Tecnológico de Monterrey

TC3006 - Inteligencia artificial avanzada para la ciencia de datos I

Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo. (Portafolio Análisis)

Profesor: Jorge Adolfo Ramírez Uresti

A01751277 - Alejandro Somarriba Aguirre

8 de septiembre de 2023

Para este reporte se escogió analizar el modelo del clasificador de perceptrón multicapa que se realizó con el framework de Scikit-Learn. El dataset elegido fue el conjunto de datos MNIST que contiene imágenes de los dígitos del 0 a 9 escritos a mano y cuyo fin es precisamente clasificarlos correctamente. Dado que el conjunto de datos se trata de imágenes que corresponden a una de 10 categorías, y que los datos son los valores de intensidad de los 64 pixeles de las imágenes, pareció adecuado utilizar una pequeña red neuronal, es decir, un perceptrón multicapa, para resolver el problema de clasificación. Adicionalmente, el conjunto de datos está bien balanceado, con cada categoría teniendo un promedio de aproximadamente 180 observaciones.

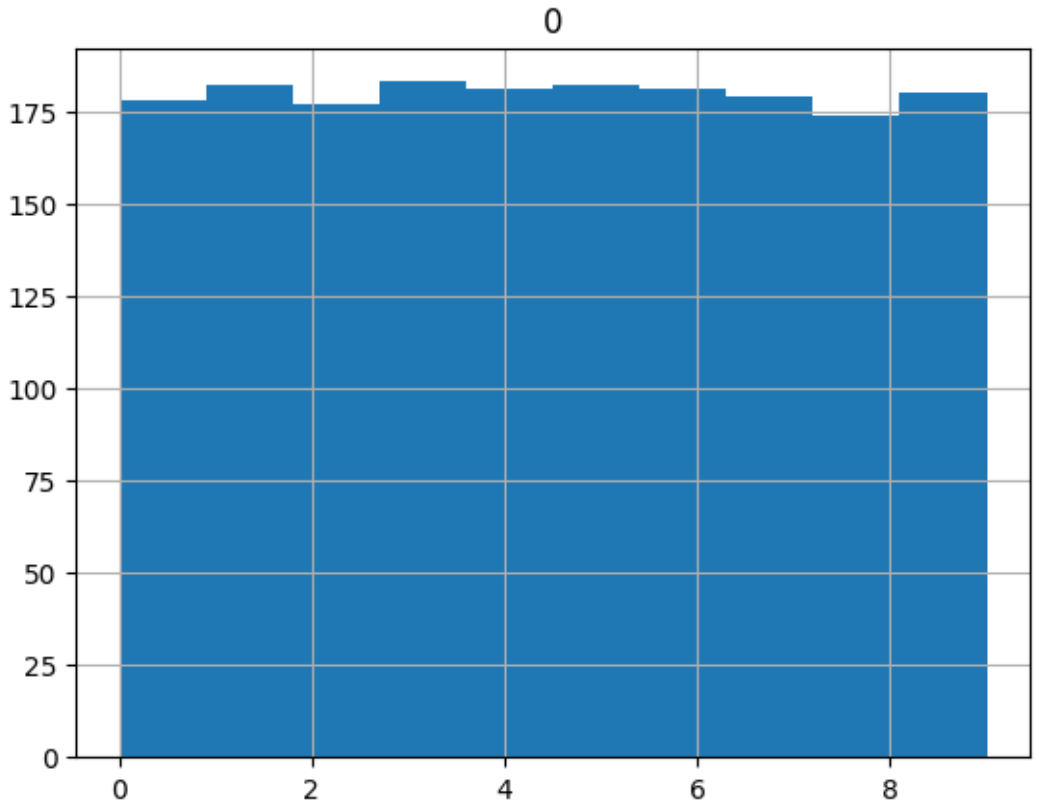
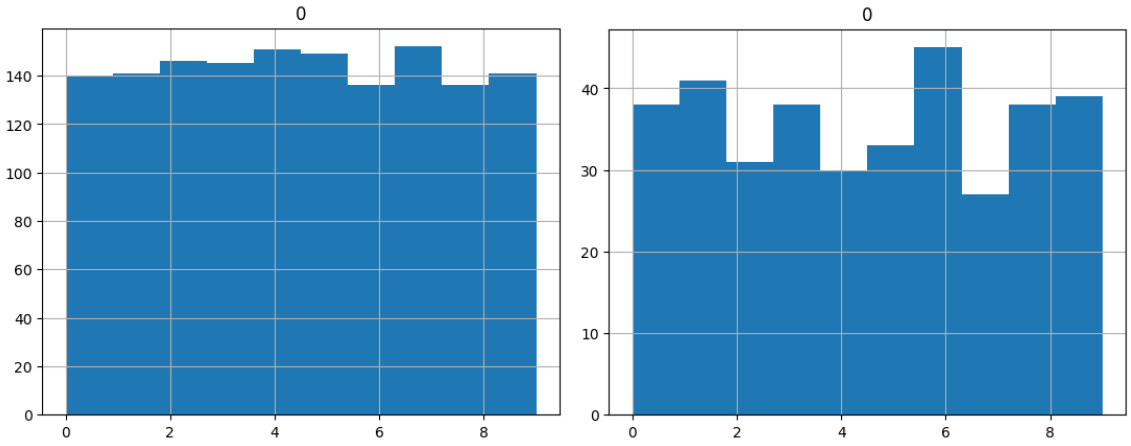


