía datos a la siguiente capa de la red. De lo contrario, los datos no se transmitirán a la siguiente capa de red. Dependen de datos de entrenamiento para aprender y mejorar su precisión con el tiempo. Pero una vez que estos algoritmos de aprendizaje se afinan, se convierten en herramientas computacionales y de inteligencia artificial que pueden clasificar y agrupar datos a alta velocidad.

El propósito de este estudio es aplicar estas redes para la clasificación de dígitos del conjunto de datos Optical Recognition of Handwritten Digits. Las redes neuronales serán entrenadas y evaluadas en diversas configuraciones para analizar los efectos de realizar cambios en su esqueleto que influyen significativamente en la capacidad de la red para aprender y generalizar patrones. El uso de redes neuronales en problemas de clasificación se basa en la capacidad de estas redes para aprender y mejorar su precisión a medida que se entrenan con datos. La elección de la función de activación en una neurona determina cómo esta responde a las señales de entrada. En este estudio, se evaluarán las funciones de activación ReLU (se utiliza en las capas ocultas de la red. ReLU es conocida por su efecto de introducir no linealidad y ayudar a la red a aprender relaciones complejas en los datos.) y Tanh (La función Tanh es una función sigmoide centrada en cero. A medida que la entrada se vuelve más positiva, la salida se acerca a 1; cuando es negativa, la salida se acerca a -1. Esto introduce no linealidad similar a la función ReLU.) para comprender su impacto en el rendimiento de la red. Donde en cada configuración de red, se realizará un proceso de entrenamiento y validación. El objetivo es calcular el loss de entrenamiento y validación, así como el tiempo de entrenamiento total requerido. El entrenamiento se detendrá cuando el loss de validación comience a aumentar, evitando así el sobreajuste de la red. el desempeño de las redes neuronales se evaluará mediante generar matrices de confusión normalizadas y el cálculo de la precisión normalizada en los

conjuntos de entrenamiento y validación. Estas matrices y métricas de precisión se utilizarán para entender cómo las redes neuronales clasifican los dígitos.

III. Metodología

Este análisis se basa en una serie de pasos para investigar y evaluar el rendimiento de la red neuronal para el problema de clasificación de dígitos, para esto se inicia con la carga y preparación de los datos, donde se cargaron dos conjuntos de datos: "1 digits train.txt" y "1 digits test.txt".

Los datos se dividieron en conjuntos de entrenamiento, validación y prueba y se normalizaron para que tengan una media de 0 y una desviación estándar de 1.

Siguiendo con el diseño de estructura de la red neuronal, la cual consta de una capa de entrada con 64 nodos (correspondiente a las características de entrada), capas ocultas y una capa de salida con 10 neuronas, utilizando la función de activación softmax.

se inicia el tiempo de entrenamiento del modelo y se inicializa una variable $best_val_loss$ y se crea una variable para rastrear las pérdidas de entrenamiento y validación, así como el tiempo para el entrenamiento. después se cambia el modelo al modo de entrenamiento (model.train()) y se inicia una variable $train_loss$ para rastrear la pérdida de entrenamiento en esta época, iterando a través del conjunto de datos de entrenamiento $(\text{dataloader}_train)enlotes$.

para cada lote, se cargan las características y etiquetas, se establecen los gradientes en cero $(optimizer.zero_grad())$ para evitar acumulación de gradientes, se pasan las características a través del modelo y se calcula la pérdida con respecto a las etiquetas reales, continuando con realizar la retropropagación de gradientes $(loss.backward())$ y se actualizan los pesos del modelo mediante el optimizador $(optimizer.step())$ sumando la pérdida de este lote a la$train_loss$. Se crea una variable val_loss para rastrear la pérdida de validación y $val_accuracy$ para rastrear la precisión de esta y se itera a través del conjunto de datos de validación $(dataloader_val)$ en lotes para cada lote, se cargan las características y etiquetas y se calcula la pérdida y la precisión.

```
# Clculo de Prdida y Precisin de Validacin
model.eval()
val_loss = 0.0
val_accuracy = 0.0

with torch.no_grad():
    for data in dataloader_val:
        val_inputs = data["features"].to(device)
        val_labels = data["labels"].to(device)
        val_outputs = model(val_inputs)
        val_loss += criterion(val_outputs, val_labels).item()
        val_accuracy += torch.sum(torch.argmax(val_outputs, dim=1) == val_labels).
        item()

train_loss /= len(dataloader_train)
val_loss /= len(dataloader_val)
val_accuracy /= len(dataloader_val.dataset)
```

