Análisis y Aplicación de autoencoders para reconstrucción y clasificación en Fashion-MNIST

Adolfo Banchio*
Facultad de Matemática, Astronomía, Física y Computación, Universidad Nacional de Córdoba
(Dated: 2 de diciembre de 2024)

En este trabajo se implementan y analizan *autoencoders* convolucionales aplicados al dataset Fashion-MNIST, explorando su capacidad de compresión y reconstrucción de imágenes, así como la reutilización de los *encoder*s en tareas de clasificación. Se presentan dos arquitecturas principales: una con una capa lineal intermedia que permite controlar el tamaño del espacio latente, y otra completamente convolucional. A partir de múltiples configuraciones y experimentos, se identificó que un tamaño de espacio latente intermedio balancea capacidad de aprendizaje y eficiencia computacional. En la tarea de clasificación, se evaluaron tres estrategias de entrenamiento: utilizando *encoders* pre-entrenados, reentrenados y entrenados desde cero. Los resultados demuestran que, aunque el preentrenamiento puede ser beneficioso, la tarea específica de clasificación puede requerir un ajuste más profundo para maximizar el desempeño.

I. INTRODUCCIÓN

En el aprendizaje no supervisado, la reconstrucción de imagenes y extracción de características de las mismas es una tarea muy realizada. Para ello los *autoencoders* son ampliamente utilizados. Por su buena capacidad de adaptación a datos espaciales y capacidad de aprendizaje de propiedades especificas de los datos.

Este trabajo utiliza el dataset Fashion-MNIST, compuesto por imágenes de 28×28 píxeles en escala de grises de prendas de vestir distribuidas en 10 categorías.[1] El objetivo es explorar cómo el tamaño del espacio latente afecta la capacidad de reconstrucción y cómo los *encoders* pre-entrenados pueden reutilizarse en problemas supervisados de clasificación.

Se implementaron dos arquitecturas principales: (1) un *autoencoder* con capa lineal intermedia que controla explícitamente la dimensionalidad del espacio latente y (2) un *autoencoder* completamente convolucional. A partir de estas implementaciones, se realizaron experimentos variando el tamaño del espacio latente y comparando los resultados en tareas de reconstrucción y clasificación.

II. TEORÍA

Un *autoencoder* convolucional es un tipo de red neuronal diseñada para extraer características y generar una representación compacta de los datos a través del uso de capas convolucionales. Esto permite crear capacidades de compresión, reconstrucción y clasificación de manera no supervisada. La arquitectura típica de un *autoencoder* consta de dos componentes principales un *encoder* y un *decoder*.[2]

En este trabajo, se implementaron dos variantes de un *autoencoder*, uno con capa lineal que incluye una capa totalmente conectada entre *encoder* y *decoder*, lo que permite controlar la dimension del espacio latente via el parametro *l_size*. La otra variante elimina la capa lineal, manteniendo la dimension del espacio latente determinado por las capas convolucionales.

En ambas variantes, las capas convolucionales de los *encoders* eran las mismas, donde luego de cada convolucion se utilizaba una funcion de activación ReLU y *dropout* de 0.2. En la tabla I se encuentran los parametros utilizados para las capas correspondientes al encoder.

Capa	*in channels*	*out channels*	*Kernel*	*stride*	*padding*
Conv2d	1	32	3	1	1
MaxPool	32	32	2	2	-
Conv2d	32	64	3	1	1
MaxPool	64	64	2	2	-

Cuadro I. Parámetros de las capas convolucionales utilizadas para el encoder del autoencoder.

El desempeño de los *autoencoders* se evaluó utilizando el Error Cuadrático Medio como función de costo y el optimizador ADAM con un learning rate de 0.001. Luego, la configuración con mejor desempeño se utilizó en la segunda etapa del trabajo para entrenar el clasificador supervisado.