Imagen 1. Distribución de categorías de todos los datos

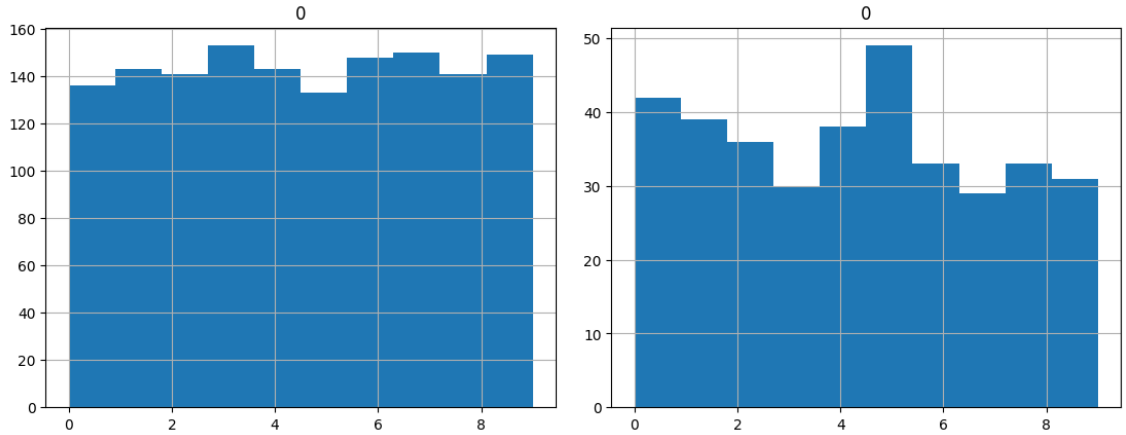
El que tengan una distribución casi uniforme reduce la posibilidad de que exista un sesgo en el modelo, siempre y cuando la aleatoriedad no escoja una categoría demasiado al momento de entrenar.

Separación de los datos

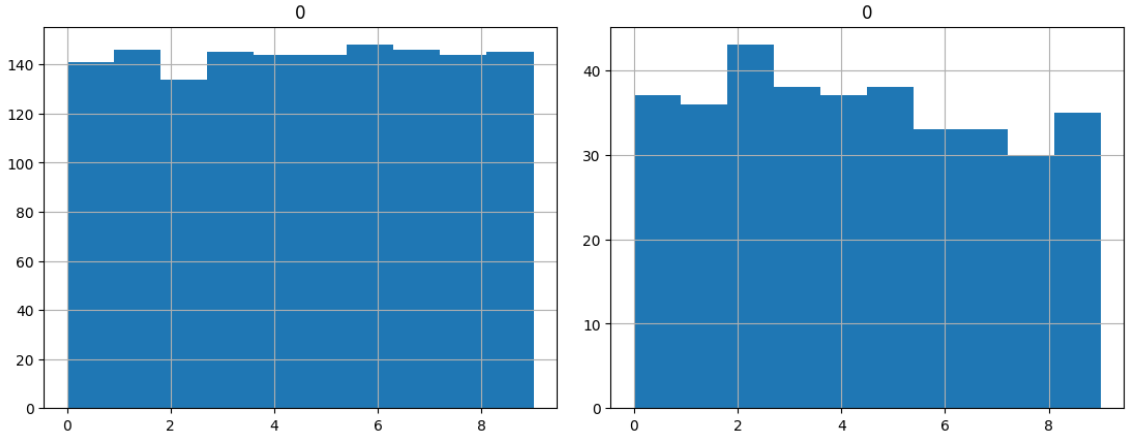
Antes de comenzar a entrenar el modelo, se separan los datos en conjuntos de entrenamiento y prueba utilizando la función de train_test_split() de la librería de sklearn. Se reservó un 80% de los datos para el entrenamiento, y el 20% restante quedó para las pruebas. A continuación, se muestran las distribuciones de los datos de entrenamiento y prueba para diferentes corridas del programa, con el fin de demostrar que la separación de los datos es en general aleatoria y uniforme.



Imágenes 2 y 3: Distribución de categorías para datos de entrenamiento (izquierda) y prueba (derecha) para una corrida del programa



Imágenes 4 y 5: Distribución de categorías para datos de entrenamiento (izquierda) y prueba (derecha) para otra corrida del programa



Imágenes 6 y 7: Distribución de categorías para datos de entrenamiento (izquierda) y prueba (derecha) para otra corrida del programa

Se puede observar que a pesar de que no haya una distribución perfectamente uniforme para los datos de prueba, no hay disparidades extremas en las categorías.

Evidencia de generalización

Una manera de demostrar que generaliza es observando las gráficas de pérdida a lo largo de las iteraciones de entrenamiento para diferentes corridas.

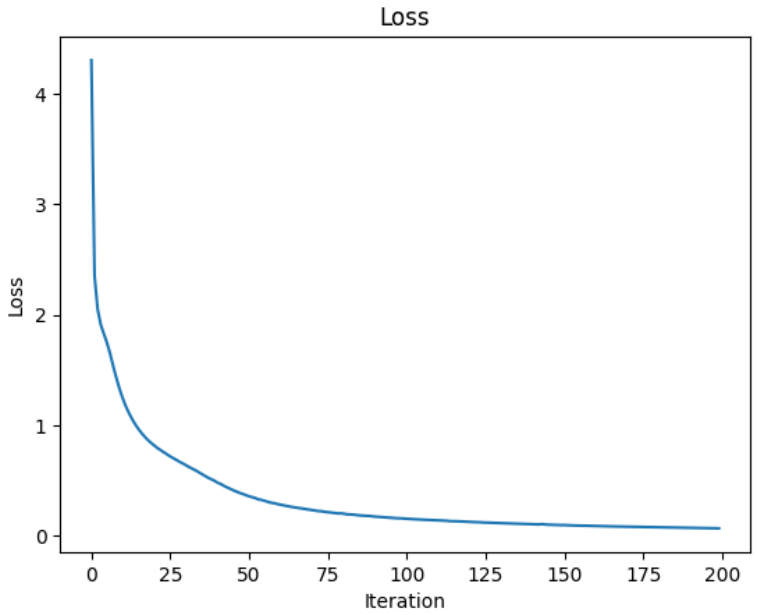


Imagen 8: Pérdida a lo largo de las iteraciones de entrenamiento (pérdida final: 0.06518978)

