Tarea N°2

Redes Neuronales

Cristian Herrera Badilla

COM4402 – Introducción a Inteligencia Artificial

Escuela de Ingeniería, Universidad de O'Higgins

sábado 28 de octubre 2023

Abstract*— En este experimento de clasificación de dígitos, se evaluaron seis modelos de redes neuronales, buscando el equilibrio entre precisión y eficiencia. Tras un análisis exhaustivo, se encontró que el "modelo_f" sobresalió al alcanzar una alta precisión en validación en tan solo 13 épocas, con un tiempo de entrenamiento excepcionalmente corto de alrededor de 0.99 segundos. En contraste, el "modelo_e" mostró un bajo rendimiento y detención temprana debido a su baja precisión. La elección del "modelo_f" se fundamenta en su equilibrio entre rendimiento y eficiencia, subrayando la importancia de evaluar y comparar diversos modelos para tareas de clasificación en el aprendizaje automático. Este análisis no solo proporciona una solución efectiva para la clasificación de dígitos, sino también sirve como un ejemplo esclarecedor en el desarrollo de proyectos de visión por computadora y aprendizaje automático.

I. Introducción

En este experimento, se exploraron varios modelos de redes neuronales convolucionales (CNN) para abordar esta tarea. La clasificación de dígitos no solo es una aplicación relevante en sí misma, sino que también sirve como un ejemplo esclarecedor de cómo abordar problemas de clasificación en el aprendizaje automático.

En el marco de este experimento, se investigaron conceptos clave relacionados con las redes neuronales artificiales, en particular, las redes neuronales profundas, y cómo se aplican en la clasificación. Además, se abordaron aspectos esenciales, como el preprocesamiento de datos, las funciones de pérdida y la evaluación del rendimiento del modelo.

El análisis y la comparación de varios modelos permitieron identificar cuál de ellos sobresalió en términos de precisión y eficiencia. En este informe, se presentarán los resultados de los modelos evaluados y se destacará el modelo con mejores resultados. Este análisis tiene como objetivo servir como un ejemplo práctico para futuros proyectos de aprendizaje automático y visión por computadora.

II. Marco teórico

La clasificación de dígitos es una tarea fundamental en el campo de la visión por computadora y el aprendizaje automático. Las redes neuronales, en particular las redes neuronales profundas, han demostrado ser eficaces para abordar esta tarea. A continuación, se presentan conceptos clave relacionados con el aprendizaje automático y la clasificación de imágenes, así como detalles sobre la matriz de confusión, la precisión (accuracy) y sus definiciones.

A. Redes Neuronales Artificiales (RNAs):

Las redes neuronales artificiales son modelos computacionales inspirados en la estructura y el funcionamiento de las redes neuronales biológicas. Están compuestas por capas de unidades de procesamiento llamadas neuronas o nodos. Cada neurona realiza una combinación lineal de las entradas, seguida de una función de activación no lineal. Las RNAs se utilizan para aproximación de funciones y tareas de aprendizaje automático, como la clasificación de imágenes.

B. Redes Neuronales Profundas:

Las redes neuronales profundas, comúnmente denominadas redes neuronales profundas o Deep Learning, son una clase de redes neuronales con múltiples capas ocultas. La profundidad de la red permite aprender representaciones jerárquicas de datos, lo que es especialmente beneficioso para

tareas de visión por computadora, como la clasificación de imágenes.

- Componentes de una Red Neuronal:

I. Neuronas (Nodos): Las neuronas son unidades de procesamiento que realizan operaciones matemáticas en las entradas. Cada neurona tiene pesos asociados que controlan la influencia de las entradas en la salida. La suma ponderada de las entradas se pasa a través de una función de activación.

II. Capas Ocultas: Las capas ocultas son capas intermedias entre la capa de entrada y la capa de salida. Cada capa oculta consta de múltiples neuronas. La combinación de múltiples capas ocultas permite a la red neuronal aprender representaciones jerárquicas de los datos.

III. Funciones de Activación: Las funciones de activación introducen no linealidad en la red. Comunes funciones de activación incluyen ReLU (Rectified Linear Unit), Sigmoide y Tangente Hiperbólica (Tanh). Estas funciones permiten que la red aprenda relaciones no lineales en los datos.

IV. Pesos y Sesgos: Cada conexión entre neuronas tiene un peso que controla la fuerza de la conexión. Los sesgos (biases) se utilizan para ajustar la salida de una neurona. Durante el entrenamiento, los pesos y los sesgos se ajustan para minimizar la función de pérdida.

C. Conjuntos de Datos:

Los conjuntos de datos juegan un papel crítico en la clasificación de dígitos escritos a mano. En este proyecto, se trabajó con tres conjuntos de datos:

Conjunto de Entrenamiento: Contiene imágenes de dígitos escritos a mano utilizados para entrenar los modelos.

