

Mini proyecto: Redes neuronales

Aplicaciones en Ciencia de Datos e Inteligencia Artificial

1. Introducción:

En este proyecto, exploraremos la implementación y el entrenamiento de redes neuronales artificiales utilizando PyTorch. Compararemos el rendimiento de un perceptrón multicapa con redes neuronales convolucionales. Para ello, llevaremos a cabo experimentos con diferentes arquitecturas y estrategias de entrenamiento con el objetivo de maximizar la capacidad predictiva en un conjunto de datos de dígitos escritos a mano.

2. Perceptrón multicapa:

2.1 Objetivo 1:

Evaluar diversas arquitecturas para el perceptrón multicapa y seleccionar el modelo que obtenga el mejor rendimiento en términos de accuracy.

Metodología 1:

Modificar los siguientes elementos:

- ✓ Número de neuronas por capa.
- ✓ Cantidad de capas ocultas.
- ✓ Tipo de función de activación utilizada.

Elabore un informe detallado de los experimentos realizados, incluyendo las configuraciones probadas, los resultados obtenidos y las conclusiones que justifiquen la selección del modelo final.

Ensayo 1:

Tiene como parámetros de la arquitectura son los siguiente: Info base.

- Capa de entrada de 28x28 pixeles
- Capa oculta 1 de 200 neuronas con ReLU.
- Capa oculta 2: de 256 neuronas con ReLU.
- Capa salida de 10 neuronas para las clases del dataset MINIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos:

- batch_size = 64 # Tamaño de lote
- learning_rate = 0.001 # Tasa de aprendizaje
- epochs = 10 # Número de épocas de entrenamiento

Se detalla la pérdida de cada una de las 10 épocas, se visualiza que es variable y va mejorando relativamente en cada una (con unos outliers).

```
✓ 2m 1.3s
Época [1/10], Pérdida: 0.0518
Época [2/10], Pérdida: 0.1243
Época [3/10], Pérdida: 0.0418
Época [4/10], Pérdida: 0.1208
Época [5/10], Pérdida: 0.1263
Época [6/10], Pérdida: 0.3031
Época [7/10], Pérdida: 0.1054
Época [8/10], Pérdida: 0.2616
Época [9/10], Pérdida: 0.0147
Época [10/10], Pérdida: 0.2910
```

De la evaluación tenemos: un **97.67%** lo cual anda bastante bien.

Accuracy en el conjunto de prueba: 97.67%

Ensayo 2:

Tiene como parámetros de la arquitectura son los siguiente: Se modifican cantidad de neuronas respecto al ensayo 1.

- Capa de entrada de 28x28 pixeles
- Capa oculta 1 de 110 neuronas con ReLU.
- Capa oculta 2: de 50 neuronas con ReLU.
- Capa salida de 10 neuronas para las clases del dataset MINIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos: se mantienen valores de ensayo 1

- batch_size = 64 # Tamaño de lote
- learning_rate = 0.001 # Tasa de aprendizaje
- epochs = 10 # Número de épocas de entrenamiento

Se detalla la pérdida de cada una de las 10 épocas, se visualiza que es variable, no presentan una mejora visible. Solo 3 épocas con valores menores de 0.04 y 0.08

```
Época [1/10], Pérdida: 0.1252
Época [2/10], Pérdida: 0.0487
Época [3/10], Pérdida: 0.2579
Época [4/10], Pérdida: 0.2893
Época [5/10], Pérdida: 0.0413
Época [6/10], Pérdida: 0.1619
Época [7/10], Pérdida: 0.1143
Época [8/10], Pérdida: 0.0862
Época [9/10], Pérdida: 0.1399
Época [10/10], Pérdida: 0.1612
```

De la evaluación tenemos: un **96.83%** siendo menor al ensayo 1. Por lo que no está siendo bien evaluado este modelo con la cantidad de neuronas dadas. Tiene menor capacidad de aprendizaje.

... Accuracy en el conjunto de prueba: 96.83%

Ensayo 3:

Tiene como parámetros de la arquitectura son los siguiente: No se modifican la cantidad de neuronas respecto al ensayo 1 y 2, pero se modifica la cantidad de capas ocultas, agregando una capa.

- Capa de entrada de 28x28 pixeles
- Capa oculta 1 de 110 neuronas con ReLU.
- Capa oculta 2: de 50 neuronas con ReLU.
- Capa oculta 3: de 10 neuronas con ReLU.
- Capa salida de 10 neuronas para las clases del dataset MNIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos: se mantienen valores de ensayo 1 y ensayo 2.

- batch_size = 64 # Tamaño de lote
- learning_rate = 0.001 # Tasa de aprendizaje
- epochs = 10 # Número de épocas de entrenamiento

Se detalla la pérdida de cada una de las 10 épocas, se visualiza que la pérdida aumenta considerablemente.

```
Época [1/10], Pérdida: 0.4817
Época [2/10], Pérdida: 0.2144
Época [3/10], Pérdida: 0.6059
Época [4/10], Pérdida: 0.3708
Época [5/10], Pérdida: 0.3627
Época [6/10], Pérdida: 0.3986
Época [7/10], Pérdida: 0.3403
Época [8/10], Pérdida: 0.4420
Época [9/10], Pérdida: 0.2423
Época [10/10], Pérdida: 0.3597
```

De la evaluación tenemos: un **96.48%** siendo menor al ensayo 1, 2 y 3. Por lo que no está siendo bien evaluado este modelo con la cantidad de neuronas y las capas dadas. Tiene menor capacidad de aprendizaje que los anteriores. Menos capas, da un modelo más simple.

Accuracy en el conjunto de prueba: 96.48%

Ensayo 4.1:

Tiene como parámetros de la arquitectura son los siguiente: No se modifican la cantidad de neuronas respecto al ensayo 1 y 2. Se mantienen solo 2 capas ocultas, ya que tenía

mejor comportamiento. Se modifica tipo de función de activación utilizada a Leaky Relu en todas las capas ocultas.

- Capa de entrada de 28x28 pixeles
- Capa oculta 1 de 110 neuronas con Leaky Relu.
- Capa oculta 2: de 50 neuronas con Leaky Relu.
- Capa salida de 10 neuronas para las clases del dataset MINIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos: se mantienen valores de ensayo 1 y ensayo 2.