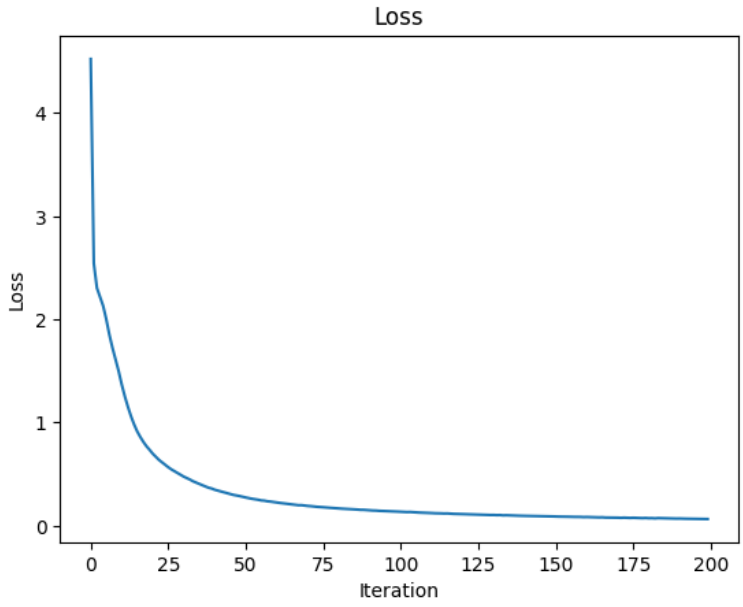


Imagen 9: Pérdida a lo largo de las iteraciones de entrenamiento (pérdida final: 0.06514400)

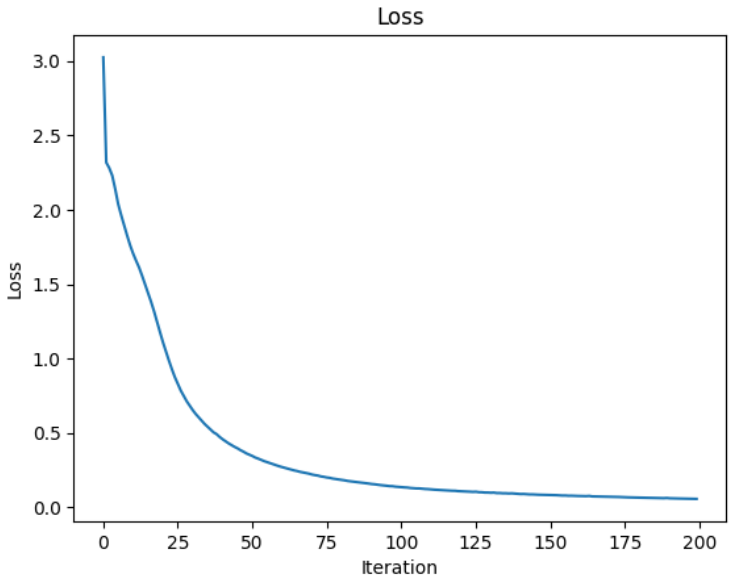


Imagen 10: Pérdida a lo largo de las iteraciones de entrenamiento (pérdida final: 0.05722363)

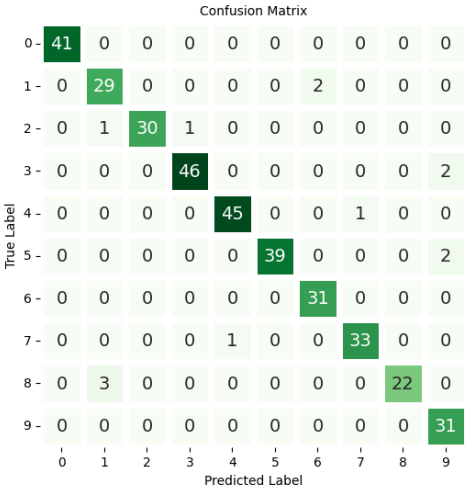
Se puede observar que, para 3 corridas diferentes, la pérdida del modelo se reduce bastante al término de las 200 iteraciones en las que se entrena; esto indica que el modelo es bueno para generalizar y para mantener pocos errores.

Análisis de Sesgo y Varianza

Para analizar el sesgo y la varianza del modelo, se pueden observar las métricas de evaluación obtenidas cuando se aplicó el modelo a los datos de prueba, al igual que la matriz de confusión resultante.

Para una corrida, se obtuvo un valor de accuracy para los datos de entrenamiento de 0.9895615866388309, mientras que para los datos de prueba fue de 0.9638888888888889.