Conjunto de Validación: Utilizado para ajustar hiperparámetros y evaluar el rendimiento del modelo durante el entrenamiento.

Conjunto de Prueba: Se utiliza para evaluar el rendimiento final del modelo en datos no vistos.

D. Preprocesamiento de Datos:

El preprocesamiento de datos es una etapa crucial en la clasificación de dígitos escritos a mano. Esto incluye la normalización de datos para asegurar que todas las características tengan una escala similar. La normalización facilita el entrenamiento de modelos y mejora el rendimiento.

E. Funciones de Pérdida:

En este proyecto, se utilizó la función de pérdida de entropía cruzada (Cross-Entropy Loss). Esta función de pérdida mide la discrepancia entre las predicciones del modelo y las etiquetas verdaderas del conjunto de entrenamiento. El objetivo es minimizar esta pérdida durante el entrenamiento.

F. Optimización:

Para entrenar modelos de redes neuronales, se utilizan algoritmos de optimización. En este caso, se empleó el optimizador Adam. El optimizador ajusta los pesos del modelo para minimizar la función de pérdida.

G. Evaluación del Modelo:

Después del entrenamiento, es crucial evaluar el rendimiento del modelo en conjuntos de datos de validación y prueba. Esto implica el cálculo de la precisión (accuracy), la matriz de confusión y otras métricas de rendimiento para evaluar cuán bien el modelo clasifica nuevas muestras.

H. Matriz de Confusión:

Una matriz de confusión es una herramienta fundamental en la evaluación de modelos de clasificación. Proporciona una representación

detallada de cómo un modelo clasifica las muestras en diferentes categorías o clases. La matriz de confusión se divide en cuatro categorías principales: Verdaderos Positivos, Falsos Positivos, Verdaderos Negativos y Falsos Negativos. Estos elementos permiten evaluar los aciertos y errores en la clasificación.

I. Precisión (Accuracy):

La precisión es una métrica fundamental utilizada para evaluar el rendimiento de un modelo de clasificación. Se define como la proporción de predicciones correctas realizadas por el modelo en relación con el número total de muestras en el conjunto de datos de prueba. La precisión se expresa como un porcentaje y varía entre 0% y 100%. Una alta precisión indica un rendimiento sólido en la clasificación, mientras que una baja precisión sugiere la necesidad de mejoras en el modelo.

III. METODOLOGÍA

1. Adquisicion y preprocesamiento de datos

Para llevar a cabo el análisis y desarrollo de un modelo de aprendizaje automático, se inicia con la adquisición de los datos de entrada, los cuales consisten en un conjunto de dígitos escritos a mano. Se utilizan dos conjuntos de datos: uno para entrenamiento y validación, y otro para pruebas.

- Se cargan los datos de entrenamiento y validación desde el archivo "1_digits_train.txt" y se asignan los nombres de las columnas, que incluyen 64 características (feat0 a feat63) y una etiqueta de clase (class).
- Se carga el conjunto de datos de prueba desde el archivo "1_digits_test.txt" con la misma estructura de columnas.

2. División en Conjuntos de Entrenamiento, Validación y Prueba

Para evaluar y ajustar el modelo de manera efectiva, se divide el conjunto de datos de entrenamiento y validación en dos subconjuntos: entrenamiento y validación. Esto se logra utilizando la función “train_test_split” con una proporción del 70% para entrenamiento y 30% para validación.

El conjunto de datos de entrenamiento y validación se divide en dos partes, donde el 70% se asigna al conjunto de entrenamiento (df_train) y el 30% al conjunto de validación (df_val). Esto se realiza para evaluar el rendimiento del modelo en un conjunto de datos independiente.

3. Normalización de datos

La normalización de datos es una práctica común en el preprocesamiento de datos para asegurar que todas las características tengan una escala similar y facilitar el entrenamiento de modelos de aprendizaje automático. Para ello, se aplica una transformación estándar (z-score) a las características del conjunto de entrenamiento y se replica en los conjuntos de validación y prueba.

Se realiza la normalización de los datos aplicando una transformación estándar utilizando “StandardScaler”. Esta transformación se ajusta a las características del conjunto de entrenamiento y se aplica tanto al conjunto de entrenamiento como al de validación y prueba, garantizando que todos los datos estén normalizados de manera consistente.

4. Creación de los modelos de red neuronal

En esta etapa se diseñan y configuran varios modelos de red neuronal con diferentes arquitecturas y configuraciones para evaluar su rendimiento en la clasificación de los dígitos escritos a mano. Cada modelo se define utilizando la biblioteca PyTorch y consta de capas de neuronas interconectadas.