En la clasificación, se añadió una sola capa lineal al *encoder* seleccionado para mapear la información comprimida a las 10 clases de Fashion-MNIST. Se compararon tres estrategias:

1. Entrenar solo la capa clasificadora con el *encoder* pre-entrenado.

2. Entrenar todas las capas, incluyendo el *encoder* pre-entrenado.

3. Entrenar toda la red desde cero, sin un *encoder* pre-entrenado.

Para estos tres experimentos se utilizó la Entropía Cruzada como función de pérdida y ADAM como optimizador con un learning rate de 0.001.

III. RESULTADOS

En esta sección se presentaran los datos obtenidos para cada caso. Primero se presentaran los datos obtenidos al variar la dimensión del espacio latente del *autoencoder*. En la tabla II podemos ver y comparar los promedios del costo para el conjunto de entrenamiento y validación para los tres casos evaluados con el *autoencoder* de capa lineal.

Caso	*l_size*	Costo entrenamiento	Costo validación
1	64	0.0085	0.0087
2	512	0.0072	0.0074
3	1024	0.0074	0.0076

Cuadro II. Tabla comparativa de los resultados para los experimentos sobre el *autoencoder* convolucional con capa lineal

A. *autoencoder*

1. *l_size = 64*

El modelo con un espacio latente de 64 mostró un desempeño limitado en la reconstrucción de imágenes. Como se observa en la figura 1, el promedio del costo (ECM) aumentó durante el entrenamiento, indicando que la red no fue capaz de comprimir y reconstruir la información de manera efectiva. Este resultado evidencia que un espacio latente demasiado pequeño limita la capacidad del modelo para aprender representaciones significativas.

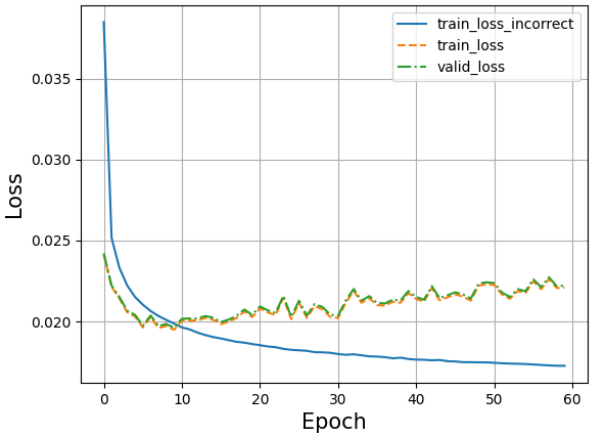

Fig. 1. Promedio de ECM para *autoencoder* con l_size = 64

2. *l_size = 512*

Incrementar el tamaño del espacio latente a 512 mejoró significativamente la capacidad del modelo para reconstruir las imágenes. Como se muestra en la figura 2, los valores de ECM disminuyeron tanto en el conjunto de entrenamiento como en el de validación, alcanzando valores finales de 0.0085 y 0.0089, respectivamente. Esto sugiere que este tamaño de espacio latente permite un balance adecuado entre compresión y reconstrucción.

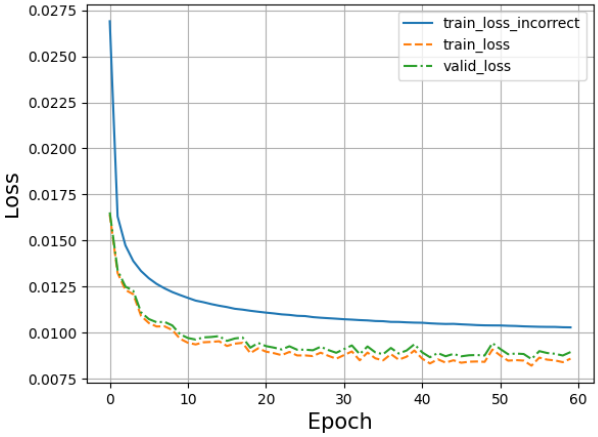

Fig. 2. Promedio de ECM para *autoencoder* con l_size = 512

3. *l_size = 1024*

Duplicar el espacio latente a 1024 no produjo una mejora significativa en la calidad de reconstrucción. Los valores finales de ECM (figura 3) fueron 0.0080 y 0.0076 para los conjuntos de validación y entrenamiento, respectivamente, lo que no justifica el aumento en el costo computacional asociado.