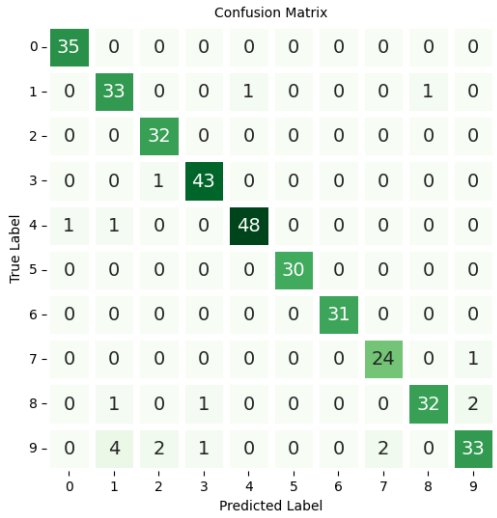
	precision	recall	f1-score	support
0	1.00	1.00	1.00	41
1	0.88	0.94	0.91	31
2	1.00	0.94	0.97	32
3	0.98	0.96	0.97	48
4	0.98	0.98	0.98	46
5	1.00	0.95	0.97	41
6	0.94	1.00	0.97	31
7	0.97	0.97	0.97	34
8	1.00	0.88	0.94	25
9	0.89	1.00	0.94	31
accuracy			0.96	360
macro avg	0.96	0.96	0.96	360
weighted avg	0.97	0.96	0.96	360



Imágenes 11 y 12: Métricas de evaluación (izquierda) y matriz de confusión (derecha) para una corrida

Para otra corrida, se obtuvo un valor de accuracy para los datos de entrenamiento de 0.9812108559498957, mientras que para los datos de prueba fue de 0.9472222222222222.

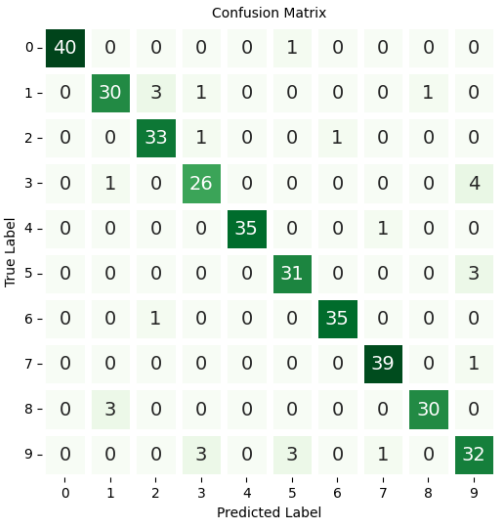
	precision	recall	f1-score	support
0	0.97	1.00	0.99	35
1	0.85	0.94	0.89	35
2	0.91	1.00	0.96	32
3	0.96	0.98	0.97	44
4	0.98	0.96	0.97	50
5	1.00	1.00	1.00	30
6	1.00	1.00	1.00	31
7	0.92	0.96	0.94	25
8	0.97	0.89	0.93	36
9	0.92	0.79	0.85	42
accuracy			0.95	360
macro avg	0.95	0.95	0.95	360
weighted avg	0.95	0.95	0.95	360



Imágenes 13 y 14: Métricas de evaluación (izquierda) y matriz de confusión (derecha) para otra corrida

Para una tercera corrida, se obtuvo un valor de accuracy para los datos de entrenamiento de 0.965205288796103, mientras que para los datos de prueba fue de 0.9194444444444444.

	precision	recall	f1-score	support
0	1.00	0.98	0.99	41
1	0.88	0.86	0.87	35
2	0.89	0.94	0.92	35
3	0.84	0.84	0.84	31
4	1.00	0.97	0.99	36
5	0.89	0.91	0.90	34
6	0.97	0.97	0.97	36
7	0.95	0.97	0.96	40
8	0.97	0.91	0.94	33
9	0.80	0.82	0.81	39
accuracy			0.92	360
macro avg	0.92	0.92	0.92	360
weighted avg	0.92	0.92	0.92	360



Imágenes 15 y 16: Métricas de evaluación (izquierda) y matriz de confusión (derecha) para otra corrida

Se puede observar que para los datos de entrenamiento siempre hay un mejor accuracy que para los datos de prueba; esto podría indicar que existe un alto grado de varianza en el modelo, pero debido a que la diferencia del accuracy entre los conjuntos de entrenamiento y prueba es bastante pequeña, el grado de varianza probablemente es bajo.

Del mismo modo, debido a que los valores de las métricas de los datos de prueba son buenos, se puede decir que el grado de sesgo (bias) es bajo, pues en diferentes corridas se mantiene consistentemente bueno.

A partir de lo anterior, se podría decir que puesto a que el accuracy siempre es mayor para los datos de entrenamiento, podría indicar que existe un grado de overfitting bajo. Sin embargo, como la diferencia con el accuracy de los datos de prueba es bastante pequeña, parece más probable que el overfitting no es un problema significativo. Del mismo modo, con los mismos resultados se puede decir que el grado de underfitting también es bajo, ya que el modelo sí es capaz de generalizar y hay un bajo grado de sesgo.

Regularización y ajuste de parámetros

A continuación, se presentan diferentes ajustes realizados al modelo para observar cambios en su desempeño.

Cambio de optimizador de parámetros

El primer cambio realizado fue utilizar el optimizador "Adam" en lugar de Stochastic Gradient Descent. Los resultados de una corrida se muestran a continuación.

Para una corrida, el valor de accuracy de los datos de entrenamiento fue de 1.0, mientras que para los datos de prueba fue de 0.9611111111111111.

