Autoencoder convolucional y clasificador sobre dataset Fashion-MNIST

Salomón, Nicolás*
Fundación Fulgor, Ernesto Romagosa 518, Córdoba, Argentina and
Universidad Nacional del Sur, Bahía Blanca, Argentina
(Dated: February 5, 2024)

Este trabajo de investigación se centra en explorar la sinergia entre dos arquitecturas de redes neuronales para la reconstrucción y clasificación de imágenes pertenecientes al conjunto de datos Fashion-MNIST. La arquitectura propuesta demuestra un rendimiento excepcional al combinar la capacidad de reconstrucción precisa inherente a los autoencoders con la eficiencia de clasificación requerida. Los resultados obtenidos revelan el potencial de estas redes para la reconstrucción y clasificación de prendas de vestir. El código de desarrollo se encuentra disponible en `https://github.com/nicolassalomon96/Redes_Neuronales_FAMAF/blob/main/Autoencoder_TP_Final/TP_Final.ipynb`.

I. INTRODUCCIÓN

En la era actual de la inteligencia artificial, las redes neuronales desempeñan un papel fundamental en la capacidad de las máquinas para aprender patrones y realizar tareas complejas. Entre las diversas arquitecturas de redes neuronales, los autoencoders y las redes clasificadoras han destacado como herramientas poderosas para la representación y la clasificación de datos.

En este contexto, el conjunto de datos Fashion-MNIST [1, 2] se ha convertido en un banco de pruebas esencial para evaluar el rendimiento de modelos de aprendizaje automático en la tarea de reconocimiento de prendas de moda.

A través de este enfoque integrado, se pretende revelar la eficacia y el potencial de la combinación de autoencoders y clasificadores en el contexto del dataset *Fashion-MNIST*.

II. MARCO TEÓRICO

El proceso de entrenamiento de una red neuronal puede dividirse en pasos perfectamente definidos:

- **Recopilación y preparación del Dataset:** El entrenamiento de una red neuronal supervisada requiere de datos de entrada con un formato específico acorde a cada tipo de red.

  El conjunto de datos Fashion-MNIST es una gran base de datos de imágenes de moda disponible gratuitamente, que se utiliza comúnmente para entrenar y probar varios sistemas de aprendizaje automático.[1, 2]. El conjunto de datos contiene 70.000 imágenes en escala de grises de 28x28 píxeles de prendas de vestir, repartidas en 10 categorías, con 7.000 imágenes por categoría (ver Tabla I). Algunos ejemplos de los distintos tipos de prendas pueden apreciarse en la Figura 1. En el presente trabajo, el conjunto de entrenamiento consta de 60.000 imágenes, el conjunto de validación consta de 9.900 imágenes y el conjunto de prueba consta de 100 imágenes.

Label	Class	Label	Class
0	T-shirt/top	5	Sandal
1	Trouser	6	Shirt
2	Pullover	7	Sneaker
3	Dress	8	Bag
4	Coat	9	Ankle boot

TABLE I. Clases del Dataset Fashion-MNIST

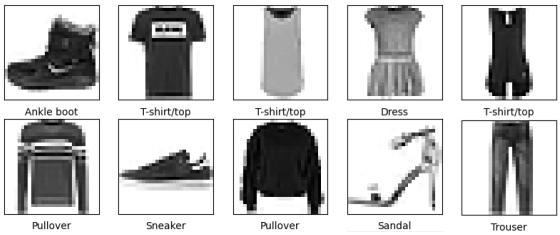

FIG. 1. Ejemplos de prendas del Dataset Fashion-MNIST

- **Creación de la arquitectura e inicialización de hiperparámetros:** Se crea la arquitectura a analizar y se fijan valores iniciales para los hiperparámetros y los pesos de la red. En el presente trabajo de investigación existirán dos arquitecturas, una compuesta por un autoencoder convolucional, para reconstruir una imagen de entrada, y como segundo paso se empleará un encoder convolucional seguido de una capa densa, para realizar la tarea de clasificación.
  La arquitectura autoencoder convolucional consta de dos partes principales: un codificador que reduce la dimensionalidad de los datos de entrada a través de capas convolucionales, resultando en una etapa intermedia conocida como espacio latente; y un decodificador que reconstruye los datos originales a partir de la representación comprimida generada por el codificador, por medio de convoluciones transpuestas, ver Figura 2.

