

Clasificación de Fashion-MNIST con Redes Neuronales Feedforward Multicapa en PyTorch

Autores

Matías Eduardo Bordone Carranza mebordone@unc.edu.ar FAMAFyC

Jeanette Peralta jeanette.peralta@mi.unc.edu.ar FAMAFyC

Resumen

Este trabajo presenta la implementación y análisis de una red neuronal feedforward multicapa para la clasificación del dataset Fashion-MNIST utilizando PyTorch. El objetivo principal fue explorar el impacto de diferentes hiperparámetros en el rendimiento del modelo mediante un proceso sistemático de experimentación. Se evaluaron variaciones en la tasa de aprendizaje (0.0001, 0.001, 0.01), optimizadores (SGD y ADAM), valores de dropout (0.0, 0.2, 0.4, 0.6), arquitecturas de red (64-32, 128-64, 256-128, 512-256), número de épocas (5-30) y tamaños de batch (32-256). Los resultados demostraron que el optimizador ADAM supera significativamente a SGD, alcanzando 88.06% de precisión en validación. La configuración óptima final, combinando ADAM con learning rate 0.001, arquitectura 256-128, batch size 32, dropout 0.2 y 30 épocas, logró una precisión de **87.99%** en el conjunto de validación. El análisis reveló que batch sizes pequeños (32) y arquitecturas moderadas (256-128) ofrecen el mejor balance entre rendimiento y eficiencia computacional, mientras que el dropout de 0.2-0.4 proporciona una regularización efectiva sin comprometer la capacidad de aprendizaje del modelo.

Introducción

El objetivo principal de este informe es implementar, entrenar y analizar un clasificador para el dataset Fashion-MNIST utilizando una Red Neuronal Artificial Feedforward Multicapa en PyTorch.

Fashion-MNIST es un dataset estándar que contiene 70.000 imágenes en escala de grises (28×28 píxeles) divididas en 10 categorías de artículos de moda, con 60.000 imágenes para entrenamiento y 10.000 para validación. Los objetivos del trabajo son: (1) explorar el conjunto de datos y establecer una arquitectura de red base, (2) evaluar el impacto de distintos hiperparámetros (tasas de aprendizaje, optimizadores, dropout, tamaños de batch) en el rendimiento del modelo, y (3) seleccionar y entrenar la configuración óptima para obtener la máxima precisión de clasificación.

Metodología

Arquitectura de la red neuronal

Se implementó una red neuronal feedforward (multilayer perceptron) con capas densas (fully connected). La estructura base consiste en:

- **Capa de entrada:** Flatten de la imagen $28 \times 28 \rightarrow$ vector de 784 características
- **Capa oculta 1:** 128 neuronas + ReLU + Dropout(0.2)
- **Capa oculta 2:** 64 neuronas + ReLU + Dropout(0.2)
- **Capa de salida:** 10 neuronas (una por clase) con CrossEntropyLoss

Hiperparámetros evaluados

Se realizó una búsqueda sistemática de hiperparámetros variando: - **Learning rate:** [0.0001, 0.001, 0.01] - **Optimizador:** SGD y ADAM - **Dropout:** [0.0, 0.2, 0.4, 0.6] - **Arquitectura:** (64,32), (128,64), (256,128), (512,256) - **Épocas:** [5, 10, 15, 20, 30] - **Batch size:** [32, 64, 100, 128, 256]

Resultados

Resumen de Experimentos

Se realizaron 6 experimentos sistemáticos variando un hiperparámetro a la vez. La siguiente tabla resume los mejores resultados obtenidos en cada experimento:

Experimento	Mejor Configuración	Validation Accuracy
Learning Rate	LR = 0.01	83.84%
Optimizador	ADAM	88.06%
Dropout	Dropout = 0.2	73.55%
Arquitectura	512-256	75.06%
Épocas	30 épocas	79.63%
Batch Size	32	79.35%