- Caso (a): Modelo con 10 neuronas en la capa oculta, función de activación ReLU y 1000 épocas.
- Caso (b): Modelo con 40 neuronas en la capa oculta, función de activación ReLU y 1000 épocas.
- Caso (c): Modelo con 10 neuronas en la capa oculta, función de activación Tangente Hiperbólica (Tanh) y 1000 épocas.
- Caso (d): Modelo con 40 neuronas en la capa oculta, función de activación Tangente Hiperbólica (Tanh) y 1000 épocas.

- Caso (e): Modelo con 2 capas ocultas, cada una con 10 neuronas, función de activación ReLU y 1000 épocas.
- Caso (f): Modelo con 2 capas ocultas, cada una con 40 neuronas, función de activación ReLU y 1000 épocas.

Estos modelos representan diferentes configuraciones de capas ocultas, funciones de activación y número de épocas para el entrenamiento. Se procederá a entrenar y evaluar el rendimiento de cada modelo en las siguientes secciones del análisis.

En el siguiente paso se preparan los datos para su uso en la construcción y entrenamiento de modelos de aprendizaje automático. Se crean conjuntos de datos y dataloaders que permitirán cargar los datos de manera eficiente durante el proceso de entrenamiento y evaluación.

5. Creación de datasets

Se transforman los conjuntos de entrenamiento, validación y prueba en formatos adecuados para PyTorch. Los datos se dividen en características (feats) y etiquetas (labels), y se almacenan en una lista de diccionarios. Cada elemento del diccionario contiene un par de características y etiquetas para una muestra de datos.

- Conjunto de Entrenamiento:
- Conjunto de Validación
- Conjunto de Prueba

6. Creación de dataloaders

Se crean dataloaders utilizando la biblioteca "torch.utils.data.DataLoader". Los dataloaders permiten cargar los datos en lotes (batches) durante el entrenamiento y la evaluación de los modelos. Cada dataloader especifica el conjunto de datos correspondiente, el tamaño de lote (batch_size), si se deben mezclar los datos (shuffle) y el número de trabajadores (num_workers) para cargar datos de manera paralela.

- Dataloader de Entrenamiento
- Dataloader de Validación
- Dataloader de Prueba

7. Page Numbers, Headers and Footers

Page numbers, headers and footers must not be used.

8. Funciones para Entrenamiento y Evaluación de Modelos

Aquí se definen y describen las funciones clave utilizadas para el entrenamiento y evaluación de los modelos de redes neuronales. Estas funciones son esenciales para medir el rendimiento de los modelos y evaluar su capacidad para realizar tareas de clasificación.

- Función "Train"

Se encarga de entrenar el modelo de red neuronal. Realiza el entrenamiento en un bucle de épocas (en este caso, hasta 1000 épocas) y realiza el ajuste de los pesos del modelo a medida que se procesan los lotes de datos del dataloader de entrenamiento. La función realiza un seguimiento de las pérdidas de entrenamiento y validación, calcula la precisión en cada época y utiliza una estrategia de parada temprana si la pérdida de validación comienza a aumentar.

Entradas:

1. model: El modelo de red neuronal.
2. dataloader_train: Dataloader que contiene el conjunto de entrenamiento.
3. dataloader_val: Dataloader que contiene el conjunto de validación.
4. labels_train: Etiquetas del conjunto de entrenamiento.
5. labels_val: Etiquetas del conjunto de validación.
6. device: Dispositivo de cómputo (en este caso GPU) en el que se ejecuta el entrenamiento.
7. criterion: La función de pérdida utilizada para calcular la pérdida del modelo.
8. optimizer: El optimizador utilizado para ajustar los pesos del modelo.

- Función calculate_normalized_confusion_matrix_and_accuracy:

Esta función se utiliza para calcular la matriz de confusión normalizada y la precisión del modelo en un conjunto de datos. Toma el modelo entrenado, el dataloader del conjunto de datos y el dispositivo de

cómputo como entradas, y devuelve la precisión y la matriz de confusión normalizada.

Entradas:

1. model: El modelo de red neuronal entrenado.
2. dataloader: Dataloader que contiene el conjunto de datos para el que se calculará la matriz de confusión y la precisión.
3. device: Dispositivo de cómputo en el que se ejecutará la evaluación.

- Función plot_normalized_confusion_matrix:

Se utiliza para visualizar la matriz de confusión normalizada y la precisión del modelo. Toma la matriz de confusión normalizada, la precisión y un título como entradas, y muestra la matriz de confusión en forma de gráfico.

Entradas:

1. conf_matrix: Matriz de confusión normalizada.
2. accuracy: Precisión del modelo.
3. titlee: Título para el gráfico de la matriz de confusión.

Estas funciones son esenciales para el entrenamiento, evaluación y visualización de los resultados de los modelos de aprendizaje automático en la tarea de clasificación de dígitos escritos a mano.

Finalmente, la etapa de entrenamiento.

