Autoencoder y Clasificador Convolucionales sobre Fashion-MNIST

Ignacio Martinez Goloboff*
Redes Neuronales, Licenciatura en Ciencias de la Computación,
Facultad de Matemática, Astronomía, Física y Computación
(Dated: February 25, 2025)

Este escrito explora el análisis y desarrollo de dos arquitecturas de redes neuronales para la reeconstrucción y clasificación de imágenes pertenecientes al conjunto de datos Fashion-MNIST [1], un recurso ampliamente utilizado en experimentos de reconocimiento de patrones en artículos de moda. Utilizando PyTorch como marco principal, se investigan los efectos del ajuste de varios hiperparámetros y se compara el desempeño del modelo.

I. INTRODUCCIÓN

En la actualidad, la clasificación de imágenes representa uno de los desafíos más importantes en el ámbito del aprendizaje automático, con aplicaciones que abarcan desde el reconocimiento de patrones hasta la toma de decisiones basada en datos visuales. En este contexto, el conjunto de datos Fashion-MNIST se ha convertido en un referente para evaluar el rendimiento de modelos de clasificación.

Este trabajo tiene como objetivo desarrollar e implementar un modelo basado en autoencoders y clasificadores que permita revelar la eficacia y potencial de la combinación de estas redes en la clasificación de imágenes de Fashion-MNIST.

II. TEORÍA

El entrenamiento de redes neuronales supervisadas [2], como las redes feed-forward, pueden dividirse en una secuencia de pasos definidos:

1. **Preparación del Dataset:** El conjunto de datos Fashion-MNIST [1] es una herramienta ampliamente utilizada en experimentos de aprendizaje automático. Este dataset consta de 70,000 imágenes en escala de grises (28x28 píxeles) divididas en 10 categorías, ver FIG 1. Las categorías incluyen prendas de vestir como camisetas, zapatos y abrigos. Para este trabajo, se divide en 60,000 imágenes para entrenamiento y 10,000 para validación, asegurando que las particiones representen equilibradamente las clases.

2. **Diseño de la Arquitectura:** El modelo implementado se compone de un autoencoder convolucional [3], utilizado para reconstrucción de imágenes, y un clasificador que emplea el encoder preentrenado con una capa lineal para la clasificación.

   El autoencoder consta de dos partes: un codificador, que reduce la dimensionalidad de los datos a través de capas convolucionales, comprimiéndolos en una etapa intermedia o espacio latente; y un decodificador, que reconstruye la imagen original a partir de esta representación comprimida, utilizando convoluciones transpuestas.

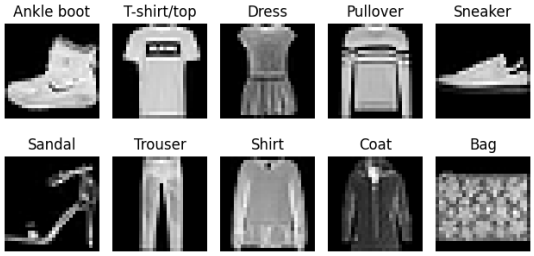

FIG. 1: Ejemplos de prendas en el dataset.

3. **Función de pérdida y optimización:** Se utiliza la Cross Entropy Loss (clasificador) y el Error Cuadrático Medio (autoencoder) como funciones de pérdida para medir el error entre las probabilidades predichas por el modelo y las clases verdaderas. Usaremos el optimizador Adam que ajusta iterativamente los pesos para mejorar el rendimiento del modelo y minimizar esta función de pérdida.

4. **Entrenamiento y Validación:** El entrenamiento del modelo se realiza de forma iterativa mediante batches, ajustando los pesos a través de backpropagation. Durante la validación, se evalúa el rendimiento en datos no vistos y se ajustan hiperparámetros si es necesario. Finalmente, en una prueba, la red se evalúa en un conjunto independiente para medir su capacidad de generalización.

A. Arquitectura del Autoencoder

La primera parte del proyecto se centró en desarrollar un autoencoder convolucional que tenga la capacidad de aprender representaciones comprimidas y eficientes de imágenes, para luego reconstruirlas de la manera más fiel posible a la original. Como ya mencionamos anteriormente, el autoencoder consta de dos módulos: El

encoder posee dos capas convolucionales 2D y una lineal, mientras que el decoder realiza una transformación inversa, primero utiliza una capa lineal y luego dos convolucionales transpuestas. Ver FIG 2.