- batch_size = 64 # Tamaño de lote
- learning_rate = 0.001 # Tasa de aprendizaje
- epochs = 10 # Número de épocas de entrenamiento

Se detalla la pérdida de cada una de las 10 épocas, se visualiza que el tiempo aumento un poco, sin embargo, los valores de pérdida anduvieron relativamente mejor que los ensayos anteriores.

```
✓ 2m 1.8s
Época [1/10], Pérdida: 0.0864
Época [2/10], Pérdida: 0.2157
Época [3/10], Pérdida: 0.3219
Época [4/10], Pérdida: 0.1564
Época [5/10], Pérdida: 0.2038
Época [6/10], Pérdida: 0.0477
Época [7/10], Pérdida: 0.0723
Época [8/10], Pérdida: 0.2338
Época [9/10], Pérdida: 0.0305
Época [10/10], Pérdida: 0.0552
```

De la evaluación tenemos: un **97.26%** algo que se pudo observar con las pérdidas, hasta ahora es el mejor comportamiento luego del primero ensayo donde se obtuvo un accuracy de 97,65% sin embargo, el tiempo es un factor a considerar para dejarlo o no.

Accuracy en el conjunto de prueba: 97.26%

Ensayo 4.2:

Tiene como parámetros de la arquitectura son los siguiente: No se modifican la cantidad de neuronas respecto al ensayo 1 y 2. Se mantienen solo 2 capas ocultas, ya que tenía mejor comportamiento. Se modifica tipo de función de activación utilizada a sigmoide todas las capas ocultas.

- Capa de entrada de 28x28 pixeles
- Capa oculta 1 de 110 neuronas con sigmoide.
- Capa oculta 2: de 50 neuronas con sigmoide.
- Capa salida de 10 neuronas para las clases del dataset MINIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos: se mantienen valores de ensayo 1 y ensayo 2.

- batch_size = 64 # Tamaño de lote
- learning_rate = 0.001 # Tasa de aprendizaje
- epochs = 10 # Número de épocas de entrenamiento

Se detalla la pérdida de cada una de las 10 épocas, se visualiza que el tiempo es relativamente menor al anterior, se visualiza que la pérdida es variable, pero con rangos mas altos, por lo que no parece ser una buena opción, solo 3 épocas con valores menores de 0.01, 0.07 y 0.09.



A terminal window showing the progress of a training process over 10 epochs. At the top, there is a green checkmark and the text '1m 39.8s'. Below this, each line represents an epoch, showing the epoch number in brackets followed by the loss value. The losses are: 0.3568, 0.3116, 0.1816, 0.2611, 0.3287, 0.1303, 0.0740, 0.0980, 0.0110, and 0.2835.

✓ 1m 39.8s
Época [1/10], Pérdida: 0.3568
Época [2/10], Pérdida: 0.3116
Época [3/10], Pérdida: 0.1816
Época [4/10], Pérdida: 0.2611
Época [5/10], Pérdida: 0.3287
Época [6/10], Pérdida: 0.1303
Época [7/10], Pérdida: 0.0740
Época [8/10], Pérdida: 0.0980
Época [9/10], Pérdida: 0.0110
Época [10/10], Pérdida: 0.2835

De la evaluación tenemos: un **96.80%** algo que se pudo observar con las pérdidas, volviendo a bajar y siendo cercano a ensayos 2 y 3. Por lo que no está siendo bien evaluado este modelo con la cantidad de neuronas, clases y función de activación dada. Tiene menor capacidad de aprendizaje. Por lo tanto, se rechaza como función de activación para este caso.

Accuracy en el conjunto de prueba: 96.80%

Ensayo 4.3:

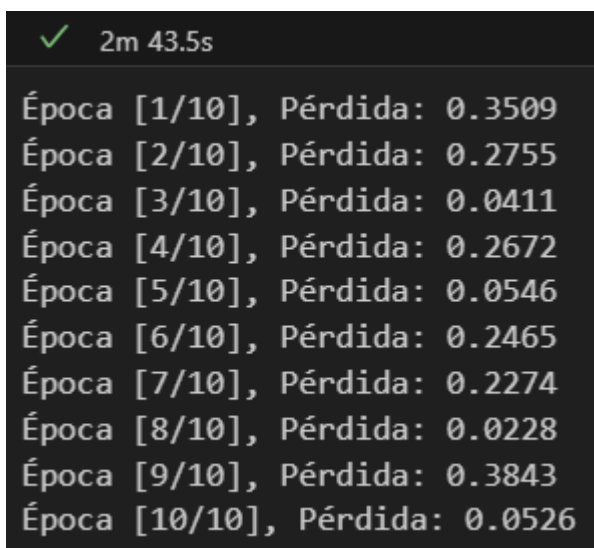
Tiene como parámetros de la arquitectura son los siguiente: No se modifican la cantidad de neuronas respecto al ensayo 1 y 2. Se mantienen solo 2 capas ocultas, ya que tenía mejor comportamiento. Se modifica tipo de función de activación utilizada a Leaky Relu en capa 1 y a sigmoide en capa 2.

- Capa de entrada de 28x28 pixeles
- Capa oculta 1 de 110 neuronas con Leaky Relu.
- Capa oculta 2: de 50 neuronas con sigmoide.
- Capa salida de 10 neuronas para las clases del dataset MINIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos: se mantienen valores de ensayo 1 y ensayo 2.

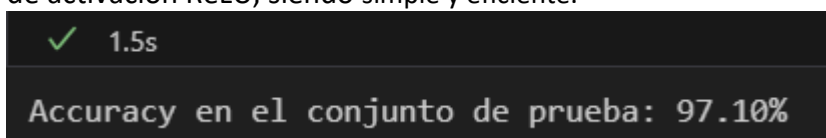
- batch_size = 64 # Tamaño de lote
- learning_rate = 0.001 # Tasa de aprendizaje
- epochs = 10 # Número de épocas de entrenamiento

Se detalla la pérdida de cada una de las 10 épocas, se visualiza que el tiempo vuelve a aumentar, siendo en este caso el mayor de los ensayos. La pérdida es variable, pero se mantienen con rangos altos, por lo que no parece ser una buena opción, solo 3 épocas con valores menores de 0.04, 0.02. y 0.05 (x2).



✓	2m 43.5s
Época [1/10],	Pérdida: 0.3509
Época [2/10],	Pérdida: 0.2755
Época [3/10],	Pérdida: 0.0411
Época [4/10],	Pérdida: 0.2672
Época [5/10],	Pérdida: 0.0546
Época [6/10],	Pérdida: 0.2465
Época [7/10],	Pérdida: 0.2274
Época [8/10],	Pérdida: 0.0228
Época [9/10],	Pérdida: 0.3843
Época [10/10],	Pérdida: 0.0526

De la evaluación tenemos: un **97.10 %** algo que se pudo observar con las pérdidas, hasta ahora se parece al mejor comportamiento luego del ensayo 1 y 4.1 donde se obtuvo un accuracy de 97,65% y 97.26% respectivamente, sin embargo, el tiempo es un factor a considerar para dejarlo o no. Hasta ahora sigue teniendo mejor comportamiento la función de activación ReLU, siendo simple y eficiente.