9. Entrenamiento y evaluación de modelos

se procede a entrenar el modelo de red neuronal y evaluar su rendimiento en los conjuntos de entrenamiento y validación. El proceso se repite para varios casos de modelos con diferentes configuraciones, pero el enfoque será en el primer caso (Caso "a") como ejemplo, ya que los demás casos siguen un procedimiento similar.

Se selecciona el modelo correspondiente al "Caso (a)" y se traslada al dispositivo de cómputo deseado (en este caso, GPU). Se configura la función de pérdida (criterion) como la pérdida de entropía cruzada y el optimizador (optimizer) como el optimizador Adam con una tasa de aprendizaje de 0.001.

Codigo

A continuación, se procede a entrenar el modelo utilizando la función “train” definida previamente. La función registra el progreso del entrenamiento, calcula las pérdidas de entrenamiento y validación, y detiene el entrenamiento temprano si la pérdida de validación comienza a aumentar.

Código

Una vez finalizado el entrenamiento, se evalúa el rendimiento del modelo en los conjuntos de entrenamiento y validación utilizando la función “calculate_normalized_confusion_matrix_and_accuracy”. Esta función calcula la precisión y la matriz de confusión normalizada para cada conjunto de datos.

1. Conjunto de Entrenamiento
2. Conjunto de Validación

Este proceso de entrenamiento y evaluación se repite para otros casos de modelos con configuraciones diferentes, y los resultados se utilizan para comparar y seleccionar el modelo con el mejor rendimiento en la tarea de clasificación de dígitos.

IV. Resultados

Se presentan las tablas de resultados y matriz de confusión de cada uno de los modelos plateados.

1. Tabla de “loss and accuracy”

A. Caso (a)

Epoch	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy
0	22.219	21.096	21.10%	34.79%
1	20.371	19.237	29.40%	40.15%
2	18.472	17.243	35.08%	45.19%
3	16.536	15.249	40.20%	49.79%
4	14.658	13.442	44.88%	53.73%
5	12.884	11.765	48.90%	57.04%
6	11.237	10.388	52.53%	60.14%
7	0.9786	0.8924	56.00%	63.14%
8	0.8485	0.7550	59.15%	65.78%
9	0.7323	0.6783	61.87%	68.09%
10	0.6344	0.5694	64.27%	70.07%
11	0.5536	0.4992	66.40%	71.81%
12	0.4896	0.4413	68.27%	73.32%
13	0.4372	0.4030	69.92%	74.68%
14	0.3964	0.3700	71.39%	75.91%
15	0.3632	0.3377	72.71%	77.01%
16	0.3340	0.3033	73.89%	77.99%
17	0.3089	0.2929	74.96%	78.89%
18	0.2880	0.2657	75.93%	79.70%
19	0.2694	0.2453	76.83%	80.45%
20	0.2529	0.2416	77.66%	81.14%
21	0.2389	0.2299	78.42%	81.78%
22	0.2264	0.2173	79.13%	82.36%

B. Caso (b)

Epoch	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy
0	21.550	19.530	26.33%	51.03%
1	17.486	15.300	44.81%	62.18%
2	13.255	11.208	55.40%	68.15%
3	0.9531	0.8202	62.15%	72.55%
4	0.6784	0.5903	67.28%	75.89%
5	0.4965	0.4586	71.16%	78.51%
6	0.3832	0.3506	74.24%	80.53%
7	0.3104	0.2974	76.69%	82.16%
8	0.2621	0.2487	78.67%	83.51%
9	0.2274	0.2233	80.31%	84.64%
10	0.2016	0.2061	81.69%	85.58%
11	0.1823	0.1849	82.88%	86.40%
12	0.1652	0.1737	83.90%	87.10%
13	0.1513	0.1666	84.80%	87.71%
14	0.1400	0.1527	85.61%	88.25%

C. Caso (c)

Epoch	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy
0	22.255	20.911	20.71%	38.16%
1	19.834	18.641	33.88%	47.62%
2	17.780	16.773	42.80%	54.18%
3	16.006	15.071	49.75%	59.31%
4	14.449	13.774	55.38%	63.51%
5	13.088	12.354	59.97%	66.93%
6	11.884	11.263	63.58%	69.66%
7	10.804	10.212	66.55%	71.83%
8	0.9859	0.9497	68.98%	73.72%
9	0.8991	0.8551	71.07%	75.37%
10	0.8212	0.7881	72.87%	76.79%
11	0.7529	0.7201	74.44%	77.99%
12	0.6915	0.6601	75.83%	79.04%
13	0.6380	0.6134	77.05%	79.98%
14	0.5898	0.5615	78.12%	80.82%
15	0.5479	0.5275	79.08%	81.60%
16	0.5107	0.4873	79.94%	82.29%
17	0.4779	0.4646	80.71%	82.91%
18	0.4499	0.4398	81.41%	83.48%
19	0.4237	0.4131	82.04%	84.01%
20	0.4013	0.3959	82.63%	84.49%
21	0.3800	0.3873	83.17%	84.94%
22	0.3619	0.3667	83.68%	85.34%
23	0.3452	0.3487	84.15%	85.72%
24	0.3301	0.3360	84.58%	86.07%
25	0.3159	0.3135	84.99%	86.41%