La precisión se calcula comparando las etiquetas predichas con las etiquetas reales y las pérdidas se promedian en función del número de lotes en el conjunto de validación. continuando con comprobar si la pérdida de validación actual es mejor que la mejor pérdida de validación registrada actual $(best_val_loss)$ si esto es así, se actualiza $best_val_loss$, y el estado del modelo se guarda en un archivo llamado $best_model.pt$ esto asegura que se guarda el mejor modelo según la pérdida de validación.

```
# Evaluar el modelo en el conjunto de prueba
model.load_state_dict(torch.load('best_model.pt'))
model.eval()
test_loss = 0.0
test_accuracy = 0.0
with torch.no_grad():
    for data in dataloader_test:
        test_inputs = data["features"].to(device)
        test_labels = data["labels"].to(device)
        test_outputs = model(test_inputs)
        test_loss += criterion(test_outputs, test_labels).item()
        test_accuracy += torch.sum(torch.argmax(test_outputs, dim=1) == test_labels).
        item()
test_loss /= len(dataloader_test)
test_accuracy /= len(dataloader_test.dataset)
```

Luego sigue con la evaluación de la red neuronal, en el conjunto de entrenamiento y validación, se calcula la matriz de confusión normalizada y el accuracy normalizado. se establece el modo de evaluación (model.eval()) para que no se realicen cambios en los parámetros de la red durante la evaluación y se hacen las predicciones en el conjunto de entrenamiento y validación. Las predicciones se almacenan en la lista y_train_pred y en la lista predictions, y las etiquetas verdaderas en el conjunto de entrenamiento se almacenan en y_train_true.

```
# Obtener las predicciones del conjunto de entrenamiento
import sklearn.metrics as metrics

model.eval()
y_train_pred = []
y_train_true = []

with torch.no_grad():
    for data in dataloader_train:
        inputs = data["features"].to(device)
        labels = data["labels"].to(device)
        outputs = model(inputs)
        _, predicted = torch.max(outputs.data, 1)
        y_train_pred.extend(predicted.tolist())
        y_train_true.extend(labels.tolist())

# Calcular la matriz de confusin y el accuracy
m = metrics.confusion_matrix(y_train_true, y_train_pred, normalize='true')

# Calcular el accuracy normalizado
normalized_accuracy = metrics.accuracy_score(y_train_true, y_train_pred)

# Graficar la matriz de confusin con colores segn el accuracy
plt.figure(figsize=(10, 8))
sns.heatmap(m, cmap="Greys", annot=True, fmt=".2f", linewidths=.5)
plt.title("Normalized Confusion Matrix\nAccuracy: {:.2%}".format(
        normalized_accuracy))
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

# Mostrar el accuracy normalizado
print("Accuracy Normalizado: {:.3f}".format(normalized_accuracy))
```

Este mismo proceso de entrenamiento de la red neuronal se repite para los múltiples casos: a, b, c, d, e, f. La red se entrena durante un número máximo de 1000 épocas en cada época y se va cambiando el modelo de la red neuronal:

IV. Resulatdos

A. Caso a:

- Test Loss: 0.125
- Test Accuracy: 0.960
- validaciones Accuracy Normalizado: 0.959
- entrenamientos Accuracy Normalizado: 0.986

se entrenó una red neuronal con una estructura que consta de una capa oculta que contiene 10 neuronas y se usó la función

de activación ReLU en la capa oculta. el test loss indica que el modelo es capaz de minimizar el error en el conjunto de prueba. él teste accuracy lo que significa que el modelo es capaz de clasificar correctamente el 96 porciento de los ejemplos en el conjunto de prueba. Sin embargo, la precisión en el conjunto de entrenamiento sueles ser más alta que en el conjunto de validación, lo que puede indicar un pequeño grado de sobreajuste. Además, está la representación gráfica de los valores de loss, que permite observar cómo evoluciona la pérdida durante el entrenamiento en comparación con el tiempo transcurrido

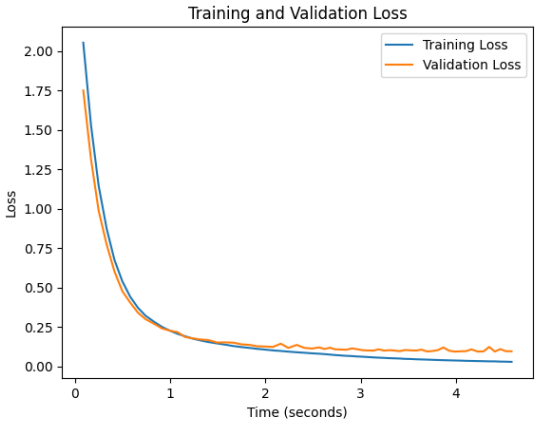

Fig. 1: Gráfico caso a

Luego, se procede a generar la matriz de confusión normalizada y el accuracy normalizado tanto para el conjunto de entrenamiento como para el conjunto de validación. El modelo muestra un rendimiento bastante consistente en los conjuntos de validación y prueba en comparación con el conjunto de entrenamiento. Esto sugiere que el modelo generaliza de manera efectiva y no muestra sobreajuste.