✓ 1.5s

Accuracy en el conjunto de prueba: 97.10%

Cambiar la función de activación afecta cómo el modelo aprende, es decir, afecta cómo se procesan las relaciones no lineales, lo que puede ser crítico para el desempeño.

2.2 Objetivo 2:

Evaluar distintas estrategias de entrenamiento y seleccionar la mejor alternativa en función de su accuracy.

Metodología 2:

Modificar los siguientes aspectos:

- ✓ Algoritmo de optimización
- ✓ Tasa de aprendizaje (learning rate)
- ✓ Tamaño del Lote (batch size)
- ✓ Número de épocas.

Elabore un informe detallado de los experimentos realizados, incluyendo las configuraciones evaluadas, los resultados obtenidos y las conclusiones que respalden la selección de la estrategia final.

Ensayo 1:

Tomando en cuenta parámetros de ensayos 1 de primera etapa, por tener mejor accuracy.

- Capa de entrada de 28x28 píxeles
- Capa oculta 1 de 200 neuronas con ReLU.
- Capa oculta 2: de 256 neuronas con ReLU.
- Capa salida de 10 neuronas para las clases del dataset MNIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos:

- batch_size = 64 # Tamaño de lote
- learning_rate = 0.001 # Tasa de aprendizaje
- epochs = 10 # Número de épocas de entrenamiento
- Modificamos Algoritmo de optimización a Optimizador AdamW.

Se detalla la pérdida de cada una de las 10 épocas, se visualiza que es variable, al principio parte con un grupo de valores altos, luego un grupo de valores bajos y vuelve a aumentar y ser variable. El detalle de valores por grupo, llama la atención, ya que se podría interpretar que mantiene épocas de mejor aprendizaje.

```
✓ 1m 54.2s
Época [1/10], Pérdida: 0.2146
Época [2/10], Pérdida: 0.3953
Época [3/10], Pérdida: 0.1872
Época [4/10], Pérdida: 0.0974
Época [5/10], Pérdida: 0.0521
Época [6/10], Pérdida: 0.0039
Época [7/10], Pérdida: 0.0280
Época [8/10], Pérdida: 0.2480
Época [9/10], Pérdida: 0.0159
Época [10/10], Pérdida: 0.2047
```

De la evaluación tenemos: un **97.47 %** algo que se pudo observar con las pérdidas, hasta ahora cae dentro de los rangos bien evaluados, luego del ensayo 1 y 4.1 donde se obtuvo un accuracy de 97,65% y 97.26% respectivamente, el tiempo es un factor a considerar para dejarlo, ya que es menor que los ensayos anteriores. Nos quedamos optimizador Adam.

Accuracy en el conjunto de prueba: 97.47%

Ensayo 2:

Tomando en cuenta parámetros de ensayos 1 de primera etapa, por tener mejor accuracy.

- Capa de entrada de 28x28 píxeles
- Capa oculta 1 de 200 neuronas con ReLU.
- Capa oculta 2: de 256 neuronas con ReLU.
- Capa salida de 10 neuronas para las clases del dataset MNIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos:

- `batch_size = 64` # Tamaño de lote
- Modificamos `learning_rate` = 0.1 # Tasa de aprendizaje cambiamos de 0.001 a 0.1
- `epochs = 10` # Número de épocas de entrenamiento
- Algoritmo de optimización= Adam.

Se detalla la pérdida de cada una de las 10 épocas, se visualiza que aumenta el tamaño, y aumenta considerablemente la pérdida en todas las épocas, siendo un mayor error. Se descarta esta opción.

```
✓ 2m 46.5s
Época [1/10], Pérdida: 2.3256
Época [2/10], Pérdida: 2.2842
Época [3/10], Pérdida: 2.3388
Época [4/10], Pérdida: 2.2564
Época [5/10], Pérdida: 2.2842
Época [6/10], Pérdida: 2.3107
Época [7/10], Pérdida: 2.3115
Época [8/10], Pérdida: 2.2662
Época [9/10], Pérdida: 2.3028
Época [10/10], Pérdida: 2.3267
```

De la evaluación tenemos: un **10.28 %** algo que se pudo observar con las pérdidas, hasta ahora cae dentro de los peores evaluados, se descarta definitivamente.

```
Accuracy en el conjunto de prueba: 10.28%
```

Esta tasa determina cuánto cambian los pesos del modelo en respuesta al gradiente. Una tasa alta, actualiza los pesos rápidamente, puede causar oscilaciones y evitar que el modelo alcance un mínimo óptimo, como lo fue en este caso, por otro lado, tasas pequeñas permiten actualizaciones más precisas y estables. En este caso nos quedamos con 0.001.

Ensayo 3.1:

Tomando en cuenta parámetros de ensayos 1 de primera etapa, por tener mejor accuracy.

- Capa de entrada de 28x28 píxeles
- Capa oculta 1 de 200 neuronas con ReLU.
- Capa oculta 2: de 256 neuronas con ReLU.
- Capa salida de 10 neuronas para las clases del dataset MNIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos:

- Modificamos batch_size = 32 # Tamaño de lote de 64 a 32
- Learning_rate = 0.001 # Tasa de aprendizaje
- epochs = 10 # Número de épocas de entrenamiento
- Algoritmo de optimización= Adam.

Se detalla la pérdida de cada una de las 10 épocas, se visualiza que el tiempo aumento un poco, sin embargo, los valores de pérdida mejoraron respecto al ensayo anterior.

```
✓ 2m 57.9s
Época [1/10], Pérdida: 0.0512
Época [2/10], Pérdida: 0.2160
Época [3/10], Pérdida: 0.5349
Época [4/10], Pérdida: 0.1454
Época [5/10], Pérdida: 0.1387
Época [6/10], Pérdida: 0.1178
Época [7/10], Pérdida: 0.0965
Época [8/10], Pérdida: 0.1114
Época [9/10], Pérdida: 0.1318
Época [10/10], Pérdida: 0.0498
```

De la evaluación tenemos: un **97.50 %** hasta ahora cae dentro de los rangos bien evaluados, luego del ensayo 1 y 4.1 donde se obtuvo un accuracy de 97,65% y 97.26% respectivamente, el tiempo es un factor a considerar para dejarlo o no, ya que presenta un alza.

```
Accuracy en el conjunto de prueba: 97.50%
```

Ensayo 3.2: se repite ensayo para seguir analizando comportamiento de batch size.

Tomando encuesta parámetros de ensayos 1 de primera etapa, por tener mejor accuracy.

- Capa de entrada de 28x28 pixeles
- Capa oculta 1 de 200 neuronas con ReLU.
- Capa oculta 2: de 256 neuronas con ReLU.
- Capa salida de 10 neuronas para las clases del dataset MINIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos:

- Modificamos batch_size = 55 # Tamaño de lote de 64 a 32 a 55
- Learning_rate = 0.001 # Tasa de aprendizaje
- epochs = 10 # Número de épocas de entrenamiento
- Algoritmo de optimización= Adam.