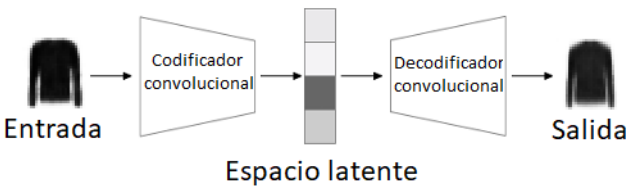

FIG. 2. Arquitectura general de una red Autoencoder

- **Establecimiento de la función de error y el optimizador a emplear:** Se fija la función que se empleará para cuantificar la diferencia entre la salida predicha por la red y la salida esperada. Adicionalmente se establece la función encargada de optimizar los parámetros de la red, considerando el error calculado previamente.
- **Entrenamiento:** Proceso iterativo, por medio del uso de batches (o lotes) de datos de entrada, para calcular el error y realizar el backpropagation de los gradientes para actualizar los pesos de la red.
- **Validación:** Empleando un conjunto de datos de validación, se evalúa el rendimiento de la red en datos no vistos durante el entrenamiento y se modifican hiperparámetros, de ser necesario, para mejorar el rendimiento.
- **Prueba:** Finalmente, la red entrenada se evalúa en un conjunto de datos de prueba independiente para medir su rendimiento general en datos no vistos.

III. IMPLEMENTACIÓN

A. Arquitectura Autoencoder convolucional para la reconstrucción de imágenes

La primera etapa del proyecto se centró en implementar un autoencoder convolucional que tenga la capacidad para aprender representaciones eficientes y comprimidas de las imágenes de entrada, para posteriormente descomprimir dichas representaciones y obtener una imagen lo más similar posible a la ingresada en la red.
Luego de múltiples pruebas, variando hiperparámetros, número de filtros convolucionales y capas ocultas, se obtuvo el mejor desempeño con la red representada en la Figura 3.

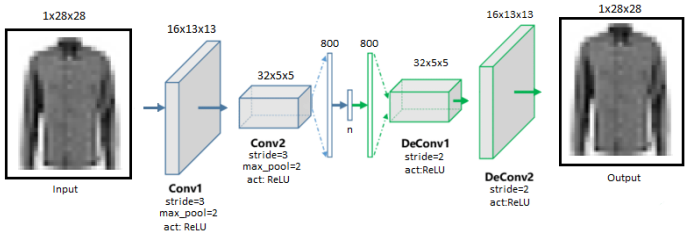

FIG. 3. Arquitectura empleada de la red Autoencoder convolucional

El entrenamiento se llevó a cabo empleando la función de Error Medio Absoluto (MAE) en conjunto con el optimizador ADAM, con un learning rate igual a 10^{-3}. La elección de la función de error se llevó a cabo luego de resultados obtenidos al realizar pruebas de contraste entre el Error Medio Absoluto y el Error Cuadrático Medio (MSE). Las mismas demostraron que el empleo del MSE introducía un error en los bordes de las imágenes reconstruidas (ver Figura 4), posiblemente por su alta sensibilidad a valores atípicos, en conjunto con el *output_padding* empleado en las capas deconvolucionales.

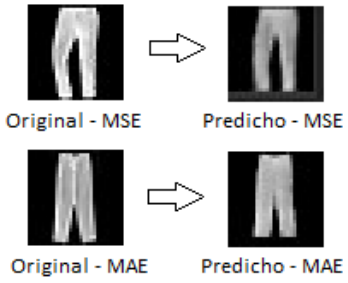

FIG. 4. Comparación entre el uso de MSE vs MAE

Por último, no se emplearon capas de Dropout debido a que las mismas empeoraban el rendimiento de la red durante el entrenamiento, haciendo que el error no decrezca establemente, como puede apreciarse en la Figura 5. Este resultado puede deberse a que el dropout agrega cierta inestabilidad al entrenamiento y en el caso de redes de pocas capas puede ser contraproducente.