Hallazgos principales: - ADAM supera significativamente a SGD (88.06% vs 73.44%) - Learning rate de 0.001-0.01 muestra buen rendimiento - Batch size pequeño (32) mejora el rendimiento - Arquitecturas más grandes (256-128, 512-256) mejoran la precisión - Dropout de 0.2-0.4 proporciona buena regularización

Modelo Final

Basándose en el análisis de hiperparámetros, se entrenó un modelo final con la siguiente configuración:

- **Optimizador:** ADAM
- **Learning Rate:** 0.001
- **Dropout:** 0.2
- **Arquitectura:** 256-128
- **Batch Size:** 32
- **Épocas:** 30

Resultado Final: - Precisión en Validación: 87.99% - Precisión en Entrenamiento: ~90.22%

El modelo muestra convergencia estable sin overfitting significativo (diferencia train/val ~2%).

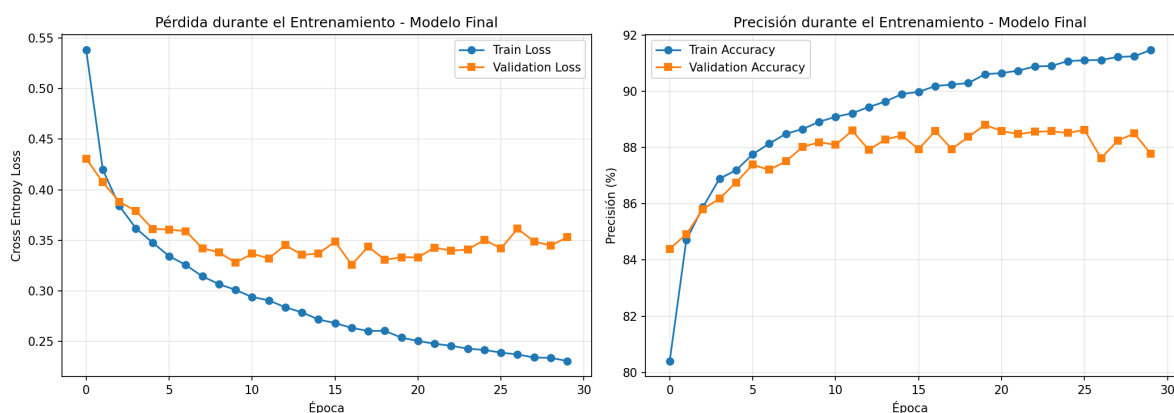


Figure 1: Curvas de Entrenamiento Final

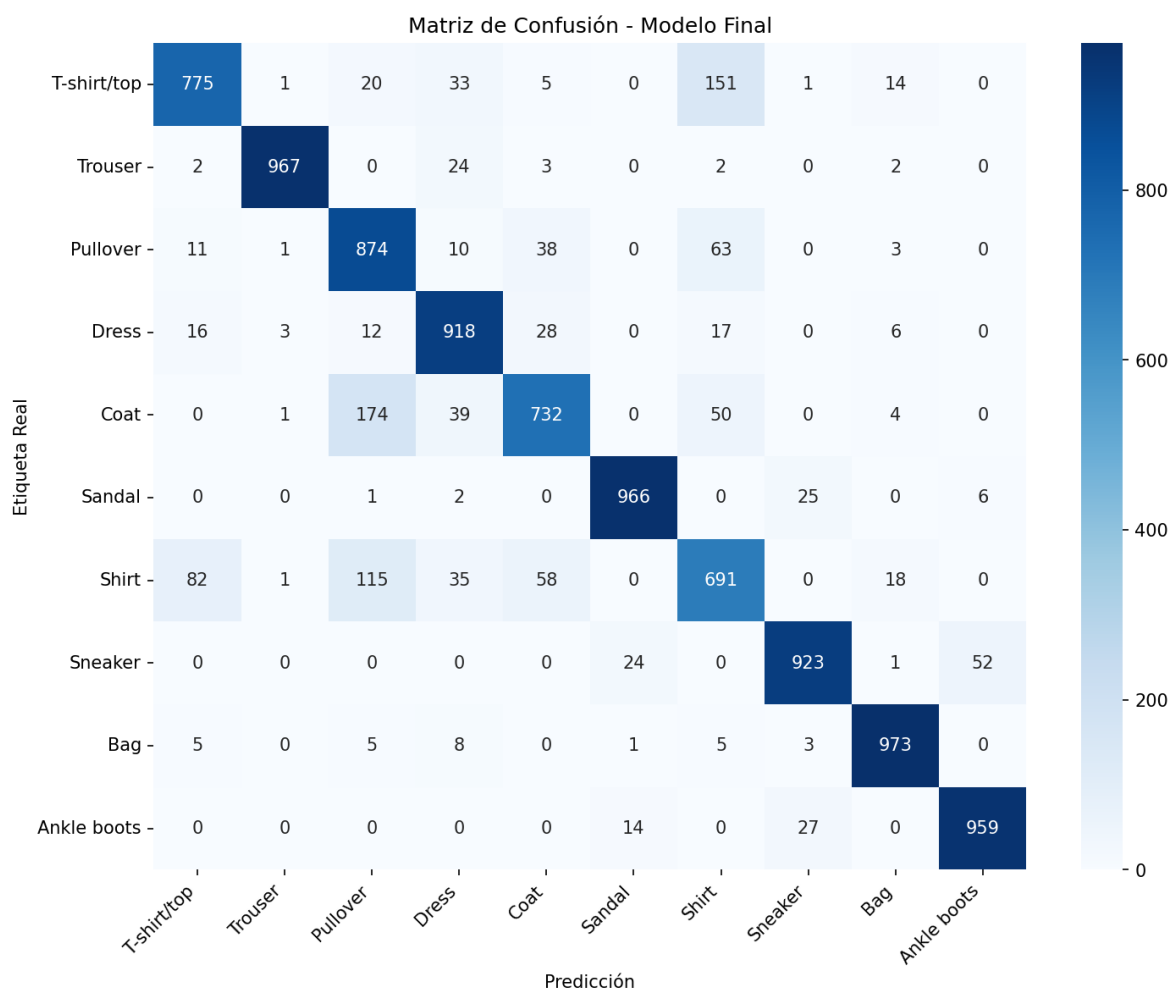


Figure 2: Matriz de Confusión Final

Discusión y Conclusiones

Los resultados demuestran que la selección adecuada de hiperparámetros es crucial para el rendimiento del modelo. El optimizador ADAM mostró una ventaja significativa sobre SGD, alcanzando 88.06% de precisión en validación frente a 73.44% de SGD. La combinación óptima de hiperparámetros (ADAM, LR=0.001, arquitectura 256-128, batch size 32, dropout 0.2, 30 épocas) logró **87.99%** de precisión en validación.

Configuración óptima recomendada: - Optimizador: ADAM - Learning Rate: 0.001 - Arquitectura: 256-128 - Batch Size: 32 - Épocas: 30 - Dropout: 0.2

El análisis de la matriz de confusión revela que la categoría “Shirt” presenta la mayor dificultad de clasificación, posiblemente debido a su similitud visual con otras prendas como “T-shirt/top” y “Pullover”. El modelo alcanza un rendimiento dentro del rango esperado para Fashion-MNIST (85-90%), demostrando un buen balance entre capacidad de aprendizaje y generalización.

Referencias

1. PyTorch Documentation. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. <https://pytorch.org/docs/stable/index.html>
2. Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. arXiv preprint arXiv:1708.07747.
3. Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980.
4. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). *Dropout: A simple way to prevent neural networks from overfitting*. Journal of Machine Learning Research, 15(1), 1929-1958.
5. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning representations by back-propagating errors*. Nature, 323(6088), 533-536.