Se detalla la pérdida de cada una de las 10 épocas, se visualiza que el tiempo se mantiene relativamente. Los valores de pérdida mejoraron respecto al ensayo anterior.

✓	2m 34.5s
Época [1/10],	Pérdida: 0.3796
Época [2/10],	Pérdida: 0.0752
Época [3/10],	Pérdida: 0.0502
Época [4/10],	Pérdida: 0.0767
Época [5/10],	Pérdida: 0.1443
Época [6/10],	Pérdida: 0.1749
Época [7/10],	Pérdida: 0.2537
Época [8/10],	Pérdida: 0.1381
Época [9/10],	Pérdida: 0.1818
Época [10/10],	Pérdida: 0.3477

De la evaluación tenemos: un **97.49 %** muy cercano al ensayo anterior, cayendo dentro de los rangos bien evaluados, luego del ensayo 1 y 4.1 donde se obtuvo un accuracy de 97,65% y 97.26% respectivamente, el tiempo es un factor a considerar para dejarlo o no, ya que presenta un alza.

Accuracy en el conjunto de prueba: 97.49%

En este caso, se valora mas la cifra de accuracy, respecto al tiempo. Esto se debe evaluar proyecto a proyecto.

Ensayo 3.3: se repite ensayo para seguir analizando comportamiento de batch size.

Tomando encuesta parámetros de ensayos 1 de primera etapa, por tener mejor accuracy.

- Capa de entrada de 28x28 pixeles
- Capa oculta 1 de 200 neuronas con ReLU.
- Capa oculta 2: de 256 neuronas con ReLU.
- Capa salida de 10 neuronas para las clases del dataset MINIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos:

- Modificamos batch_size = 100 # Tamaño de lote de 64 a 32 a 55 a 100
- Learning_rate = 0.001 # Tasa de aprendizaje
- epochs = 10 # Número de épocas de entrenamiento
- Algoritmo de optimización= Adam.

Se detalla la pérdida de cada una de las 10 épocas, se visualiza que el tiempo se mantiene relativamente. Los valores de pérdida mejoraron respecto al ensayo anterior.

```
✓ 2m 13.9s
Época [1/10], Pérdida: 0.1612
Época [2/10], Pérdida: 0.2013
Época [3/10], Pérdida: 0.1893
Época [4/10], Pérdida: 0.2028
Época [5/10], Pérdida: 0.0927
Época [6/10], Pérdida: 0.1708
Época [7/10], Pérdida: 0.1781
Época [8/10], Pérdida: 0.0637
Época [9/10], Pérdida: 0.0910
Época [10/10], Pérdida: 0.0580
```

De la evaluación tenemos: un **97.76 %** aumentando respecto al basal. Superando a los ensayos 1 y 4.1 donde se obtuvo un accuracy de 97,65% y 97.26% respectivamente. Computador se esfuerza mas en obtener el dato, por ende, hay un mayor consumo, lo cual le resta como mejor opción.

```
✓ 1.7s
Accuracy en el conjunto de prueba: 97.76%
```

Ensayo 4.1:

Tomando en cuenta parámetros de ensayos 1 de primera etapa, por tener mejor accuracy.

- Capa de entrada de 28x28 pixeles
- Capa oculta 1 de 200 neuronas con ReLU.
- Capa oculta 2: de 256 neuronas con ReLU.
- Capa salida de 10 neuronas para las clases del dataset MINIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos:

- Batch_size = 100 # Tamaño de lote de 32 mantenemos en 100 para realizar pruebas de consumo.
- Learning_rate = 0.001 # Tasa de aprendizaje
- Modificamos epochs = 15 # Número de épocas de entrenamiento de 10 a 15
- Algoritmo de optimización= Adam.

Se detalla la pérdida de cada una de las 15 épocas con un batch size de 100, se visualiza que leve aumento del tiempo. Los valores de pérdida se ven bien,

```
✓ 2m 46.0s
Época [1/15], Pérdida: 0.1924
Época [2/15], Pérdida: 0.1242
Época [3/15], Pérdida: 0.0946
Época [4/15], Pérdida: 0.1296
Época [5/15], Pérdida: 0.1313
Época [6/15], Pérdida: 0.1224
Época [7/15], Pérdida: 0.1427
Época [8/15], Pérdida: 0.1345
Época [9/15], Pérdida: 0.1118
Época [10/15], Pérdida: 0.1986
Época [11/15], Pérdida: 0.0802
Época [12/15], Pérdida: 0.0469
Época [13/15], Pérdida: 0.1244
Época [14/15], Pérdida: 0.1656
Época [15/15], Pérdida: 0.0984
```

De la evaluación tenemos: un **97.82 %** siendo el mejor resultado hasta el momento, aumentando respecto al basal y al ensayo anterior. Superando a los ensayos 1 y 4.1 donde se obtuvo un accuracy de 97,65% y 97.26% respectivamente. Sin embargo, se repite condición anterior, de que computador se esfuerza más en obtener el dato, por ende, hay un mayor consumo, lo cual le resta como mejor opción.

```
✓ 1.6s
Accuracy en el conjunto de prueba: 97.82%
```

Ensayo 4.2: se repite ensayo para seguir analizando comportamiento de épocas.

Tomando encuesta parámetros de ensayos 1 de primera etapa, por tener mejor accuracy.

- Capa de entrada de 28x28 pixeles
- Capa oculta 1 de 200 neuronas con ReLU.
- Capa oculta 2: de 256 neuronas con ReLU.
- Capa salida de 10 neuronas para las clases del dataset MINIST.
- Dropout de 0.2 para evitar sobreajuste.

Del entrenamiento tenemos:

- Batch size = 32 # Tamaño de lote volvemos a 32
- Learning rate = 0.001 # Tasa de aprendizaje
- Modificamos epochs = 6 # Número de épocas de entrenamiento de 10 a 15 a 6
- Algoritmo de optimización= Adam.

Se detalla la pérdida de cada una de las 6 épocas con un batch size de 32, se visualiza disminución del tiempo. Los valores de pérdida se ven oscilantes.

```
✓ 1m 59.9s
Época [1/6], Pérdida: 0.1348
Época [2/6], Pérdida: 0.5432
Época [3/6], Pérdida: 0.2093
Época [4/6], Pérdida: 0.2145
Época [5/6], Pérdida: 0.0954
Época [6/6], Pérdida: 0.1779
```

De la evaluación tenemos: un **96.96 %** bajando valor de los ensayos. El computador parece no esforzarse tanto al bajar el batch size, se da una comparación entre accuracy y la eficiente, en ambos pierde 1 punto o gana 1 punto y es algo a evaluar en cada proyecto. En ese caso priorizamos el accuracy del ensayo anterior.