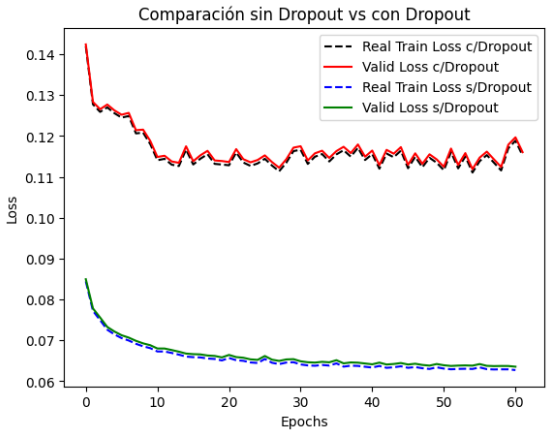

FIG. 5. Comparación entre el uso o no de Dropout

B. Arquitectura Autoencoder convolucional para clasificación de imágenes

La segunda etapa del presente proyecto se basó en la implementación de un autoencoder convolucional que tenga la capacidad de determinar a que clase pertenece cada imagen de entrada. Para llevar a cabo este trabajo, se implementó un encoder convolucional (primera etapa de una red autoencoder, ver Figura 2), al cual se le agregó una capa densa compuesta por un número arbitrario de neuronas, seguido otra capa de salida con 10 neuronas (una por clase). Una representación de la arquitectura puede observarse en la Figura 6. Para el entrenamiento de la misma se empleó la Entropía Cruzada (CrossEntropy) como función de pérdida, en conjunto con el optimizador ADAM, con un learning rate de 10^{-3}.

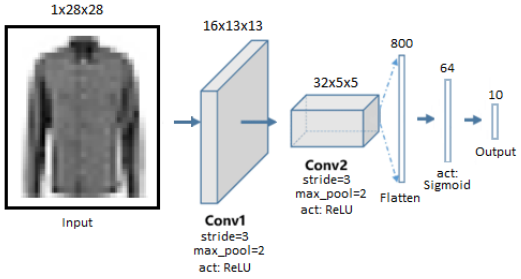

FIG. 6. Arquitectura autoencoder clasificador

Luego, a la salida se obtienen las probabilidades de pertenencia a cada clase para la imagen de entrada. Por lo que, tomando la posición de la máxima probabilidad, podemos obtener la clase predicha por la red neuronal.

Se llevaron a cabo dos tipos de entrenamientos:

- En primer lugar, se entrenó toda la red (encoder + clasificador) partiendo de valores iniciales aleatorios para los pesos de la misma.
- En segundo lugar, se entrenó únicamente la red clasificadora (últimas capas densas), "congelando" los pesos pertenecientes a la mejor red encoder entrenada y descripta en la subsección anterior.

IV. RESULTADOS

A. Arquitectura Autoencoder convolucional para la reconstrucción de imágenes

Se realizó el entrenamiento de tres modelos de autoencoder convolucionales (cada uno con un número diferente de neuronas en su capa densa intermedia) y se evaluó la pérdida de entrenamiento y validación de cada uno de ellos. Los resultados de estas pruebas pueden observarse en la Figura 7.

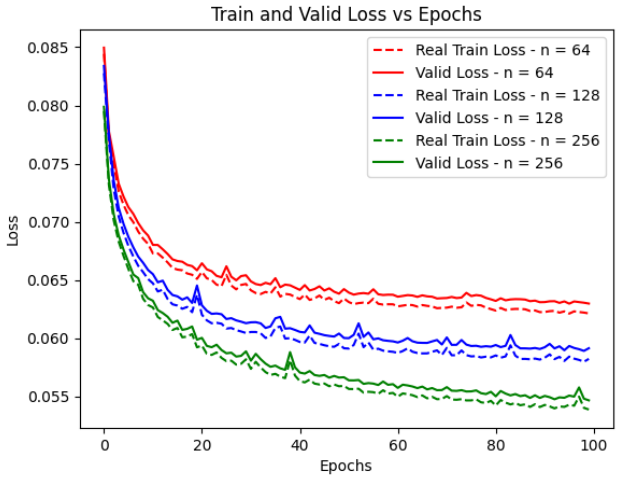

FIG. 7. Pérdida de entrenamiento y validación de cada modelo respecto a la época.

Tanto para el error de entrenamiento cómo para el error de validación puede observarse como al aumentar el numero de neuronas en la capa densa oculta, se reduce el error. En este punto nos encontramos con un *trade-off* entre simpleza y precisión de nuestra red. Dado que el fin de este trabajo no es una implementación optimizada que reduzca al máximo el tiempo de inferencia y tamaño de la red, podría considerarse como superadora a la red que posee 256 neuronas en su capa densa oculta. Por último, en las distintas pruebas realizadas no se encontró un punto en el cual el error comience a incrementarse, denotando un overfitting de la red, sino que el mismo tiende a estabilizarse y continuar decreciendo de forma muy lenta. Por lo tanto se determinó que 100 épocas fue un valor apropiado, y a partir del cual aumentar las iteraciones no mejora significativamente el rendimiento de la red.