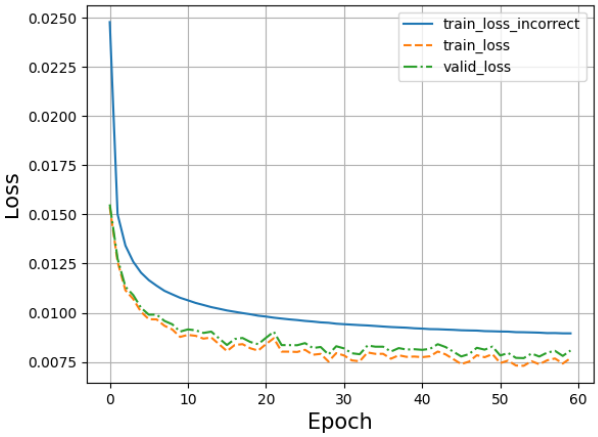

Fig. 3. Promedio de ECM para *autoencoder* con l_size = 1024

B. *autoencoder* convolucional

El *autoencoder* completamente convolucional eliminó la capa lineal, manteniendo las mismas capas convolucionales que las configuraciones previas. Como se observa en la figura 4, el modelo mostró una leve mejora en la reconstrucción de imágenes comparado con el *autoencoder* con $l\_size = 512$, probablemente debido a la continuidad en el flujo de información entre el *encoder* y el decoder.

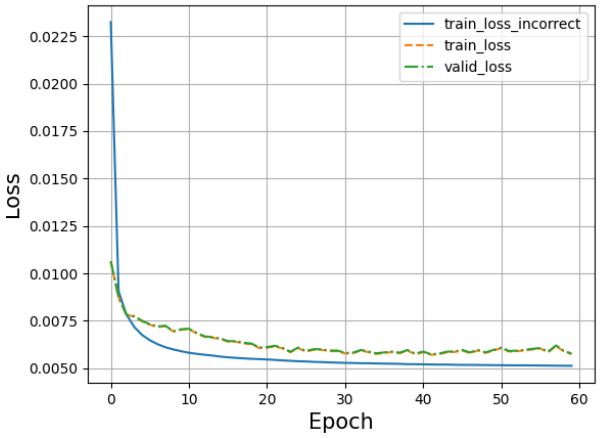

Fig. 4. Promedio de ECM para *autoencoder* totalmente convolucional

Para una comparación visual, la figura 5 muestra una imagen original del conjunto de validación junto con las reconstrucciones generadas por cada arquitectura.

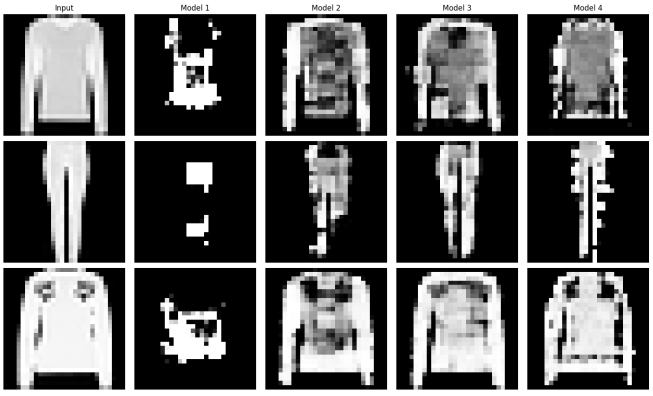

Fig. 5. Comparación de reconstrucción de imágenes para los 4 *autoencoder*s analizados

C. Clasificador

El *encoder* con $l\_size = 512$ se utilizó para construir un clasificador supervisado, dado que ofrecía un balance óptimo entre precisión de reconstrucción y eficiencia computacional.