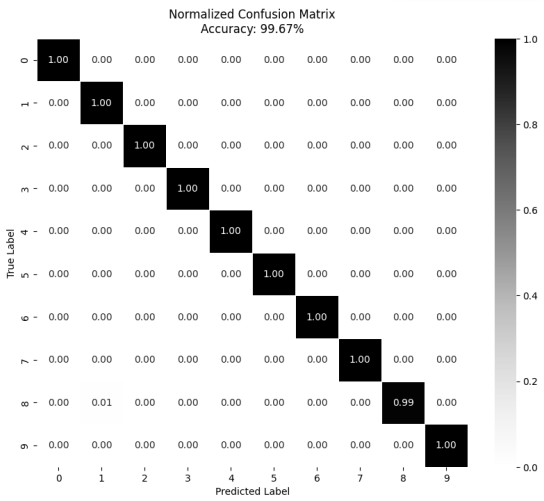

Fig. 2: Matriz, confusión y accuracy normalizado, entrenamiento

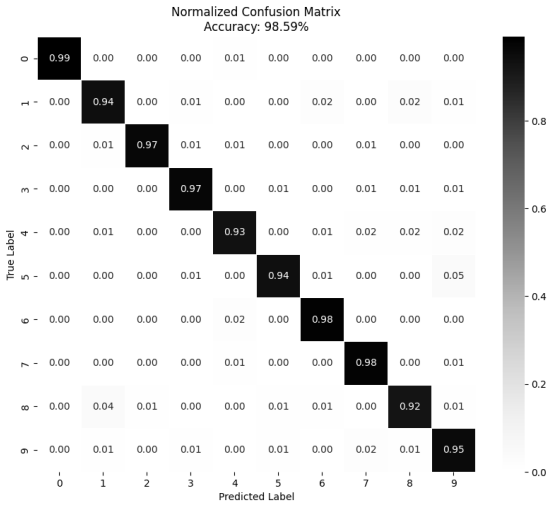

Fig. 3: Matriz, confusión y accuracy normalizado, validación

B. Caso b:

- Test Loss: 0.069
- Test Accuracy: 0.980
- validaciones Accuracy Normalizado: 0.976
- entrenamientos Accuracy Normalizado: 1.000

  El modelo para 40 neuronas en la capa oculta y función de activación ReLU, y 1000 épocas muestra un alto rendimiento tanto en el conjunto de validación como en el conjunto de prueba. El modelo muestra su robustez y capacidad para aplicar lo que ha aprendido. En este caso, la precisión en el conjunto de validación también es alta, lo que sugiere que el modelo es capaz de generalizar de manera efectiva. la precisión de entrenamiento del 100 porciento indica que el modelo se ha sobreajustado a los datos de entrenamiento para evaluar la capacidad del modelo, además la alta precisión en el conjunto de prueba refuerza la idea de que el modelo no solo ha memorizado los datos de entrenamiento, sino que también ha aprendido a realizar buenas predicciones