El valor de pérdida al final del entrenamiento fue de 0.00734130 y la gráfica de pérdida es la siguiente:

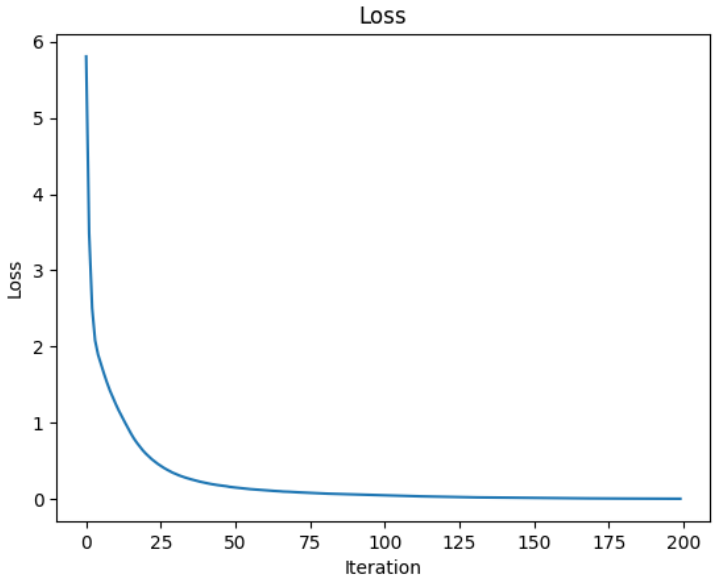
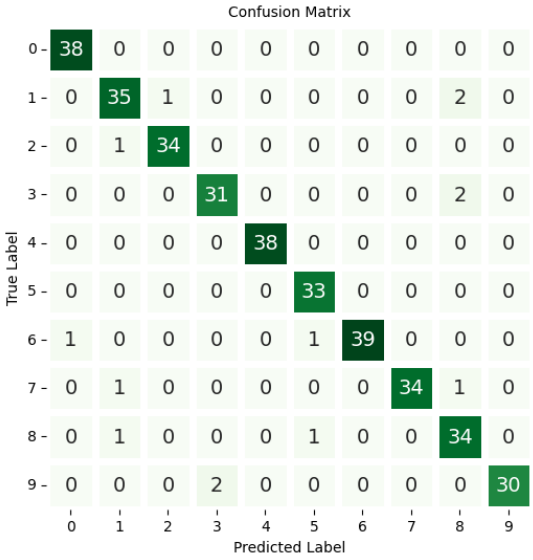


Imagen 17: Pérdida a lo largo de las iteraciones con optimizador Adam

A continuación, se muestran las métricas de evaluación de los datos de prueba, al igual que la matriz de confusión:

	precision	recall	f1-score	support
0	0.97	1.00	0.99	38
1	0.92	0.92	0.92	38
2	0.97	0.97	0.97	35
3	0.94	0.94	0.94	33
4	1.00	1.00	1.00	38
5	0.94	1.00	0.97	33
6	1.00	0.95	0.97	41
7	1.00	0.94	0.97	36
8	0.87	0.94	0.91	36
9	1.00	0.94	0.97	32
accuracy			0.96	360
macro avg	0.96	0.96	0.96	360
weighted avg	0.96	0.96	0.96	360



Imágenes 18 y 19: Métricas de evaluación (izquierda) y matriz de confusión (derecha) para una corrida con optimizador Adam

Ahora se muestran los resultados para otra corrida:

Para esta corrida, el valor de accuracy de los datos de entrenamiento fue de 0.9979123173277662, mientras que para los datos de prueba fue de 0.9555555555555556.

El valor de pérdida al final del entrenamiento fue de 0.01464045 y la gráfica de pérdida es la siguiente:

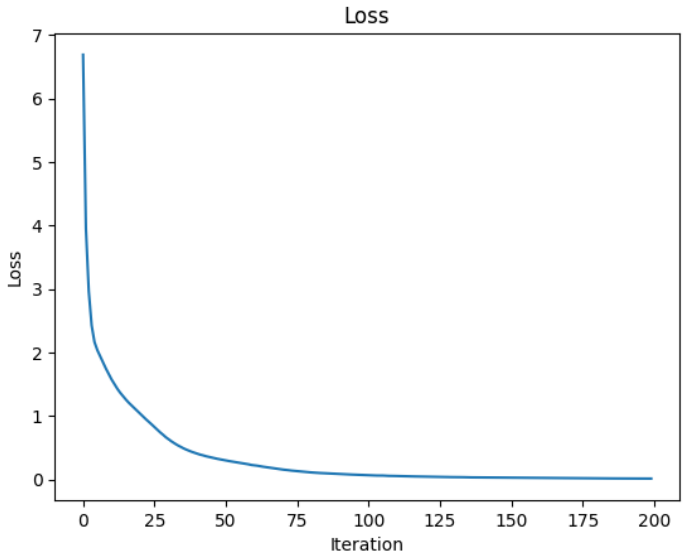
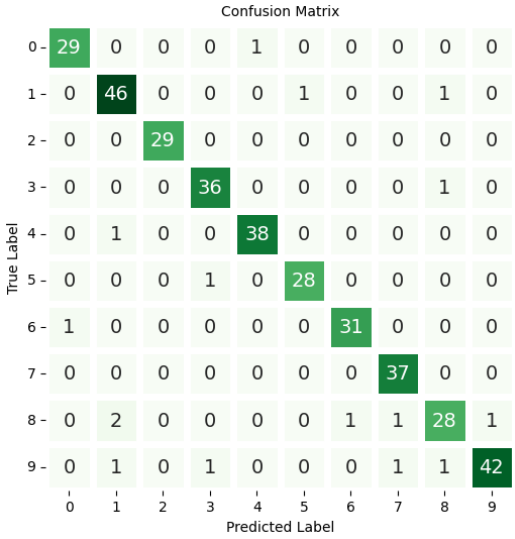