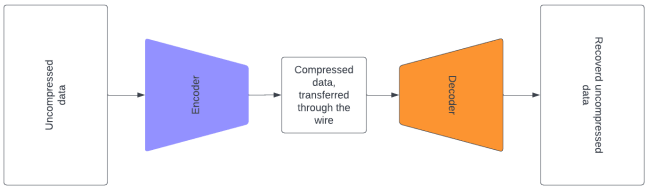

FIG. 2: Arquitectura general de una red Autoencoder.

Para el entrenamiento del autoencoder, se utilizaron los siguientes parámetros:

Hiperparámetro	**Valor utilizado**
Tasa de aprendizaje (lr)	0.003
Dropout (p)	0.2
Tamaño del batch	100
Épocas	60
Loss Function	Error Cuadrático Medio
Optimizador	Adam

TABLE I: Hiperparámetros del Autoencoder

B. Arquitectura del Clasificador

La segunda parte consistió en adaptar el autoencoder para realizar tareas de clasificación de imágenes. Para ello, se reutilizó el encoder convolucional del modelo previo y se añadió una capa lineal, obteniendo así una salida de 10 neuronas (una por cada categoría del dataset).

Para el entrenamiento del clasificador, se utilizaron los mismos hiperpárámetros que el modelo anterior, nada mas cambiando la función de error, la cual ahora pasó a ser la Cross Entropy Loss.

III. RESULTADOS

En esta sección, se analizan los resultados obtenidos durante las pruebas de entrenamiento de la redes neuronales.

A. Entrenamiento Inicial

Como ejemplo inicial se utilizará el autoencoder con los hiperparámetros básicos, mencionados en la TABLA I. Luego, los resultados que se obtuvieron se pueden ver en FIG 3.

Se puede observar que a medida que van pasando las épocas la pérdida se empieza a estabilizar. Por lo tanto de ahora en adelante se utilizarán 20 epochs en vez de 60 para los posteriores análisis, ya que se obtienen valores razonables sin tanta necesidad de mayor cómputo.

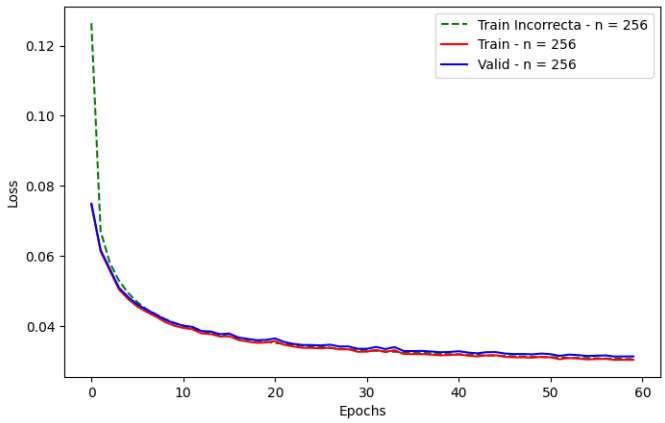

FIG. 3: Loss vs Epochs del entrenamiento inicial

B. Número de neuronas en la capa intermedia

Empezaremos viendo como influyen en el aprendizaje de el autoencoder convolucional un número diferente de neuronas en su capa oculta intermedia. A continuación, se presentan los gráficos obtenidos y su análisis correspondiente.

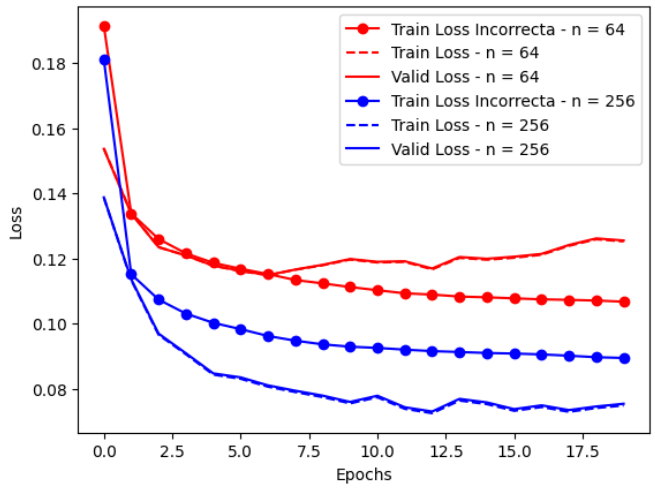

FIG. 4: Loss vs Epochs con distintos n° de neuronas en la capa intermedia del Autoencoder

Los resultados obtenido, ver FIG 4, muestran que en valores bajos de n, la pérdida es más alta, lo que sugiere que un mayor espacio latente en capas intermedias permite una mejor capacidad del modelo para reconstruir las

imágenes de entrada. Debido a esto, a partir de ahora usaremos n = 256 para los posteriores análisis.