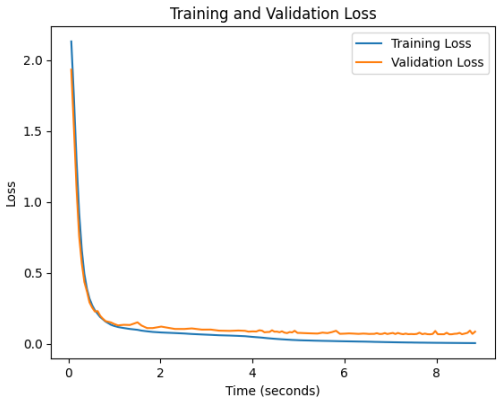

Fig. 4: Gráfico caso b

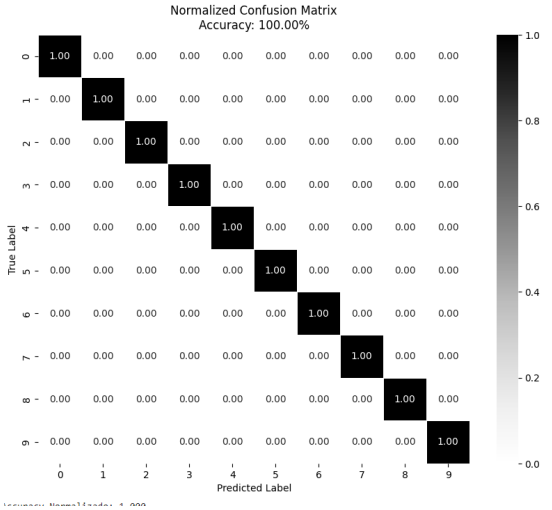

Fig. 5: Matriz, confusión y accuracy normalizado, entrenamiento

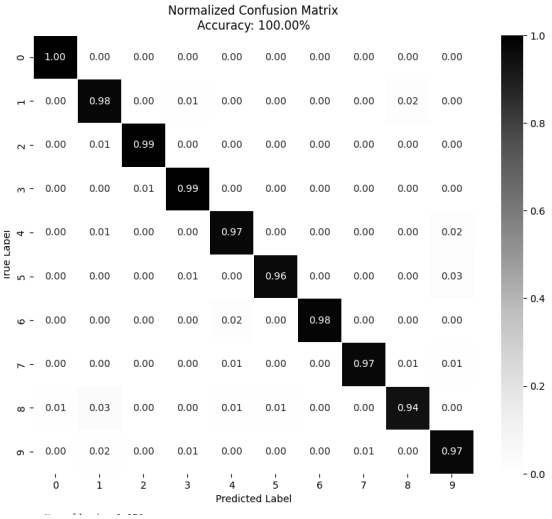

Fig. 6: Matriz, confusión y accuracy normalizado, validacion

C. *Caso c:*

- Test Loss: 0.141
- Test Accuracy: 0.956
- validaciones Accuracy Normalizado: 0.959
- entrenamientos Accuracy Normalizado:0.989

igual que el caso a, el modelo muestra una precisión consistente en los conjuntos de validación y prueba para 10 neuronas en la capa oculta, función de activación Tanh y 1000 épocas. La red en este caso no mostró impacto significativo al usar la función de activación Tanh esta funcionó bien para estos conjuntos de datos y estructura de la red, a precisión en el conjunto de prueba es del 95.6 porciento lo que significa que el modelo ha acertado correctamente en aproximadamente el 95.6 porciento de las instancias del conjunto de prueba.

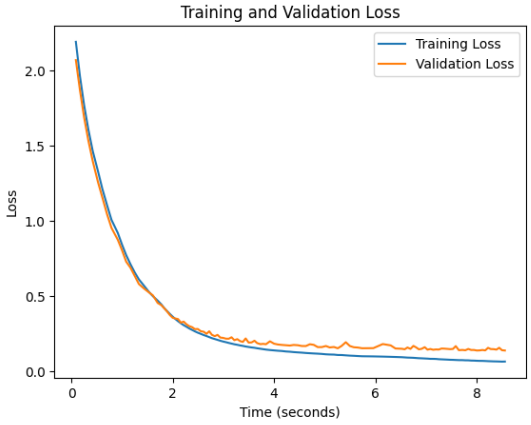

Fig. 7: Gráfico caso c

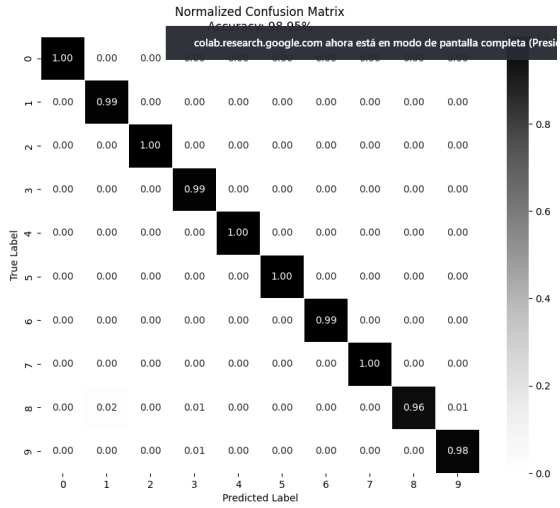

Fig. 8: Matriz, confusión y accuracy normalizado, entrenamiento

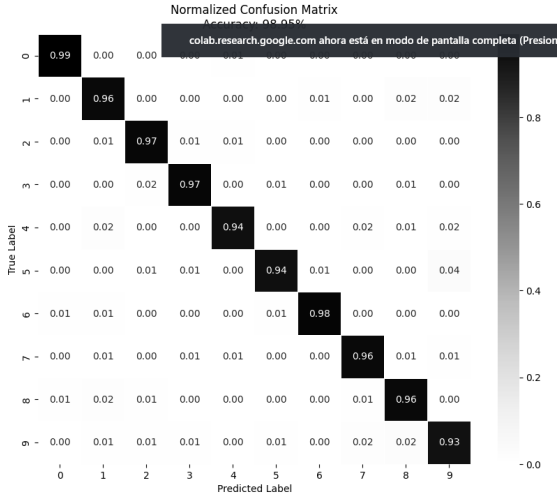

Fig. 9: Matriz, confusión y accuracy normalizado, validación

D. *Caso d:*

∗ Test Loss: 0.086

* Test Accuracy: 0.972
* validaciones Accuracy Normalizado: 0.972
* entrenamientos Accuracy Normalizado:0.997

El modelo de 40 neuronas en la capa oculta y función de activación Tanh, y 1000 épocas como máximo el modelo sigue siendo sólido en términos de generalización, con una alta precisión en los tres conjuntos. Esto sugiere que el modelo es capaz de mantener un buen rendimiento sin sobreajuste. La métrica de precisión normalizada en el conjunto de validación refleja la proporción de predicciones correctas en el conjunto de validación y está normalizada para tener en cuenta posibles desequilibrios en las clases.