Imagen 20: Pérdida a lo largo de las iteraciones con optimizador Adam

A continuación, se muestran las métricas de evaluación de los datos de prueba, al igual que la matriz de confusión:

	precision	recall	f1-score	support
0	0.97	0.97	0.97	30
1	0.92	0.96	0.94	48
2	1.00	1.00	1.00	29
3	0.95	0.97	0.96	37
4	0.97	0.97	0.97	39
5	0.97	0.97	0.97	29
6	0.97	0.97	0.97	32
7	0.95	1.00	0.97	37
8	0.90	0.85	0.88	33
9	0.98	0.91	0.94	46
accuracy			0.96	360
macro avg	0.96	0.96	0.96	360
weighted avg	0.96	0.96	0.96	360



Imágenes 21 y 22: Métricas de evaluación (izquierda) y matriz de confusión (derecha) para otra corrida con optimizador Adam

Con estas dos corridas, se podría inferir que, al haber cambiado el optimizador de parámetros a Adam, el rendimiento del modelo parece haber mejorado, ya que el valor de accuracy para los datos de entrenamiento y prueba aumentó un poco a comparación de las corridas con el optimizador de Stochastic Gradient Descent. Del mismo modo, el valor de pérdida al final de las iteraciones de entrenamiento también se redujo al utilizar el nuevo optimizador. En general, al haber usado el optimizador Adam se obtuvieron mejores resultados.

Cambio del término de regularización L2

En las corridas originales, el modelo tenía un valor alpha de 0.0001 por default; para experimentar, se hicieron corridas con un valor de 0.01. A continuación se muestran los resultados.

Para la primera corrida, el valor de accuracy de los datos de entrenamiento fue de 0.9387613082811412, mientras que para los datos de prueba fue de 0.8888888888888888.

El valor de pérdida al final del entrenamiento fue de 0.31928464 y la gráfica de pérdida es la siguiente:

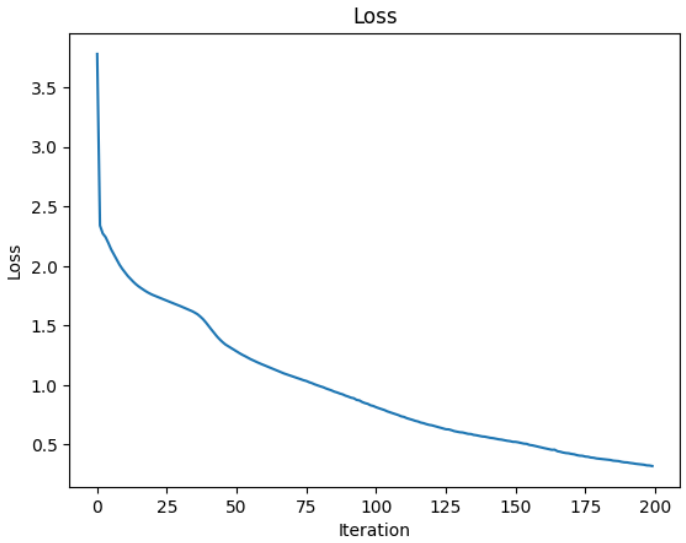
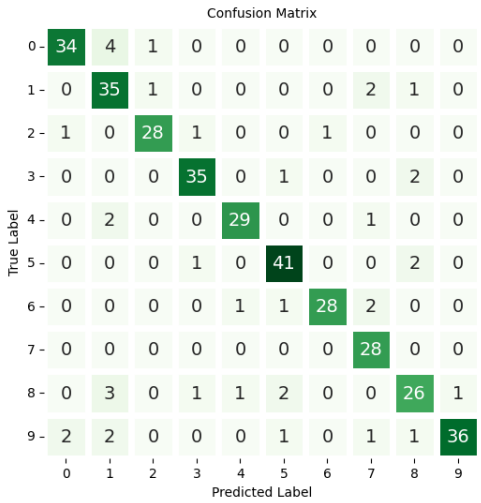


Imagen 23: Pérdida a lo largo de las iteraciones con un alpha de 0.01

A continuación, se muestran las métricas de evaluación de los datos de prueba, al igual que la matriz de confusión:

	precision	recall	f1-score	support
0	0.92	0.87	0.89	39
1	0.76	0.90	0.82	39
2	0.93	0.90	0.92	31
3	0.92	0.92	0.92	38
4	0.94	0.91	0.92	32
5	0.89	0.93	0.91	44
6	0.97	0.88	0.92	32
7	0.82	1.00	0.90	28
8	0.81	0.76	0.79	34
9	0.97	0.84	0.90	43
accuracy			0.89	360
macro avg	0.89	0.89	0.89	360
weighted avg	0.89	0.89	0.89	360



Imágenes 24 y 25: Métricas de evaluación (izquierda) y matriz de confusión (derecha) para una corrida con alpha 0.01

Ahora se muestran los resultados para otra corrida:

Para la primera corrida, el valor de accuracy de los datos de entrenamiento fue de 0.9972164231036882, mientras que para los datos de prueba fue de 0.9583333333333334.