D. Caso (d)

Epoch	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy
0	20.382	17.441	38.89%	65.06%
1	15.146	12.859	57.53%	73.75%
2	11.264	0.9663	66.87%	78.03%
3	0.8499	0.7328	72.43%	80.96%
4	0.6573	0.5725	76.12%	83.10%
5	0.5235	0.4673	78.84%	84.66%
6	0.4302	0.3917	80.91%	85.91%
7	0.3639	0.3323	82.55%	86.91%
8	0.3140	0.2945	83.91%	87.72%
9	0.2763	0.2600	85.05%	88.41%
10	0.2471	0.2370	86.03%	89.01%
11	0.2242	0.2290	86.87%	89.53%
12	0.2045	0.1961	87.61%	90.00%
13	0.1885	0.1910	88.27%	90.39%
14	0.1746	0.1776	88.84%	90.74%
15	0.1629	0.1685	89.36%	91.04%
16	0.1526	0.1577	89.81%	91.32%

E. Caso (e)

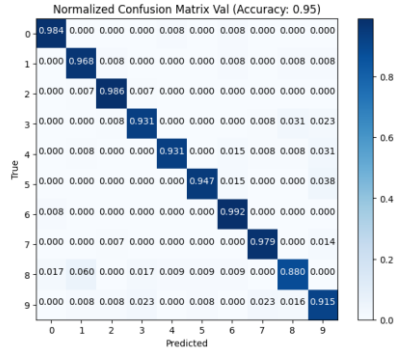
Epoch	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy
0	22.987	22.765	8.97%	10.27%
1	22.268	21.715	12.02%	16.05%
2	21.179	20.267	17.94%	22.94%
3	19.439	18.151	23.49%	28.39%
4	16.890	15.057	28.26%	34.50%
5	13.926	12.098	34.37%	40.50%
6	11.187	0.9659	39.45%	45.06%
7	0.9178	0.8117	43.63%	48.75%
8	0.7701	0.6863	47.47%	52.67%
9	0.6572	0.5778	51.20%	56.21%
10	0.5609	0.5055	54.54%	59.22%
11	0.4832	0.4212	57.46%	61.84%
12	0.4154	0.3899	60.01%	64.14%
13	0.3641	0.3318	62.27%	66.14%
14	0.3238	0.2887	64.28%	67.91%

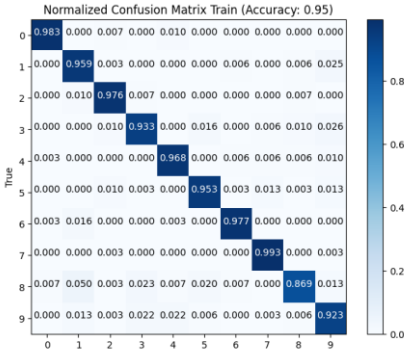
F. Caso (f)

Epoch	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy
0	21.875	20.323	29.62%	54.71%
1	18.090	15.272	47.52%	64.79%
2	12.175	0.9074	57.77%	71.37%
3	0.6974	0.5153	64.99%	75.88%
4	0.4150	0.3396	70.14%	78.97%
5	0.2847	0.2508	74.01%	81.46%
6	0.2160	0.1952	76.99%	83.32%
7	0.1763	0.1937	79.34%	84.78%
8	0.1497	0.1529	81.23%	85.98%
9	0.1300	0.1386	82.76%	86.96%
10	0.1158	0.1304	84.06%	87.79%
11	0.1024	0.1274	85.17%	88.49%
12	0.0900	0.1200	86.13%	89.11%

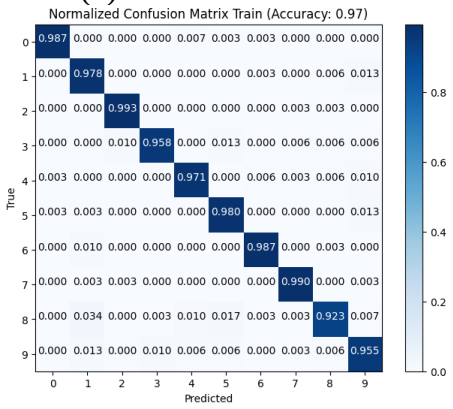
2. Matriz de confusión

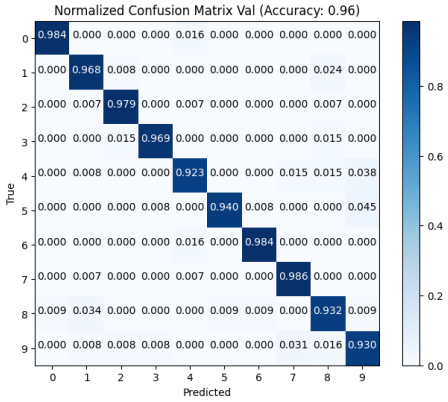
A. Caso (a)