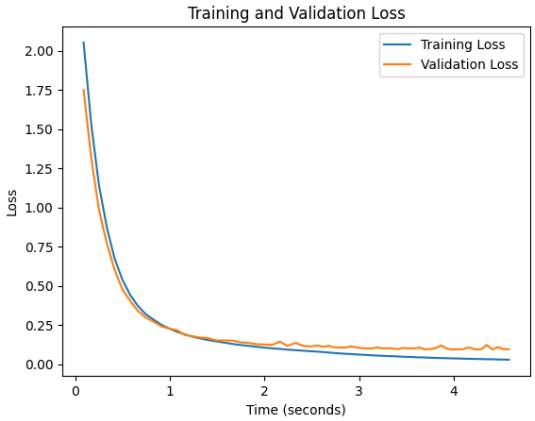

Fig. 10: Gráfico caso d

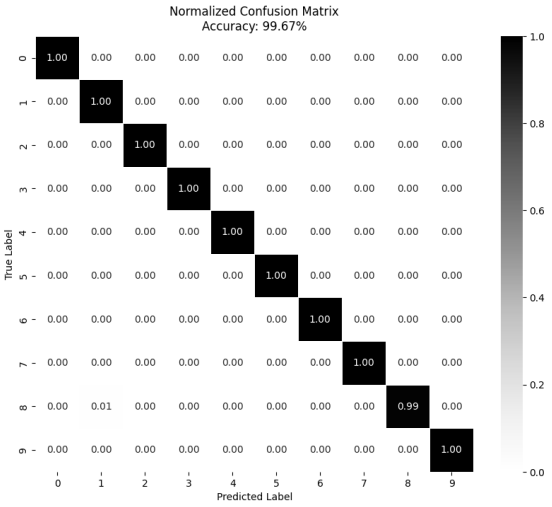

Fig. 11: Matriz, confusión y accuracy normalizado, entrenamiento

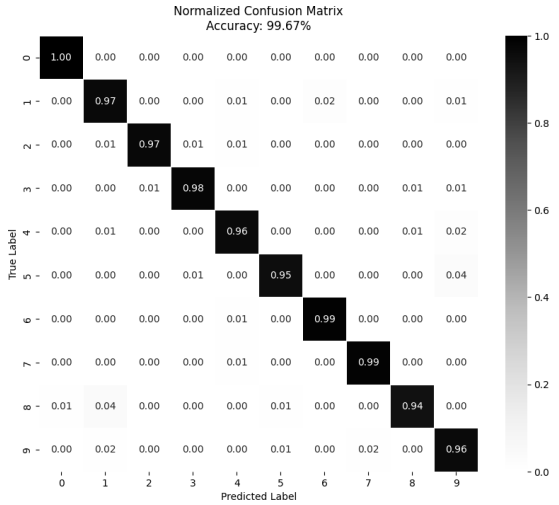

Fig. 12: Matriz, confusión y accuracy normalizado, validación

E. Caso e:

· Test Loss: 0.149
· Test Accuracy: 0.960
· validaciones Accuracy Normalizado: 0.959
· entrenamientos Accuracy Normalizado:0.985

el modelo de 2 capas ocultas con 10 y 10 neuronas cada una y función de activación ReLU, y 1000 épocas como máximo muestra una precisión consistente en los conjuntos de validación y prueba. El valor del loss en el conjunto de prueba es de 0.086. Este bajo valor de pérdida indica que el modelo ha realizado predicciones precisas en el conjunto de prueba. El error es una medida de cuán cerca están las predicciones del modelo de las etiquetas reales, y un valor bajo sugiere una alta precisión, como lo es en esta situación.
el test accuracy muestra que el modelo ha acertado correctamente, muestra que este tiene una alta precisión en el conjunto de prueba e indica la capacidad del modelo para clasificar los datos de manera efectiva.
su alta precisión en el entrenamiento es esperada, ya que el modelo se ha entrenado específicamente en estos datos y su precisión en la validación refleja la proporción de predicciones correctas