C. Dropout

Ahora compararemos la pérdida en el entrenamiento con y sin dropout, ver FIG 5. Se observa que sin dropout, la pérdida en entrenamiento es menor, lo que indica que el modelo aprende más rápido. Con dropout el modelo presenta una mayor pérdida, dandonos a entender que en redes con pocas capas puede ser contraproducente.

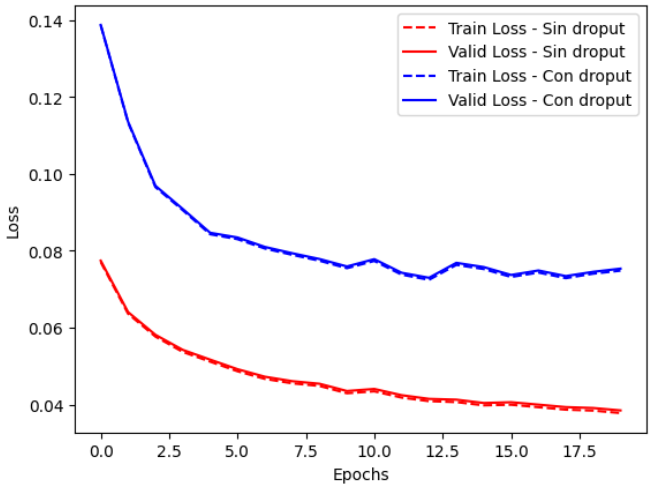

FIG. 5: Comparación entre el uso o no del dropout en el Autoencoder

D. Preentrenado del Clasificador

En este experimento, se compararon dos enfoques para el entrenamiento de la red: por un lado, se entrenó completamente la arquitectura, ajustando tanto los pesos del encoder como los de la capa clasificadora; por otro, se utilizó un modelo preentrenado, donde los pesos del encoder fueron tomados del autoencoder previamente entrenado y solo se ajustaron los de la capa clasificadora.

En el primer gráfico, ver FIG 6, se observa que la red entrenada completamente desde cero logra una disminución más pronunciada de la pérdida a lo largo de las épocas, alcanzando valores finales más bajos en comparación con el modelo preentrenado.

En el segundo gráfico, ver FIG 7, se refuerza esta conclusión. Pero aunque la red entrenada desde cero alcanza mayor precisión, también muestra overfitting, evidenciado por la creciente separación entre las curvas de entrenamiento y validación. En cambio, la red preentrenada mantiene curvas más alineadas, indicando mejor generalización a costa de una ligera pérdida de precisión.

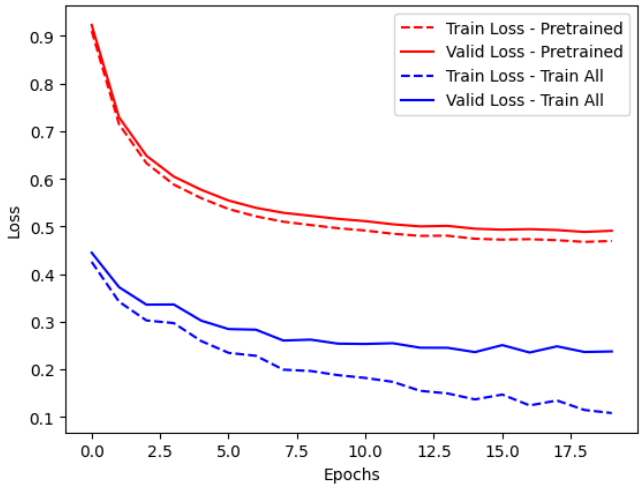

FIG. 6: Comparación de la Pérdida entre la red completamente entrenada y la preentrenada