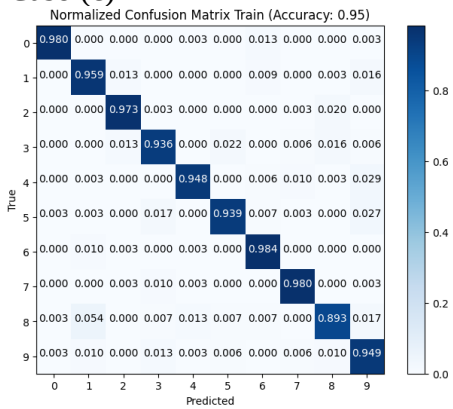


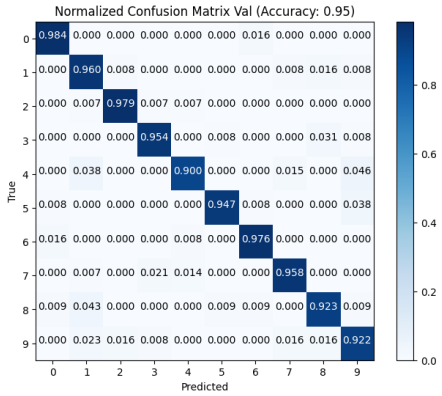
B. Caso (b)



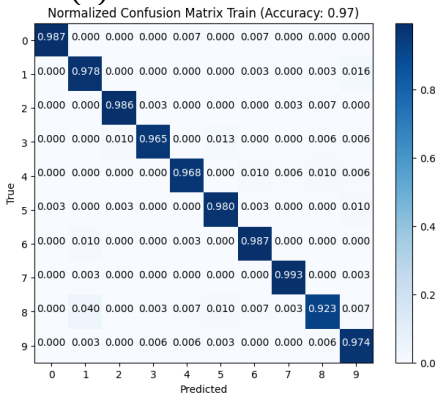


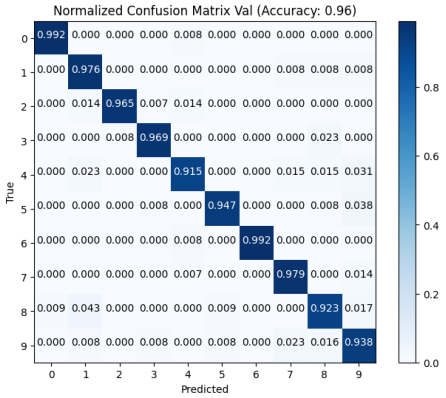
C. Caso (c)



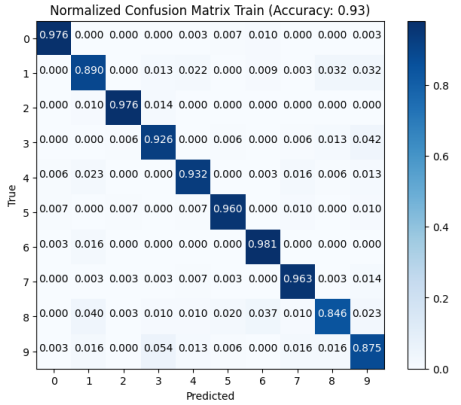


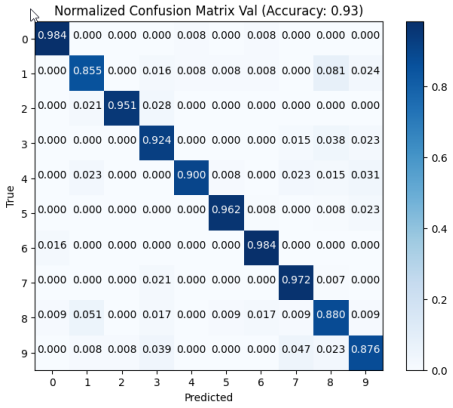
D. Caso (d)



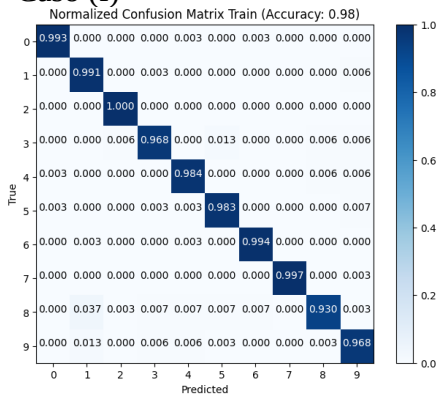


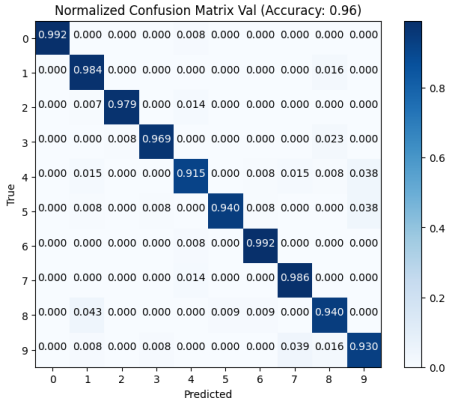
E. Caso (e)