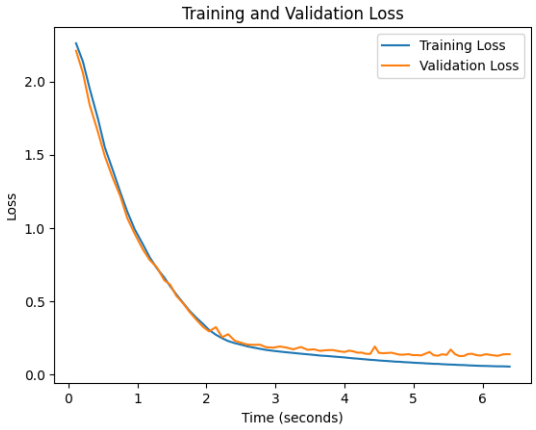

Fig. 13: Gráfico caso e

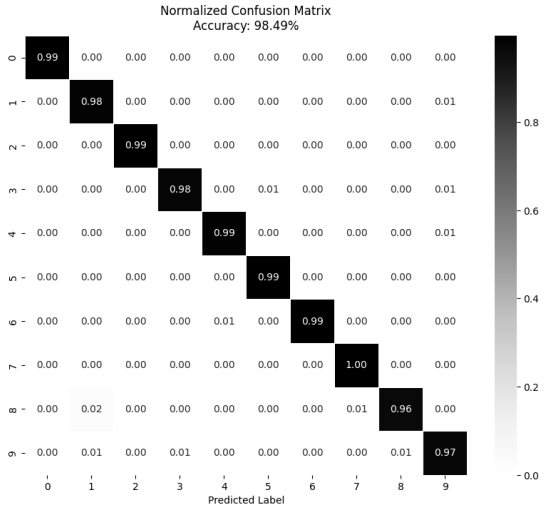

Fig. 14: Matriz, confusión y accuracy normalizado, entrenamiento

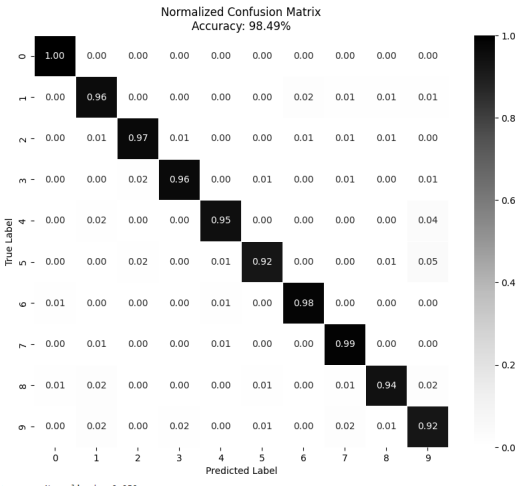

Fig. 15: Matriz, confusión y accuracy normalizado, validacion

F. Caso f:

· Test Loss: 0.074
· Test Accuracy: 0.983
· validaciones Accuracy Normalizado: 0.977
· entrenamientos Accuracy Normalizado:1.000

el modelo de 2 capas ocultas con 40 y 40 neuronas cada una y función de activación ReLU, y 1000 épocas como máximo demuestra un alto rendimiento en todos los conjuntos, lo que sugiere una buena generalización sin sobreajuste. La precisión en el conjunto de prueba es del 98.3 porciento, lo que significa que el modelo llegó a un nivel de precisión muy alto.
La precisión normalizada en el conjunto de entrenamiento es del 100 porciento, esto indica que el modelo se ha ajustado perfectamente a los datos de entrenamiento, obteniendo una precisión del 100 porciento.