El valor de pérdida al final del entrenamiento fue de 0.03748273 y la gráfica de pérdida es la siguiente:

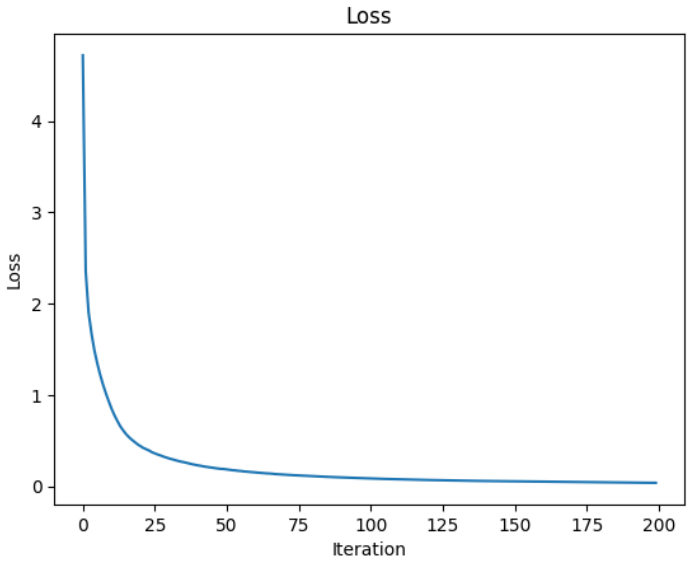
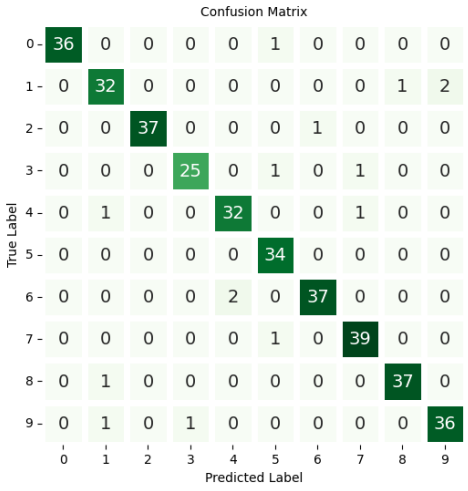


Imagen 26: Pérdida a lo largo de las iteraciones con un alpha de 0.01

A continuación, se muestran las métricas de evaluación de los datos de prueba, al igual que la matriz de confusión:

	precision	recall	f1-score	support
0	1.00	0.97	0.99	37
1	0.91	0.91	0.91	35
2	1.00	0.97	0.99	38
3	0.96	0.93	0.94	27
4	0.94	0.94	0.94	34
5	0.92	1.00	0.96	34
6	0.97	0.95	0.96	39
7	0.95	0.97	0.96	40
8	0.97	0.97	0.97	38
9	0.95	0.95	0.95	38
accuracy			0.96	360
macro avg	0.96	0.96	0.96	360
weighted avg	0.96	0.96	0.96	360



Imágenes 27 y 28: Métricas de evaluación (izquierda) y matriz de confusión (derecha) para otra corrida con alpha 0.01

Comparando estas corridas con las corridas originales, es un poco difícil decir definitivamente si los resultados fueron mejores o peores, ya que en la primera corrida fueron peores, pero en la segunda corrida fueron similares. En todo caso, se podría decir que aumentar el valor de alpha de 0.0001 a 0.01 no resulta en cambios significativos en el rendimiento del modelo.

Cambio de la tasa de aprendizaje inicial

En las corridas originales, el valor de la tasa de aprendizaje era de 0.001. Para experimentar, se cambió a 0.005. A continuación, se muestran los resultados.

Para la primera corrida, el valor de accuracy de los datos de entrenamiento fue de 0.9993041057759221, mientras que para los datos de prueba fue de 0.9666666666666667.

El valor de pérdida al final del entrenamiento fue de 0.00470133 y la gráfica de pérdida es la siguiente:

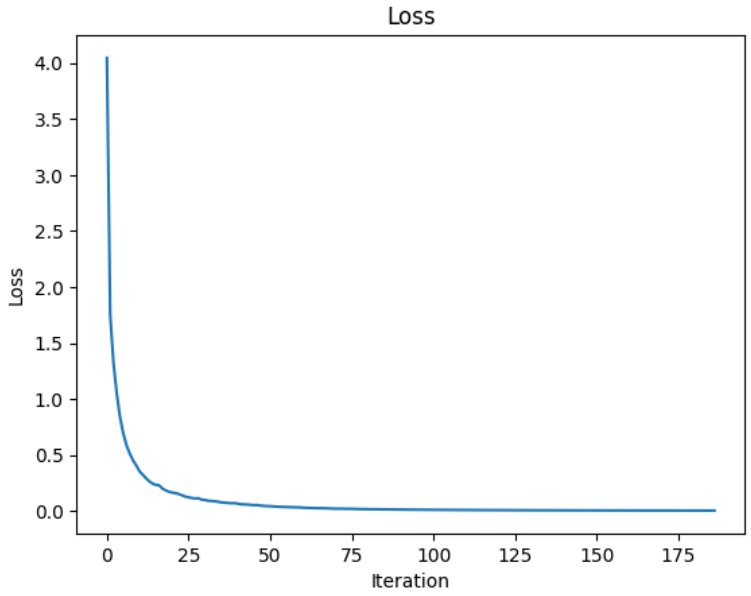
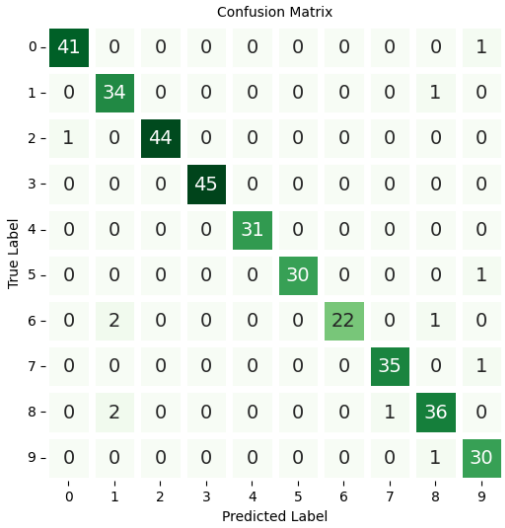