Accuracy en el conjunto de prueba: 96.96%

Conclusión perceptrón multicapa: La mejor configuración fue el ensayo 1, 2 capas ocultas con n° de neuronas 200/256/10, función de activación ReLU y accuracy de 97.67%, presenta equilibrio entre precisión y eficiencia. Reducir la cantidad de neuronas o agregar una tercera capa, disminuyó el desempeño del modelo a los valores mas bajos del ensayo. Modificar funciones de activación a Leaky ReLU tuvo buen rendimiento pero en un mayor tiempo, mientras que la función sigmoide tiene menos capacidad de aprendizaje similar a las de ReLU con menos neuronas.

3. Redes convolucionales

3.1 Objetivo 1:

Evaluar diversas arquitecturas para el perceptrón multicapa y seleccionar el modelo que obtenga el mejor rendimiento en términos de accuracy.

Metodología 1:

Modificar los siguientes elementos:

- ✓ Número de filtros.
- ✓ Número de neuronas en la capa posterior a las capas convolucionales.
- ✓ Agregando más capas lineales.
- ✓ Dropout.

Documente los resultados obtenidos, seleccione una arquitectura y entregue sus conclusiones.

Ensayo 1.1:

Del modelo tenemos:

- filters l1=8, filters l2=32
- dropout=0.2
- final_layer_size=128

El final test Accuracy es de [0.9917](#), se detalla la pérdida de cada una de las 10 épocas con filtro (8 y 32), se visualiza que tomó bastante tiempo. Los valores de pérdida se ven buenos y el test accuracy también.

```
✓ 3m 56.3s
Epoch [1/10], Loss: 0.3647, Test Accuracy: 0.9758
Epoch [2/10], Loss: 0.0898, Test Accuracy: 0.9840
Epoch [3/10], Loss: 0.0644, Test Accuracy: 0.9857
Epoch [4/10], Loss: 0.0504, Test Accuracy: 0.9874
Epoch [5/10], Loss: 0.0447, Test Accuracy: 0.9862
Epoch [6/10], Loss: 0.0363, Test Accuracy: 0.9892
Epoch [7/10], Loss: 0.0306, Test Accuracy: 0.9889
Epoch [8/10], Loss: 0.0280, Test Accuracy: 0.9899
Epoch [9/10], Loss: 0.0246, Test Accuracy: 0.9906
Epoch [10/10], Loss: 0.0207, Test Accuracy: 0.9917
Final Test Accuracy: 0.9917
```

Ensayo 1.2: repetimos para ver comportamiento de filtros.

Del modelo tenemos:

- Modificamos filters l1=16, filters l2=64 # se cambia de 8 y 32 a 16 y 64
- dropout=0.2
- final_layer_size=128

El final test Accuracy es de [0.9918](#), se detalla la pérdida de cada una de las 10 épocas con filtro (16 y 64) se visualiza que tomó casi el doble de tiempo de entrenamiento. Los valores de pérdida se ve que parte alto y luego mejora y el test accuracy se ve bueno. Considerando que el final test accuracy mejoró, pero no varió mucho nos quedamos con los primeros filtros, priorizando eficiencia del modelo.

✓ 5m 2.7s

```
Epoch [1/10], Loss: 0.2828, Test Accuracy: 0.9805
Epoch [2/10], Loss: 0.0748, Test Accuracy: 0.9860
Epoch [3/10], Loss: 0.0538, Test Accuracy: 0.9880
Epoch [4/10], Loss: 0.0433, Test Accuracy: 0.9881
Epoch [5/10], Loss: 0.0357, Test Accuracy: 0.9892
Epoch [6/10], Loss: 0.0285, Test Accuracy: 0.9916
Epoch [7/10], Loss: 0.0237, Test Accuracy: 0.9916
Epoch [8/10], Loss: 0.0222, Test Accuracy: 0.9921
Epoch [9/10], Loss: 0.0190, Test Accuracy: 0.9916
Epoch [10/10], Loss: 0.0169, Test Accuracy: 0.9918
Final Test Accuracy: 0.9918
```

Ensayo 2.1:

Del modelo tenemos:

- filters_l1=8, filters_l2=32
- dropout=0.2
- Se modifica final layer size= de 128 a 200 #Aumenta el N° de neuronas en la capa posterior a las capas convolucionales.

El final test Accuracy es de [0.9919](#), se detalla la pérdida de cada una de las 10 épocas con filtro (8 y 32) se visualiza que tiempo disminuyo casi una cifra respecto al anterior. Los valores de pérdida se ve que parte alto y luego mejora igual que anterior y el test accuracy se ve bueno . Consideran que el final test accuracy no tiene un gran cambio.

✓ 4m 20.9s

```
Epoch [1/10], Loss: 0.2970, Test Accuracy: 0.9792
Epoch [2/10], Loss: 0.0757, Test Accuracy: 0.9845
Epoch [3/10], Loss: 0.0545, Test Accuracy: 0.9890
Epoch [4/10], Loss: 0.0425, Test Accuracy: 0.9886
Epoch [5/10], Loss: 0.0343, Test Accuracy: 0.9890
Epoch [6/10], Loss: 0.0278, Test Accuracy: 0.9898
Epoch [7/10], Loss: 0.0241, Test Accuracy: 0.9903
Epoch [8/10], Loss: 0.0212, Test Accuracy: 0.9898
Epoch [9/10], Loss: 0.0177, Test Accuracy: 0.9920
Epoch [10/10], Loss: 0.0162, Test Accuracy: 0.9919
Final Test Accuracy: 0.9919
```

Ensayo 2.2: Se repite ensayo para ver comportamiento de neuronas en capa posterior.

Del modelo tenemos:

- filters_l1=8, filters_l2=32
- dropout=0.2
- Se modifica final_layer_size= de 128 a 200 /200 a 60 #Disminuye el N° de neuronas en la capa posterior a las capas convolucionales.