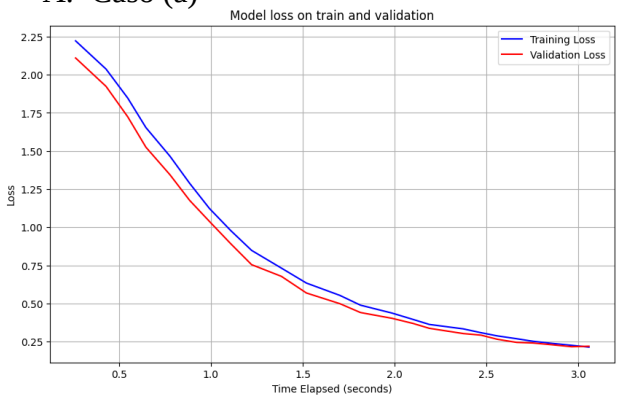
F. Caso (f)



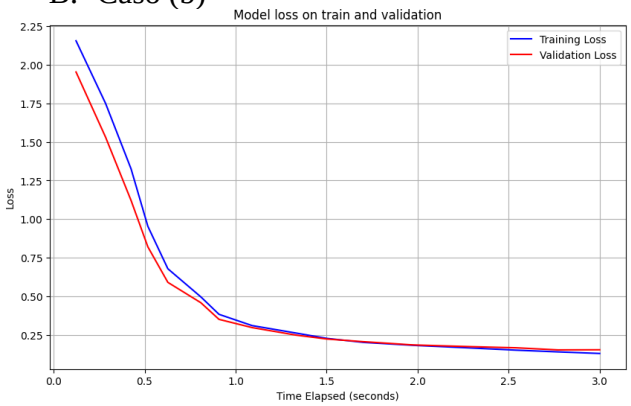


3. Gráfico de "loss" entrenamiento y validación en función del tiempo

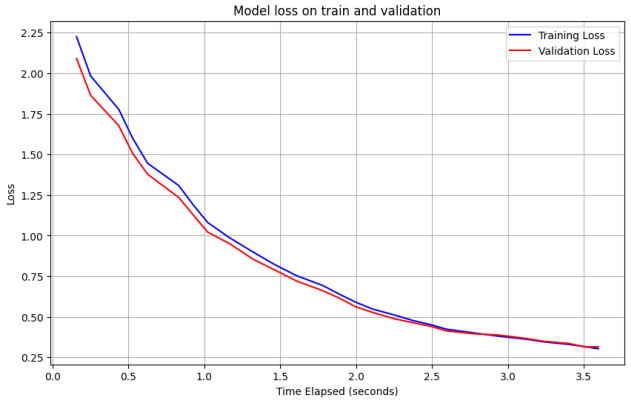
A. Caso (a)



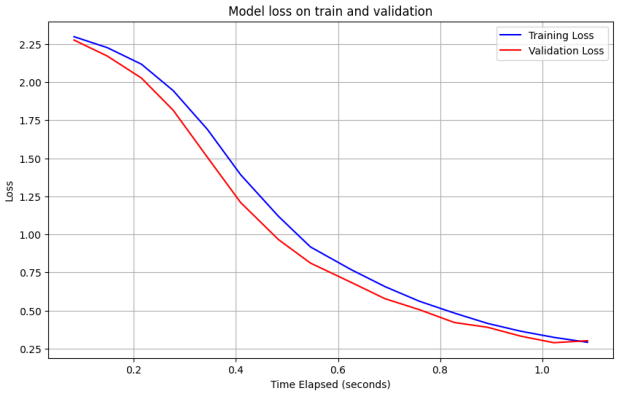
B. Caso (b)



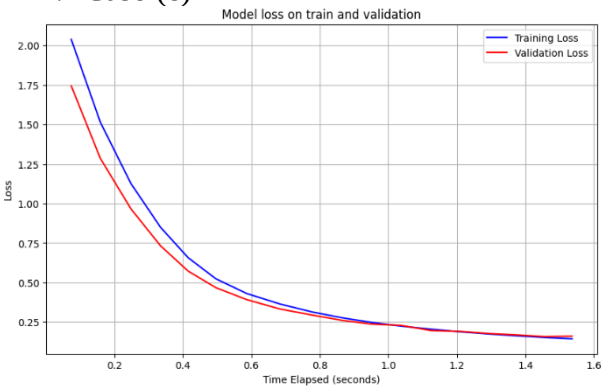
C. Caso (c)