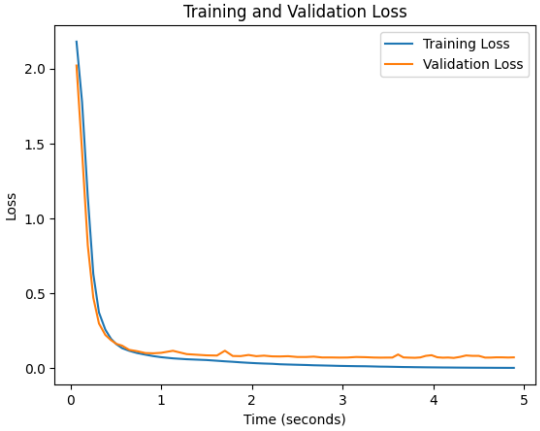

Fig. 16: Gráfico caso f

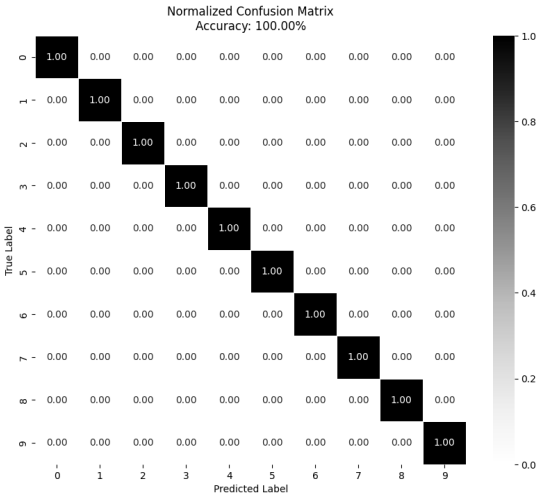

Fig. 17: Matriz, confusión y accuracy normalizado, entrenamiento

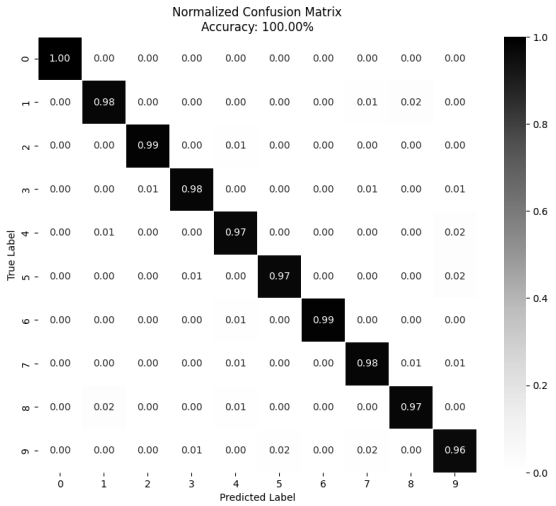

Fig. 18: Matriz, confusión y accuracy normalizado, validación

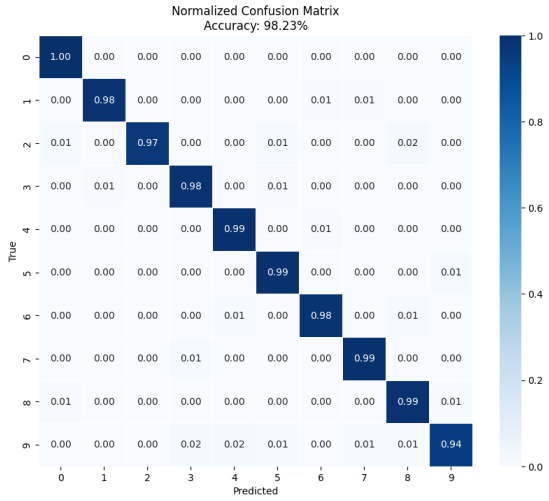

Fig. 19: Matriz, confusión y accuracy normalizado

Cuando se utiliza la mejor red encontrada en el conjunto de validación, que es aquella con la mayor precisión en validación, para calcular la matriz de confusión normalizada y el accuracy normalizado en el conjunto de prueba, se obtiene una evaluación del rendimiento del modelo en datos no vistos. Esto es importante para determinar si el modelo generaliza bien. Se calculó una matriz de confusión normalizada y un accuracy normalizado en el conjunto de prueba, obteniendo como resultado:
$NormalizedAccuracy(Test)$:
0.9822910867789092
indican que el modelo tiene un alto rendimiento en la clasificación de datos no vistos en el conjunto de prueba. El accuracy normalizado del 98.23 porciento sugiere que el modelo es capaz de clasificar con precisión la gran mayoría de las muestras en el conjunto de prueba, la matriz de confusión normalizada muestra cómo se distribuyen las predicciones en función de las clases reales, lo que es útil para evaluar el rendimiento en diferentes categorías.

V. Análisis

Analizando los resultados obtenidos para las diferentes configuraciones de la red neuronal, se observan diferencias dentro del rendimiento de la red.
-Efecto de Variar la Cantidad de Neuronas en la Capa Oculta:
Caso a: La red con 10 neuronas en la capa oculta y función de activación ReLU obtuvo un test loss de 0.125 y una precisión en el conjunto de prueba del 96.0 porciento. Aunque el rendimiento es bueno, no es el mejor.
Caso b: Al aumentar a 40 neuronas en la capa oculta con función de activación ReLU, la red mejoró significativamente. Obtuvo un test loss de 0.069 y una alta precisión del 98.0 porciento.
- Efecto de la Función de Activación tahn:
Caso c: Esta configuración utilizó 10 neuronas en la capa oculta y función de activación Tanh. Obtuvo un test loss de 0.141 y una precisión del 95.6 porciento. A pesar de tener menos neuronas, la función Tanh no entregó un rendimiento muy alto.
Caso d: Con 40 neuronas en la capa oculta y función de activación Tanh, la red logró un test loss de 0.086 y una precisión del 97.2 porciento. Tanh funcionó mejor que ReLU en este caso.
Caso e: Utilizando 2 capas ocultas, cada una con 10 neuronas y función de activación ReLU, la red obtuvo un test loss de 0.149 y una precisión del 96.0 porciento. Aunque la precisión es decente, no superó a los casos anteriores.
Caso f: En este caso, con 2 capas ocultas, cada una con 40 neuronas y función de activación ReLU, la red logró un test loss de 0.074 y una alta precisión del 98.3 porciento. Este caso resultó ser el mejor rendimiento de la red.
-Análisis de los Tiempos de Entrenamiento:
Los tiempos de entrenamiento varían entre los casos. Los casos d y f tienen los tiempos de entrenamiento más bajos, alrededor de 4.6 segundos. Por otro lado, los casos b y c tienen tiempos de entrenamiento más largos, alrededor de 8.5 segundos.
-Análisis de Matrices de Confusión y Accuracies en Validación:
Caso b y el caso f lograron las accuracies normalizadas más altas en validación, lo que significa que estos modelos tuvieron el mejor rendimiento en la clasificación de dígitos.
-Comparación entre Conjuntos de Validación y Prueba:
Caso a: La red muestra un rendimiento bastante consistente en los conjuntos de validación y prueba en comparación con el conjunto de entrenamiento.