Imagen 29: Pérdida a lo largo de las iteraciones con una tasa de aprendizaje de 0.005

A continuación, se muestran las métricas de evaluación de los datos de prueba, al igual que la matriz de confusión:

	precision	recall	f1-score	support
0	0.98	0.98	0.98	42
1	0.89	0.97	0.93	35
2	1.00	0.98	0.99	45
3	1.00	1.00	1.00	45
4	1.00	1.00	1.00	31
5	1.00	0.97	0.98	31
6	1.00	0.88	0.94	25
7	0.97	0.97	0.97	36
8	0.92	0.92	0.92	39
9	0.91	0.97	0.94	31
accuracy			0.97	360
macro avg	0.97	0.96	0.96	360
weighted avg	0.97	0.97	0.97	360



Imágenes 30 y 31: Métricas de evaluación (izquierda) y matriz de confusión (derecha) para una corrida con una tasa de aprendizaje de 0.005

Ahora se muestran los resultados para otra corrida:

Para la primera corrida, el valor de accuracy de los datos de entrenamiento fue de 0.9965205288796103, mientras que para los datos de prueba fue de 0.9444444444444444.

El valor de pérdida al final del entrenamiento fue de 0.01394778 y la gráfica de pérdida es la siguiente:

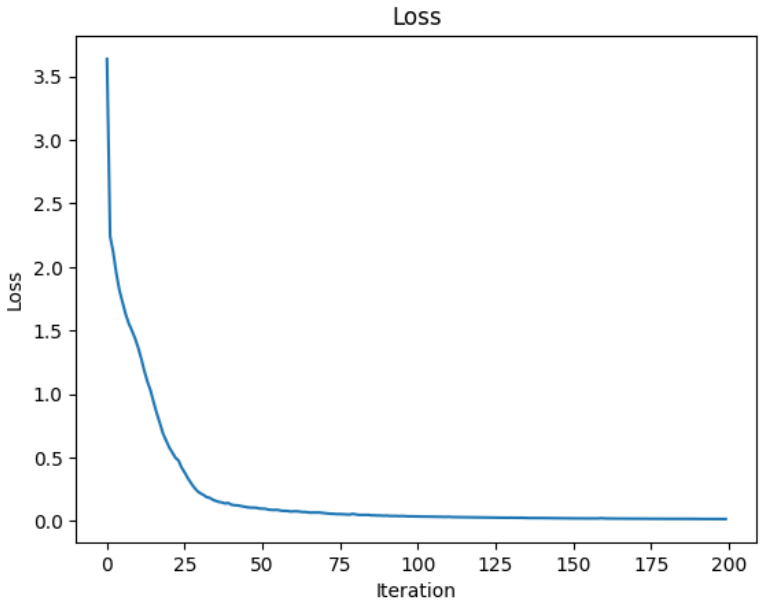
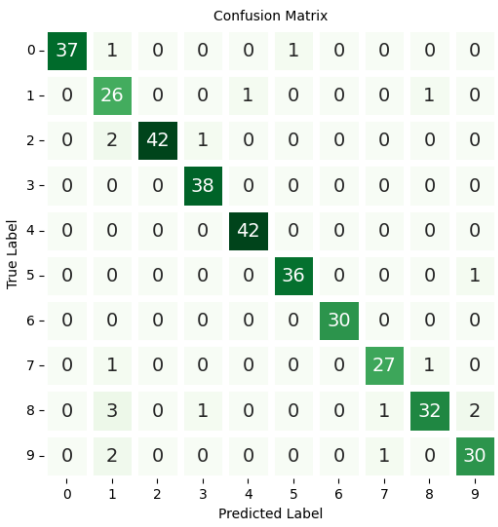


Imagen 32: Pérdida a lo largo de las iteraciones con una tasa de aprendizaje de 0.005

A continuación, se muestran las métricas de evaluación de los datos de prueba, al igual que la matriz de confusión:

	precision	recall	f1-score	support
0	1.00	0.95	0.97	39
1	0.74	0.93	0.83	28
2	1.00	0.93	0.97	45
3	0.95	1.00	0.97	38
4	0.98	1.00	0.99	42
5	0.97	0.97	0.97	37
6	1.00	1.00	1.00	30
7	0.93	0.93	0.93	29
8	0.94	0.82	0.88	39
9	0.91	0.91	0.91	33
accuracy			0.94	360
macro avg	0.94	0.94	0.94	360
weighted avg	0.95	0.94	0.95	360



Imágenes 33 y 34: Métricas de evaluación (izquierda) y matriz de confusión (derecha) para otra corrida con una tasa de aprendizaje de 0.005

Comparando estas dos corridas con las corridas del principio, se podría decir que, al incrementar ligeramente la tasa de aprendizaje, el modelo logró aprender un poco más y obtener métricas mejores. De manera similar al cambio del optimizador a "Adam", es una mejora pequeña. El modelo inicial ya era decentemente bueno para empezar, así que es algo complicado observar mejoras significativas. Sin embargo, parece ser que con cambiar el optimizador de Stochastic Gradient Descent a Adam y aumentar ligeramente la tasa de aprendizaje de 0.001 a 0.005, hay una mejora en los resultados del modelo.