1. *Entrenamiento solo clasificación*

En este experimento, solo se entrenaron los parámetros de la capa clasificadora, dejando el *encoder* pre-entrenado fijo. La figura 6 muestra que la precisión en validación alcanzó un máximo del 87 %, mientras que la precisión en el conjunto de entrenamiento fue del 89 %. Estos resultados reflejan limitaciones inherentes al no ajustar los pesos del *encoder*.

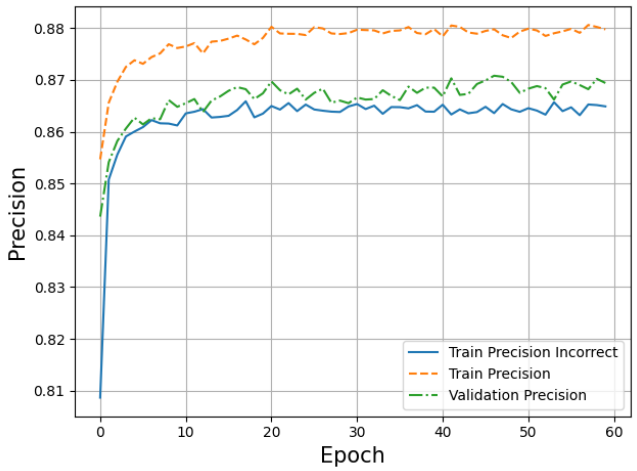

Fig. 6. Promedio de la precisión de la red clasificadora donde solo se entrena la capa de clasifiación y el encoder esta pre-entrenado

2. *Entrenamiento de encoder y clasificación*

Cuando se permitió el ajuste fino (*fine-tuning*) del encoder pre-entrenado, se observó una mejora significativa en la capacidad de generalización. Como muestra la figura 7, la precisión alcanzó el 90.2 % en validación y el 93.1 % en entrenamiento. Aunque se identificaron signos de sobreajuste, el modelo logró una distribución más balanceada de los aciertos como se puede ver en la matriz de confusión de la figura 9.

3. *Sin pre-entrenamiento*

Finalmente, se entrenó un clasificador desde cero, es decir, ajustando tanto el encoder como la capa clasificadora. Como se observa en la figura 8, la precisión final fue del 89 % en validación y del 92.8 % en entrenamiento, lo que lo hace competitivo con el ajuste fino, aunque con costos de entrenamiento más elevados. Al igual que en el

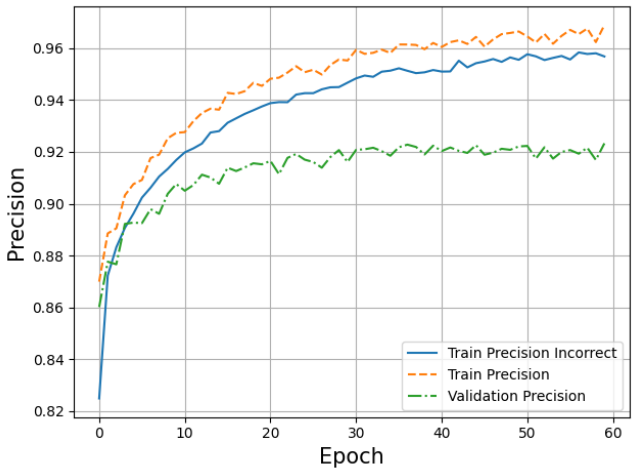

Fig. 7. Promedio de la precisión de la red clasificadora donde se realiza *fine tunning* sobre un encoder pre-entrenado

caso anterior, la matriz de confusión (fig 10) nos muestra una distribución balanceada de los aciertos.