El correcto funcionamiento de la red puede apreciarse en la Figura 8, donde se evalúa el proceso de aprendizaje durante el entrenamiento de la red con 256 neuronas en su capa densa oculta. Se observa cómo al aumentar la época de entrenamiento, las predicciones son cada vez mejores. Además, se observa en la Figura 9 como las predicciones finales realizadas por la red sobre el dataset de prueba son acordes a las imágenes de entrada.

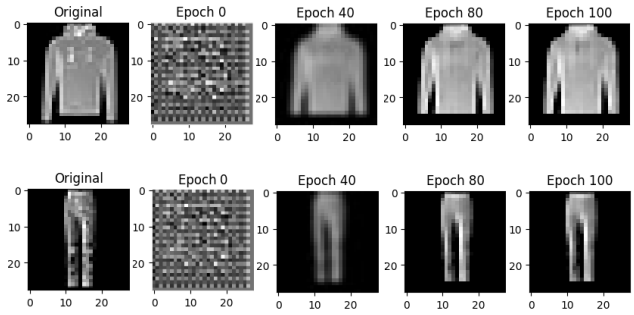

FIG. 8. Predicción de una imagen respecto a la época

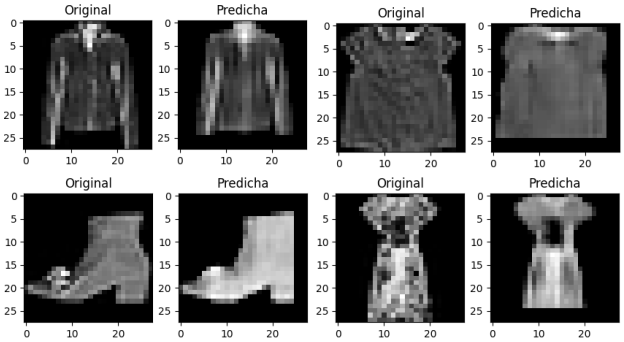

FIG. 9. Inferencias realizadas sobre el dataset de test

B. Arquitectura Autoencoder convolucional para clasificación de imágenes

Una vez desarrollada de forma satisfactoria la red encargada de la reconstrucción de las imágenes de entrada, se entrenó una red de tipo encoder + clasificador tomando como punto de partida el encoder entrenado anteriormente, "congelando" sus pesos y entrenando únicamente la red clasificadora (capas densas, ver Figura 6). Además se evaluaron dos arquitecturas distintas, variando el número de neuronas de la capa densa oculta en 64 o 256 neuronas.

Por último, se realizó el mismo entrenamiento, pero partiendo de valores aleatorios para los pesos iniciales, sin considerar el encoder preentrenado, y variando de igual forma el número de neuronas de la capa densa oculta en 64 o 256 neuronas.

Los resultados de estas pruebas pueden apreciarse en las Figuras 10 y 11, en conjunto con las pruebas realizadas sobre el dataset de test y su matriz de confusión (Figuras 12 y 13), sobre el modelo de 256 neuronas en su capa densa oculta, con pesos iniciales aleatorios.