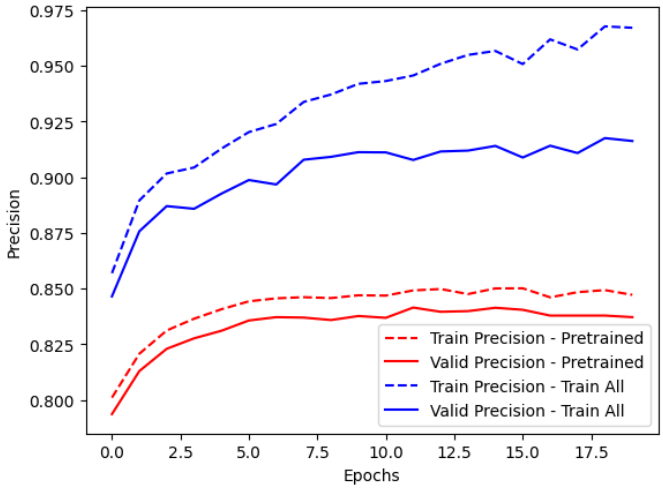

FIG. 7: Comparación de la Precisión entre la red completamente entrenada y la preentrenada

IV. DISCUSIÓN

El análisis de la matriz de confusión, ver FIG 8, la cual representa el desempeño del clasificador indicando aciertos y errores para cada clase. La misma nos muestra que el modelo preentrenado tiene un buen rendimiento en la mayoría de las clases, con alta precisión en la mayoría de categorías con algunas pequeñas confusiones en determinadas clases.

Pero, en la segunda imagen, ver FIG 9, las predicciones del modelo coinciden en su mayoría con las etiquetas originales, confirmando que el modelo ha aprendido representaciones útiles.

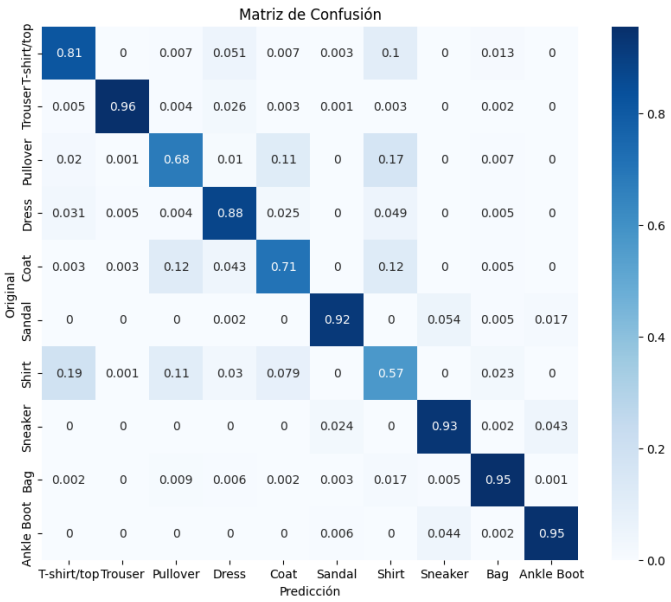

FIG. 8: Matriz de Confusión entrenando solo la capa clasificadora

FIG. 9: Resultados predichos del Clasificador

V. CONCLUSIONES

Este trabajo abordó la implementación de un autoencoder convolucional y su posterior adaptación como clasificador de imágenes de Fashion-MNIST. Se compararon distintas configuraciones, evaluando el impacto del preentrenamiento sobre la precisión y la generalización del modelo.

- En el autoencoder, pudimos observar que aumentar las neuronas en las capas ocultas disminuyó el error. Se evidencia un equilibrio entre mejorar la precisión a costa de mayor complejidad.

- Existe un trade-off entre reducir el error y el tiempo de entrenamiento (*epochs*), ya que, tras varias épocas, la mejora se estabiliza y seguir entrenando aporta ganancias marginales.

- El dropout en redes de pequeñas dimensiones no presenta una mejora en la precisión del modelo.

- Los resultados muestran que el preentrenamiento del autoencoder contribuye a mejorar la estabilidad del entrenamiento y mitiga mejor el sobreajuste observado en la red completamente entrenada, ofreciendo la misma resultados razonables con menor costo computacional.

El estudio destaca el valor del preentrenamiento como una estrategia clave para mejorar la eficiencia en tareas de aprendizaje profundo.

VI. AGRADECIMIENTOS

IM agradece a los profes por la oportunidad y la orientación, que fueron clave para la realización de este trabajo.

* ignacio.martinez.goloboff@mi.unc.edu.ar

[1] Fashion-MNIST, Fashion-mnist — Wikipedia, the free encyclopedia (2025), [Online; accessed 24-February-2025].
[2] M. A. Nielsen, *Deep Learning* (Determination Press, 2016) [Online; accessed 24-February-2025].
[3] Autoencoder Convolucional, Geeksforgeeks (2025), [Online; accessed 24-February-2025].