El final test Accuracy es de [0.9902](#), se detalla la pérdida de cada una de las 10 épocas con filtro (8 y 32) se visualiza que tiempo disminuyo casi una cifra respecto al anterior. Los valores de pérdida se ven oscilantes y el test accuracy tiene solo un valor cercano a 0.99. Al final ambos presentan buena capacidad para procesar características aprendidas.

```
✓ 4m 20.5s
```

Epoch [1/10],	Loss: 0.3875,	Test Accuracy: 0.9711
Epoch [2/10],	Loss: 0.1126,	Test Accuracy: 0.9808
Epoch [3/10],	Loss: 0.0824,	Test Accuracy: 0.9838
Epoch [4/10],	Loss: 0.0688,	Test Accuracy: 0.9862
Epoch [5/10],	Loss: 0.0570,	Test Accuracy: 0.9883
Epoch [6/10],	Loss: 0.0497,	Test Accuracy: 0.9873
Epoch [7/10],	Loss: 0.0434,	Test Accuracy: 0.9889
Epoch [8/10],	Loss: 0.0390,	Test Accuracy: 0.9892
Epoch [9/10],	Loss: 0.0358,	Test Accuracy: 0.9890
Epoch [10/10],	Loss: 0.0313,	Test Accuracy: 0.9902
Final Test Accuracy:	0.9902	

Ensayo 3:

Del modelo tenemos:

- filters_l1=8, filters_l2=32
- dropout=0.2
- Final_layer_size= 128 #Mantenemos N° de neuronas iniciales en la capa posterior a las capas convolucionales.
- Modificamos N° de capas lineales= Se agrega una nueva capa lineal (fc2) y una capa de salida (fc3).

El final test Accuracy es de [0.9899](#), siendo el peor hasta el momento, se detalla la pérdida de cada una de las 10 épocas, se visualiza que tiempo no varía respecto a los ensayos anteriores. Los valores de pérdida se ven que parte bastante alto y luego mejora a partir de la época 2, igual que anterior y el test accuracy se ve que baja. Por lo que no se recomienda para este caso aumentar las capas lineales.


```
✓ 4m 23.0s
Epoch [1/10], Loss: 0.4184, Test Accuracy: 0.9763
Epoch [2/10], Loss: 0.0954, Test Accuracy: 0.9811
Epoch [3/10], Loss: 0.0678, Test Accuracy: 0.9853
Epoch [4/10], Loss: 0.0552, Test Accuracy: 0.9881
Epoch [5/10], Loss: 0.0441, Test Accuracy: 0.9842
Epoch [6/10], Loss: 0.0383, Test Accuracy: 0.9864
Epoch [7/10], Loss: 0.0311, Test Accuracy: 0.9892
Epoch [8/10], Loss: 0.0270, Test Accuracy: 0.9905
Epoch [9/10], Loss: 0.0225, Test Accuracy: 0.9902
Epoch [10/10], Loss: 0.0209, Test Accuracy: 0.9899
Final Test Accuracy: 0.9899
```

Ensayo 3.1:

Del modelo tenemos:

- filters_l1=8, filters_l2=32
- dropout=0.2
- Final_layer_size= 128 #Mantenemos N° de neuronas iniciales en la capa posterior a las capas convolucionales.
- N° de capas lineales=1, Mantenemos una capa lineal (fc1) y una capa de salida (fc2).

El final test Accuracy es de **0.9924**, siendo el mejor hasta ahora, se detalla la pérdida de cada una de las 10 épocas, se visualiza que tiempo no varía respecto a los ensayos anteriores. Los valores de pérdida se ven que parte bastante alto y luego mejora a partir de la época 2, igual que anterior. Al evaluarse con ensayo 1, hay que priorizar tiempo vs accuracy, en este caso, priorizamos tiempo, por lo que se recomienda mantener solo parámetros iniciales de solo 1 clase lineal, al ser un entrenamiento es más rápido y tiene menos probabilidades de sobreajuste.

✓ 4m 2.3s

```
Epoch [1/10], Loss: 0.3065, Test Accuracy: 0.9761
Epoch [2/10], Loss: 0.0852, Test Accuracy: 0.9842
Epoch [3/10], Loss: 0.0602, Test Accuracy: 0.9871
Epoch [4/10], Loss: 0.0485, Test Accuracy: 0.9885
Epoch [5/10], Loss: 0.0395, Test Accuracy: 0.9899
Epoch [6/10], Loss: 0.0327, Test Accuracy: 0.9893
Epoch [7/10], Loss: 0.0281, Test Accuracy: 0.9904
Epoch [8/10], Loss: 0.0263, Test Accuracy: 0.9885
Epoch [9/10], Loss: 0.0204, Test Accuracy: 0.9922
Epoch [10/10], Loss: 0.0189, Test Accuracy: 0.9924
Final Test Accuracy: 0.9924
```

Ensayo 4.1:

Del modelo tenemos:

- filters_l1=8, filters_l2=32
- modificando dropout=**0.5** #se modifica de 0.2 a 0.5
- Final_layer_size= 128 #Número de neuronas en la capa posterior a las capas convolucionales.
- Mantenemos N° de capas lineales original.

El final test Accuracy es de **0.9895**, disminuyendo respecto al anterior, se detalla la pérdida de cada una de las 10 épocas con filtro, se visualiza que tiempo no varía mucho respecto a los ensayos anteriores. Los valores de pérdida se ven que parte alto y luego mejora a partir de la época 3, igual que anterior. Por lo que no se recomienda aumentar a 0.5, sigue siendo mejor con 0.2

```

✓ 3m 53.0s

Epoch [1/10], Loss: 0.4413, Test Accuracy: 0.9731
Epoch [2/10], Loss: 0.1336, Test Accuracy: 0.9818
Epoch [3/10], Loss: 0.0995, Test Accuracy: 0.9874
Epoch [4/10], Loss: 0.0851, Test Accuracy: 0.9885
Epoch [5/10], Loss: 0.0726, Test Accuracy: 0.9878
Epoch [6/10], Loss: 0.0656, Test Accuracy: 0.9874
Epoch [7/10], Loss: 0.0598, Test Accuracy: 0.9894
Epoch [8/10], Loss: 0.0533, Test Accuracy: 0.9893
Epoch [9/10], Loss: 0.0495, Test Accuracy: 0.9912
Epoch [10/10], Loss: 0.0427, Test Accuracy: 0.9895
Final Test Accuracy: 0.9895

```

Ensayo 4.2 : se repite ensayo para evaluar comportamiento de dropout

Del modelo tenemos:

- filters_l1=8, filters_l2=32
- modificando dropout=**0.3** #se modifica de 0.2 a 0.5 / 0.5 a 0.3
- Final_layer_size= 128 #Número de neuronas en la capa posterior a las capas convolucionales.
- Mantenemos N° de capas lineales original.