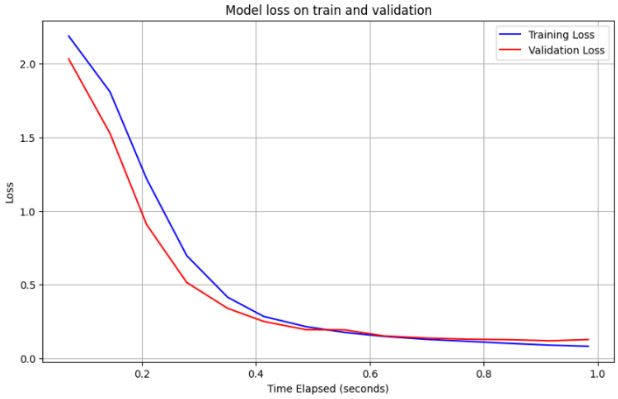
D. Caso (d)



E. Caso (e)



F. Caso (f)



V. Análisis

Analizando los modelos proporcionados, podemos determinar cuál es el modelo con mejor desempeño.

A continuación, se presenta un resumen de los resultados clave de cada modelo:

A. *modelo_a: Este modelo logró una precisión moderada en la validación después de 23 épocas y mostró un buen rendimiento general.*
B. *modelo_b: Alcanzó una alta precisión en la validación después de 15 épocas y tuvo un tiempo de entrenamiento de alrededor de 3 segundos.*
C. *modelo_c: También logró una alta precisión en la validación después de 26 épocas con un tiempo de entrenamiento de aproximadamente 3.65 segundos.*
D. *modelo_d: Obtuvo una alta precisión en la validación después de 17 épocas y tuvo un tiempo de entrenamiento relativamente corto de alrededor de 1.54 segundos.*
E. *modelo_e: Este modelo no funcionó bien, alcanzando una precisión baja en la validación, lo que resultó en una detención temprana en la época 15.*
F. *modelo_f: Fue el modelo más eficiente, alcanzando una alta precisión en la validación después de 13 épocas y con el tiempo de entrenamiento más corto de aproximadamente 0.99 segundos.*

Análisis Comparativo:

Mejor Rendimiento: Los modelos modelo_b, modelo_c, modelo_d y modelo_f alcanzaron altos niveles de precisión en la validación. Entre ellos, modelo_f se destacó por su eficiencia en términos de tiempo de entrenamiento.

Peor Rendimiento: modelo_e fue el modelo menos efectivo, con una precisión muy baja en la validación y una detención temprana.

Rendimiento General: Los modelos modelo_b y modelo_d lograron una alta precisión en la validación con tiempos de entrenamiento moderados. Estos modelos se destacan como opciones sólidas.

Eficiencia: modelo_f fue el modelo más eficiente en términos de tiempo de entrenamiento, pero también alcanzó una precisión alta. Esto lo convierte en una opción atractiva.

VI. Conclusión

Durante este experimento, se evaluaron seis modelos de redes neuronales (modelos a, b, c, d, e y f) para determinar cuál ofrece el mejor rendimiento en términos de precisión y eficiencia.

El modelo_f destacó como el modelo más eficiente y preciso. Alcanzó una alta precisión en la validación después de solo 13 épocas, y su tiempo de entrenamiento fue el más corto de todos los modelos, aproximadamente 0.99 segundos. Esto lo convierte en la elección preferida debido a su equilibrio entre rendimiento y eficiencia.

El modelo_e mostró el peor rendimiento en este experimento. Experimentó una detención temprana debido a su baja precisión en la validación.

Es importante seleccionar cuidadosamente el modelo de aprendizaje automático, ya que diferentes configuraciones y hiperparámetros pueden tener un impacto significativo en el rendimiento y la eficiencia. El modelo_f se destacó como el mejor modelo en términos de equilibrio entre precisión y tiempo de entrenamiento, mientras que el modelo_e requiere una revisión y optimización significativas. Estos hallazgos resaltan la importancia de la evaluación comparativa en el desarrollo de modelos de aprendizaje automático para tareas de clasificación.

Referencias

[1] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[2] Szeliski, R. (2010). Computer Vision: Algorithms and Applications. Springer.

[3] Aggarwal, C. (2018). Neural Networks and Deep Learning: A Textbook. Springer.

[4] Hastie, Trevor, Robert Tibshirani, Jerome H. Friedman, and Jerome H. Friedman. The elements of statistical learning: Data Mining, Inference, and Prediction. Vol. 2. New York: Springer, 2009.

[5] Francois Chollet, Deep learning with Python, Second Edition

[6] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

[7] Simon Prince. Computer Vision: Models, Learning, and Inference. Cambridge University Press 2012