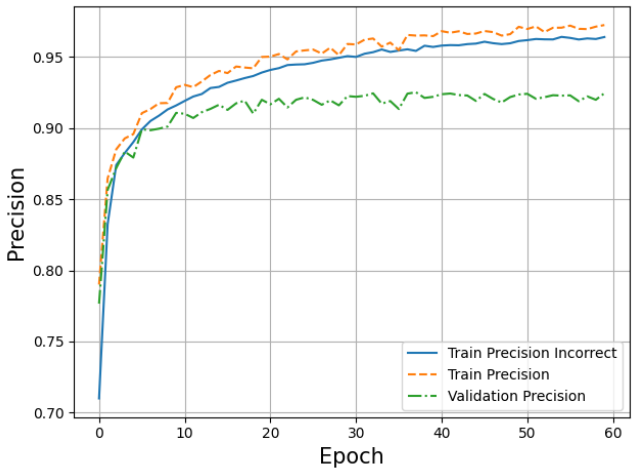

Fig. 8. Promedio de la precisión de la red clasificadora donde se la entrena desde cero.

IV. DISCUSIÓN

El objetivo principal de este trabajo fue analizar el impacto del tamaño del espacio latente en *autoencoders* convolucionales y evaluar el desempeño de *encoders* pre-entrenados en tareas de clasificación. A continuación, se discuten los hallazgos más relevantes y sus implicancias:

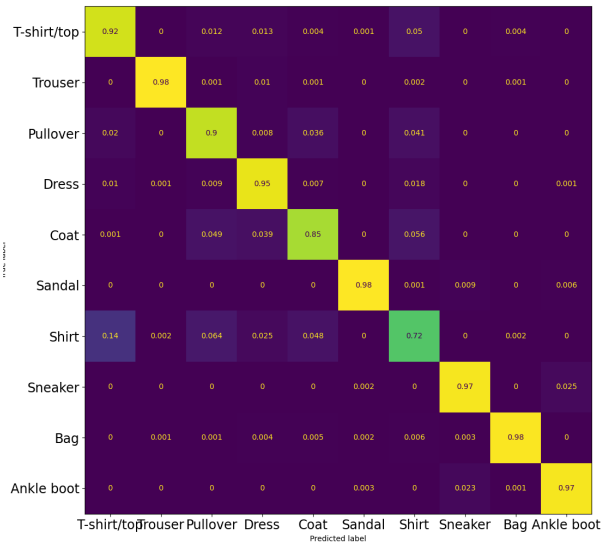

Fig. 9. Matriz de confusión para el clasificador con ajuste fino del *encoder* (caso 2).

A. Impacto del espacio latente

Los experimentos con diferentes tamaños del espacio latente (l_size) mostraron que un tamaño pequeño (64) limitó severamente la capacidad de reconstrucción debido a una compresión excesiva de la información, mientras que tamaños mayores (512 y 1024) lograron reconstrucciones más precisas. Sin embargo, la mejora entre l_size 512 y 1024 fue marginal, indicando que un tamaño intermedio es suficiente para capturar las características relevantes del dataset para nuestra arquitectura elegida sin incurrir en costos computacionales innecesarios.

B. Comparación de arquitecturas

El *autoencoder* completamente convolucional eliminó la capa lineal intermedia, reduciendo la necesidad de compresión extrema en el espacio latente. Aunque esta arquitectura mostró una leve mejora en la reconstrucción, las diferencias no fueron significativas. Esto sugiere que, para tareas con datos relativamente simples como Fashion-MNIST, ambas arquitecturas son comparables en términos de desempeño.

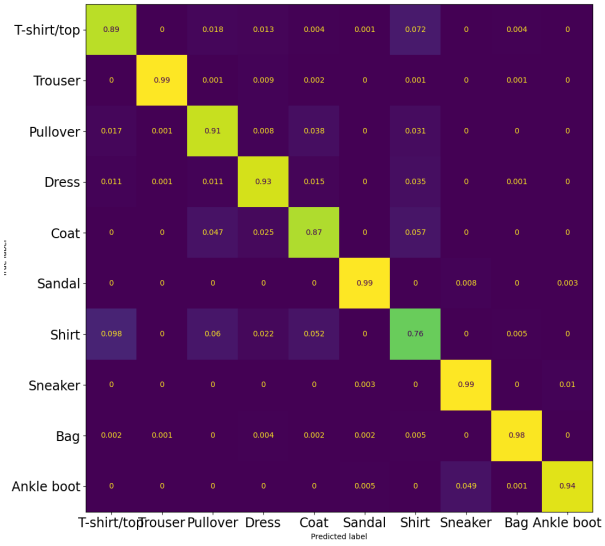

Fig. 10. Matriz de confusión para el clasificador entrenado desde cero (caso 3)