El final test Accuracy es de **0.9917**, aumentando levemente, sin embargo, es similar al 0.2. Se detalla la pérdida de cada una de las 10 épocas, se visualiza que tiempo disminuye un poco, pero no varía mucho respecto a los ensayos anteriores. Los valores de pérdida se ven que parte alto y luego mejora a partir de la segunda época. Se recomienda mantener parámetros originales de 0.2 .

```

✓ 3m 38.9s

Epoch [1/10], Loss: 0.3536, Test Accuracy: 0.9753
Epoch [2/10], Loss: 0.0988, Test Accuracy: 0.9806
Epoch [3/10], Loss: 0.0729, Test Accuracy: 0.9859
Epoch [4/10], Loss: 0.0579, Test Accuracy: 0.9869
Epoch [5/10], Loss: 0.0494, Test Accuracy: 0.9880
Epoch [6/10], Loss: 0.0434, Test Accuracy: 0.9873
Epoch [7/10], Loss: 0.0374, Test Accuracy: 0.9911
Epoch [8/10], Loss: 0.0340, Test Accuracy: 0.9901
Epoch [9/10], Loss: 0.0297, Test Accuracy: 0.9908
Epoch [10/10], Loss: 0.0254, Test Accuracy: 0.9917
Final Test Accuracy: 0.9917

```

Para complementar, el dropout controla el nivel de regularización para evitar el sobreajuste, durante el entrenamiento, desactiva de manera aleatoria un porcentaje de neuronas en una capa. Esto fuerza al modelo a no depender demasiado de ciertas características. Un valor alto reduce el sobreajuste, puede ralentizar el aprendizaje al eliminar muchas neuronas en cada iteración. Un menor valor, hace una menos regularización, lo que puede resultar en sobreajuste si los datos no son suficientemente diversos.

La mejor combinación a modo de conclusión de estructura para es: filtros (8, 32), dropout: 0.2 y capas luego de convolución=128. Teniendo buena capacidad para procesar, sin mayor esfuerzo de computador, al ser un entrenamiento es más rápido y tiene menos probabilidades de sobreajuste. Aumentar N° de filtro o neuronas, no presenta una mejora importante y ambos casos aumenta tiempo, por ende, se descarta. Aumentar dropout empeoró la precisión, la regularización extrema, afecta negativamente.

3.2 Objetivo 2:

Modificaciones del entrenamiento en ítem de Redes convolucionales.

Metodología 2:

- ✓ Compare diferentes algoritmos de optimización para el entrenamiento.
- ✓ Evalúe el comportamiento del entrenamiento para diferentes valores del learning rate.

Concluya comparando el rendimiento, el número de parámetros y tiempo de ejecución de cada una de las arquitecturas.

Ensayo 1: basal.

Usando Optimizador Adam con tasa de aprendizaje 0.001. El final test Accuracy es de **0.9888**, se detalla la pérdida de cada una de las 10 épocas, siendo alta en la primera y luego mejora en cada época. Alcanza un rendimiento aceptable desde la época 1, optimizándose en las últimas épocas. Se observa bien en general.

```
✓ 3m 39.4s
Epoch [1/10], Loss: 0.3848, Test Accuracy: 0.9711
Epoch [2/10], Loss: 0.0975, Test Accuracy: 0.9820
Epoch [3/10], Loss: 0.0682, Test Accuracy: 0.9858
Epoch [4/10], Loss: 0.0562, Test Accuracy: 0.9866
Epoch [5/10], Loss: 0.0454, Test Accuracy: 0.9878
Epoch [6/10], Loss: 0.0391, Test Accuracy: 0.9851
Epoch [7/10], Loss: 0.0344, Test Accuracy: 0.9892
Epoch [8/10], Loss: 0.0296, Test Accuracy: 0.9889
Epoch [9/10], Loss: 0.0273, Test Accuracy: 0.9904
Epoch [10/10], Loss: 0.0256, Test Accuracy: 0.9888
Final Test Accuracy: 0.9888
```

Ensayo 1.1: Cambiando tasa de aprendizaje.

Usando Optimizador Adam con tasa de aprendizaje **0.1** #se cambia tasa de aprendizaje de 0.001 a **0.1**. El final test Accuracy es de **0.1009**, disminuyendo considerablemente. Se detalla la pérdida de cada una de las 10 épocas, siendo muy alta. Alcanza un pobre rendimiento en todas las épocas. Por lo que no se recomienda aumentar a 0.1 los valores de learning rate.

```
✓ 3m 14.4s
Epoch [1/10], Loss: 3.4178, Test Accuracy: 0.0974
Epoch [2/10], Loss: 2.3072, Test Accuracy: 0.1028
Epoch [3/10], Loss: 2.3068, Test Accuracy: 0.1135
Epoch [4/10], Loss: 2.3065, Test Accuracy: 0.1028
Epoch [5/10], Loss: 2.3076, Test Accuracy: 0.0980
Epoch [6/10], Loss: 2.3067, Test Accuracy: 0.1135
Epoch [7/10], Loss: 2.3075, Test Accuracy: 0.1135
Epoch [8/10], Loss: 2.3069, Test Accuracy: 0.0974
Epoch [9/10], Loss: 2.3081, Test Accuracy: 0.1028
Epoch [10/10], Loss: 2.3079, Test Accuracy: 0.1009
Final Test Accuracy: 0.1009
```

Ensayo 1.2:

Usando Optimizador Adam con tasa de aprendizaje **0.01** #se cambia tasa de aprendizaje de 0.001 a **0.01**. El final test Accuracy es de **0.9890**, volviendo a aumentar. Se detalla la pérdida de cada una de las 10 épocas, siendo alta en la primera y luego mejora en cada época. Alcanza un rendimiento aceptable desde la época 1, pero aún se puede optimizar mas. Por

lo que no se recomienda subir a 0.01 los valores de learning rate, sigue siendo mejor el 0.001

```
✓ 3m 36.9s
Epoch [1/10], Loss: 0.1753, Test Accuracy: 0.9841
Epoch [2/10], Loss: 0.0668, Test Accuracy: 0.9859
Epoch [3/10], Loss: 0.0561, Test Accuracy: 0.9860
Epoch [4/10], Loss: 0.0507, Test Accuracy: 0.9849
Epoch [5/10], Loss: 0.0499, Test Accuracy: 0.9833
Epoch [6/10], Loss: 0.0462, Test Accuracy: 0.9876
Epoch [7/10], Loss: 0.0422, Test Accuracy: 0.9868
Epoch [8/10], Loss: 0.0448, Test Accuracy: 0.9872
Epoch [9/10], Loss: 0.0452, Test Accuracy: 0.9862
Epoch [10/10], Loss: 0.0418, Test Accuracy: 0.9890
Final Test Accuracy: 0.9890
```

Ensayo 2.1: cambio de optimizador.

Usando Optimizador **RMSprop** # Optimizador RMSprop con tasa de aprendizaje 0.001.

El final test Accuracy es de **0.9897**, volviendo a aumentar y siendo el mejor en esta etapa.