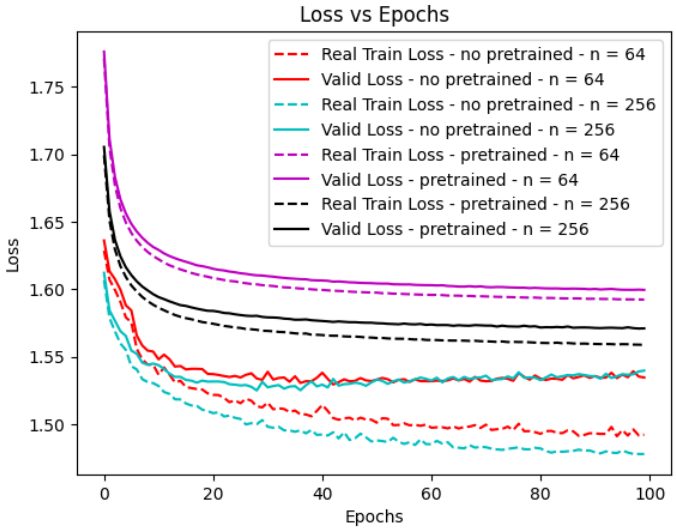

FIG. 10. Loss vs Epochs durante el ciclo en entrenamiento

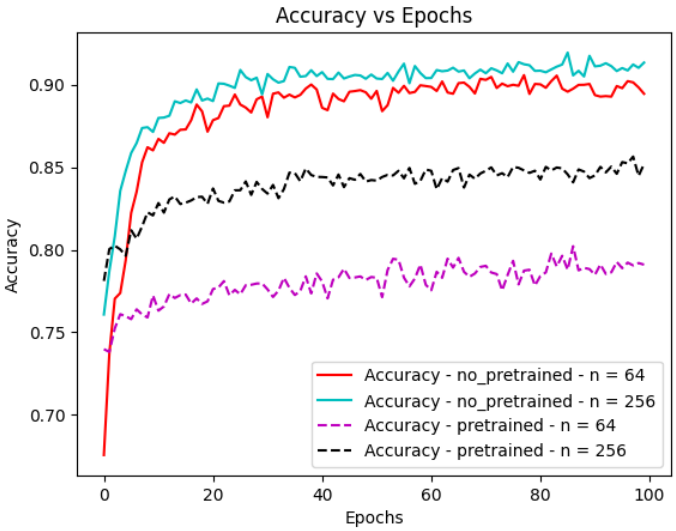

FIG. 11. Accuracy vs Epochs durante el ciclo en entrenamiento

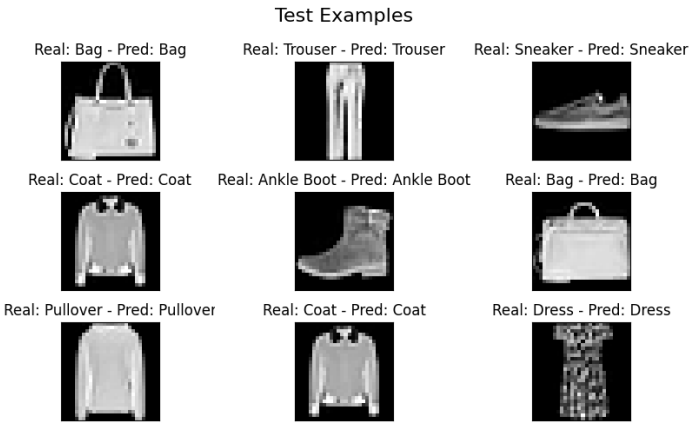

FIG. 12. Resultados predichos sobre el dataset de test

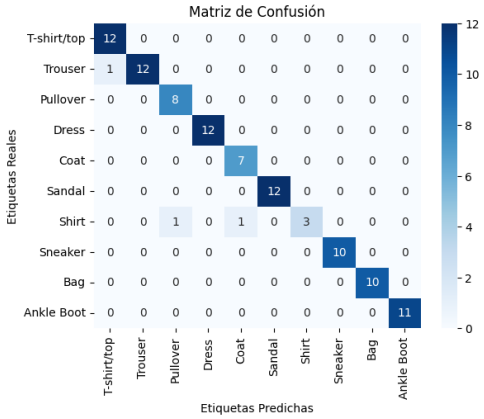

FIG. 13. Matriz de confusión sobre el dataset de test

V. CONCLUSIÓN

En este trabajo de investigación, se exploró la eficacia de una red autoencoder convolucional con capacidad reconstructora y clasificadora en el desafiante conjunto de datos Fashion-MNIST. Los resultados obtenidos revelan que la combinación de la capacidad de reconstrucción y la habilidad de clasificación inherentes a los autoencoders convolucionales puede ofrecer mejoras significativas en la precisión de la clasificación de prendas de vestir.

Se evaluaron múltiples arquitecturas, de las cuales podemos concluir:

- Para redes de pequeñas dimensiones, como la presentada, el uso de dropout no mejora la precisión de la misma.
- Es de suma importancia contemplar múltiples funciones de error para analizar cual se adapta mejor a nuestro propósito.
- Tanto para la red autoencoder reconstructora de imágenes, como para la red clasificadora, el empleo de un mayor número de neuronas en sus capas densas ocultas significó una disminución en el error de inferencia. Esto representa un compromiso entre obtener mayor precisión a costa de mayor complejidad.
- La arquitectura clasificadora junto con la red encoder preentrenada no brindó un mejor desempeño en cuanto a la pérdida y precisión alcanzada, siendo superada por la red entrenada desde valores iniciales aleatorios, logrando un 91% de precisión para la red de 256 neuronas. Además no hubo una diferencia de tiempo entre ambos tipos de entrenamiento.
- Los resultados de las pruebas realizadas sobre el dataset de test demuestran la versatilidad y el potencial de la arquitectura autoencoder analizada, en conjunto con una matriz de confusión con muy pocos fallos

* nicolassalomon96@gmail.com

[1] H. Xiao, K. Rasul, and R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017), arXiv:1708.07747 [cs.LG].

[2] Fashion-mnist dataset, `https://github.com/zalandoresearch/fashion-mnist`.