Esto sugiere que el modelo generaliza de manera efectiva y no muestra signos de sobreajuste.
Caso b: El modelo sigue teniendo un rendimiento sólido, con una alta precisión en validación, entrenamiento y prueba. Esto indica una buena generalización y falta de sobreajuste.
Caso c: El modelo muestra una precisión consistente en los conjuntos de validación y prueba.
Caso d: El modelo sigue siendo sólido en términos de generalización, con una alta precisión en los tres conjuntos. Esto sugiere que el modelo es capaz de mantener un buen rendimiento sin sobreajuste.
Caso e: El modelo muestra una precisión consistente en los conjuntos de validación y prueba.
Caso f: El modelo demuestra un alto rendimiento en todos los conjuntos, lo que sugiere una buena generalización sin sobreajuste.

VI. CONCLUSIONES GENERALES

Concluyendo, en este análisis de diferentes configuraciones de redes neuronales para la clasificación de dígitos, se observó:
La cantidad de Neuronas en la Capa Oculta. Variar la cantidad de neuronas en la capa oculta tiene un impacto importante no así significativo en el rendimiento de la red. En general, un aumento en el número de neuronas, como en el "Caso b," mejora la precisión de la red en la clasificación. Una mayor cantidad de neuronas en la capa oculta permite a la red neuronal tener una mayor capacidad de representación. Cuantas más neuronas haya, mayor será la capacidad del modelo para aprender y capturar relaciones complejas entre las características de entrada y la salida deseada. Esto significa que una red con más neuronas en la capa oculta tiene la capacidad de aprender patrones más complejos y representar funciones más complicadas, a medida que aumenta la cantidad de neuronas en la capa oculta, la red neuronal se vuelve más flexible y puede ajustarse mejor a los datos de entrenamiento. y aunque una mayor cantidad de neuronas en la capa oculta puede aumentar la capacidad de la red para ajustarse a los datos de entrenamiento, también existe un mayor riesgo de sobreajuste. El sobreajuste ocurre cuando el modelo se ajusta demasiado a los datos de entrenamiento y no generaliza bien a nuevos datos. Si se utilizan demasiadas neuronas, la red puede memorizar el ruido o las peculiaridades de los datos de entrenamiento en lugar de aprender patrones generales. Esto puede resultar en una disminución del rendimiento en conjuntos de datos de validación o prueba. Las funciones de Activación. La elección de la función de activación también influye en el rendimiento. La función ReLU generalmente es efectiva y conduce a un aprendizaje más rápido, como se vio en el "Caso b." La función Tanh funcionó bien en el "Caso d," aunque no siempre superó a ReLU. Generalización. se observó que los modelos en los "Casos a," "b," "c," y "d" lograron una alta precisión en los conjuntos de validación y prueba, lo que indica una buena capacidad de generalización. Esto sugiere que estos modelos no sufrieron de sobreajuste a los datos de entrenamiento. El tiempo de Entrenamiento. los tiempos de entrenamiento varían entre las configuraciones. En general, los modelos más grandes y complejos tienden a requerir más tiempo de entrenamiento. La eficiencia en el tiempo de entrenamiento puede ser un factor importante en la elección del modelo. Análisis de Matrices de Confusión. Las matrices de confusión normalizadas y los accuracies en validación proporcionan información valiosa sobre el rendimiento de la red en la clasificación de dígitos. Los modelos que lograron las accuracies normalizadas más altas en validación demostraron un mejor rendimiento en general. por lo que en vista de lo detallado anteriormente se llega a la selección del mejor modelo. Basado en el análisis de los resultados, el "Caso f" con dos capas ocultas, cada una con 40 neuronas y función de activación ReLU, demostró ser el mejor modelo, con una alta precisión y generalización en validación y prueba.

REFERENCIAS

[1] Redes neuronales con phyton (Joaquín Amat Rodrigo Mayo, 2021) https://cienciadedatos.net/documentos/py35-redes-neuronales-python
[2] Ciencia de Datos. (s.f.). Redes neuronales con Python: guía completa. Recuperado de https://www.cienciadedatos.net/documentos/py35-redes-neuronales-python
[3] early stopping in PyTorch. https://stackoverflow.com/questions/71998978/early-stopping-in-pytorch
[4] Bootcamp AI. (2019, Noviembre, 14). Redes neuronales. Recuperado de https://bootcampai.medium.com/redes-neuronales-13349dd1a5bb
[5] ¿como funciona sotfmax?(2018, noviembre) https://www.youtube.com/watch?v=ma-F0RsMAjQ