Se detalla la pérdida de cada una de las 10 épocas, siendo alta en la primera y luego mejora en cada época. Alcanza un rendimiento alto en la mitad de las épocas, desde la tercera.

```
✓ 3m 23.2s
Epoch [1/10], Loss: 0.2821, Test Accuracy: 0.9720
Epoch [2/10], Loss: 0.0837, Test Accuracy: 0.9799
Epoch [3/10], Loss: 0.0596, Test Accuracy: 0.9856
Epoch [4/10], Loss: 0.0470, Test Accuracy: 0.9874
Epoch [5/10], Loss: 0.0382, Test Accuracy: 0.9889
Epoch [6/10], Loss: 0.0310, Test Accuracy: 0.9883
Epoch [7/10], Loss: 0.0251, Test Accuracy: 0.9879
Epoch [8/10], Loss: 0.0233, Test Accuracy: 0.9881
Epoch [9/10], Loss: 0.0193, Test Accuracy: 0.9903
Epoch [10/10], Loss: 0.0171, Test Accuracy: 0.9897
Final Test Accuracy: 0.9897
```

Ensayo 2.2:

Usando Optimizador **adamW** con tasa de aprendizaje 0.001.

El final test Accuracy es de **0.9887**, disminuyendo levemente. Se detalla la pérdida de cada una de las 10 épocas, siendo alta en la primera y luego mejora en cada época. Alcanza un rendimiento alto en la mitad de las épocas. Sin embargo, no tiene grandes mejoras respecto

a los valores iniciales entregados. Se recomienda usarlo, es similar a resultados con optimizador Adam.

✓	3m 24.4s
Epoch [1/10],	Loss: 0.2847, Test Accuracy: 0.9740
Epoch [2/10],	Loss: 0.0757, Test Accuracy: 0.9848
Epoch [3/10],	Loss: 0.0532, Test Accuracy: 0.9875
Epoch [4/10],	Loss: 0.0423, Test Accuracy: 0.9885
Epoch [5/10],	Loss: 0.0352, Test Accuracy: 0.9916
Epoch [6/10],	Loss: 0.0302, Test Accuracy: 0.9905
Epoch [7/10],	Loss: 0.0254, Test Accuracy: 0.9904
Epoch [8/10],	Loss: 0.0219, Test Accuracy: 0.9901
Epoch [9/10],	Loss: 0.0187, Test Accuracy: 0.9904
Epoch [10/10],	Loss: 0.0167, Test Accuracy: 0.9887
Final Test Accuracy:	0.9887

Conclusión perceptrón multicapa: La mejor configuración fue el ensayo 1, 2 capas ocultas con n° de neuronas 200/256/10, función de activación ReLU y accuracy de 97.67%, presenta equilibrio entre precisión y eficiencia. Reducir la cantidad de neuronas o agregar una tercera capa, disminuyó el desempeño del modelo a los valores mas bajos del ensayo. Modificar funciones de activación a Leaky ReLU tuvo buen rendimiento pero en un mayor tiempo, mientras que la función sigmoide tiene menos capacidad de aprendizaje similar a las de ReLU con menos neuronas.

Perceptron multicapa	Ensayo 1	Ensayo 2	Ensayo 3	Ensayo 4.1	Ensayo 4.2	Ensayo 4.3		
✓ Número de neuronas por capa.	200 / 256 / 10	110/50/10	110/50/10	110/50/10	110/50/10	110/50/10		
✓ Cantidad de capas ocultas.	2	2	3	2	2	2		
✓ Tipo de función de activación utilizada.	ReLU	ReLU	ReLU	Leaky Relu.	sigmoide	Leaky Relu en capa 1 y a sigmoide en capa 2.		
Tiempo (minutos segundos)	2m 1.3 s			2m 1.8 s	1m 39.8 s	2m 43.5 s		
Accuracy (%)	97.67	96.83	96.48	97.26	96.80	97.10		
Perceptron multicapa (200)	Ensayo 1	Ensayo 2	Ensayo 3.1	Ensayo 3.2	Ensayo 3.3	Ensayo 4.1	Ensayo 4.2	
✓ Algoritmo de optimización	AdamW	Adam	Adam	Adam	Adam	Adam	Adam	
✓ Tasa de aprendizaje (learning rate)	0.001	0.1	0.001	0.001	0.001	0.001	0.001	
✓ Tamaño del Lote (batch size)	64	64	32	55	100	100	32	
✓ Número de épocas.	10	10	10	10	10	15	6	
Tiempo (min. Seg.)	1m 54.2s	2m 46.5s	2m 58.9s	2m 34.5s	2m 13.9s	2m 46s	1m 59.9s	
Accuracy (%)	97.47	10.28	97.5	97.49	97.76	97.82	96.96	
mayor consumo pc								

Conclusión redes convolucionales: La mejor combinación a modo de conclusión de estructura para es: filtros (8, 32), dropout: 0.2 y capas luego de convolución=128. Teniendo buena capacidad para procesar, sin mayor esfuerzo de computador, al ser un entrenamiento es más rápido y tiene menos probabilidades de sobreajuste. Aumentar N° de filtro o neuronas, no presenta una mejora importante y ambos casos aumenta tiempo, por ende, se descarta. Aumentar dropout empeoró la precisión, la regularización extrema, afecta negativamente. El mejor rendimiento se obtuvo con el optimizador RMSprop con tasa de aprendizaje de 0.001, luego sigue el optimizador ADAM, aunque AdamW no presenta grandes mejoras, igual anda bien. Se evidencia que altas tasas de aprendizaje tienen como resultado un bajo desempeño.

Redes convolucionales	Ensayo 1.1	Ensayo 1.2	Ensayo 2.1	Ensayo 2.2	Ensayo 3	Ensayo 3.1	Ensayo 4.1	Ensayo 4.2					
✓ Número de filtros.	8 y 32	16 y 64	8 y 32	8 y 32	8 y 32	8 y 32	8 y 32	8 y 32					
✓ Número de neuronas en la capa posterior a las capas convolucionales.	128	128	200	60	128	128	128	128					
✓ Agregando más capas lineales.	1	1	1	1	2	1	1	1					
✓ Dropout.	0.2	0.2	0.2	0.2	0.2	0.2	0.5	0.3					
Tiempo (min. Seg.)	3m 56.3s	5m 2.7s	4m 20.9	4m 20.5s	4m 23s	4m 2.3s	3m 53 s	3m 38.9					
Accuracy (%)	0.9917	0.9918	0.9919	0.9902	0.9899	0.9924	0.9895	0.9917					
resentan buena capacidad para procesar característ Al evaluarse con ensayo 1, hay que priorizar tiempo vs accuracy, en este caso, priorizamos tiempo.													
Redes convolucionales	Ensayo 1	Ensayo 1.1	Ensayo 1.2	Ensayo 2.1	Ensayo 2.2								
✓ valores del learning rate.	0.001	0.1	0.01	0.001	0.001								
✓ algoritmos de optimización	Adam	Adam	Adam	RMSprop	AdamW								
Tiempo (min. Seg.)	3m 39.4s	3m 14.4s	3m 36.9s	3m 23.2s	3m 24.4s								
Accuracy (%)	0.9888	0.1009	0.9890	0.9897	0.9887								