C. Tareas de clasificación

En la etapa de clasificación, los *encoders* pre-entrenados fueron útiles, pero su efectividad dependió del grado de ajuste realizado. El ajuste fino (fine-tuning) permitió mejorar la precisión en el conjunto de validación (90.2 %), superando tanto al entrenamiento solo de la capa clasificadora como al entrenamiento desde cero. Sin embargo, entrenar toda la red desde cero logró resultados competitivos (89 %), posiblemente porque los pesos iniciales del preentrenamiento no estaban perfectamente alineados con la tarea de clasificación.

Las matrices de confusión para los casos 2 y 3 (figuras 9 y 10, respectivamente) muestran que ambos clasificadores convergen en fallos similares, especialmente en la clase *shirt*, que tiende a confundirse con clases como *t-shirt/top*. Este comportamiento sugiere que las características aprendidas en ambas configuraciones no son completamente discriminativas para esta categoría.

Este hallazgo resalta que el pre-entrenamiento en tareas no supervisadas no siempre garantiza un mejor desempeño en tareas supervisadas, esto es una limitación de nuestra arquitectura actual del *encoder*. Junto con lo que se puede observar en la fig 5, deducimos que nuestra arquitectura no es optima y puede ser modificada en busca de una red que logre aprender las características necesarias para una correcta reconstrucción y que su reutilización sea útil para problemas de clasificación.

V. CONCLUSIONES

En este trabajo, analizamos y comparamos la importancia de la dimensión del espacio latente en autoencoders convolucionales. Observamos que un espacio latente muy pequeño limita la capacidad de reconstrucción, mientras que tamaños excesivamente grandes incrementan el costo computacional sin mejoras significativas en la calidad de las reconstrucciones. Esto sugiere que un tamaño intermedio, como $l\_size = 512$, es adecuado para equilibrar la capacidad de aprendizaje y la eficiencia computacional en tareas de reconstrucción.

Los resultados de los experimentos de clasificación mostraron que el ajuste fino del encoder pre-entrenado es una estrategia efectiva para mejorar la capacidad de generalización, alcanzando precisiones superiores al entrenamiento solo de la capa clasificadora. Sin embargo, también se encontró que entrenar toda la red desde cero puede generar resultados competitivos, lo que indica que, en algunos casos, el pre-entrenamiento no siempre es el mejor enfoque, especialmente cuando las características aprendidas no son directamente útiles para la tarea de clasificación. Siendo este el caso de nuestra arquitectura elegida, de lo que concluimos que el diseño inicial podría ser mejorado y optimizado.

En conclusión, este trabajo demuestra que tanto el tamaño del espacio latente como la estrategia de pre-entrenamiento juegan un papel crucial en el desempeño de los modelos. Las arquitecturas y configuraciones adecuadas deben ser seleccionadas de acuerdo con la tarea específica y el conjunto de datos. Futuros trabajos podrían explorar el reemplazo de *MaxPooling* por operaciones con *stride* > 1 para mejorar la eficiencia y robustez de los modelos, así como investigar arquitecturas más avanzadas. En busca de redes mas robustas y versátiles para así ampliar las aplicaciones y efectividad de los modelos.

VI. AGRADECIMIENTOS

A la cátedra de Redes Neuronales por las clases claras y apoyo constante durante toda la cursada, también a FAMAFyC por brindar un espacio de aprendizaje de calidad y de facil acceso.

* adolfo.banchio@mi.unc.edu.ar

[1] MNIST, Fasgion-mnist (2024).
[2] Dave Bergmann, ¿que es un autocodificador? (2023).