Autoencoder y clasificador convolucional de Fashion-MNIST con pre-entrenamiento

Brunello, Florencia Luciana*
Facultad de Matemática, Astronomía, Física y Computación
Universidad Nacional de Córdoba, Ciudad Universitaria, 5000 Córdoba, Argentina
(Dated: 30 de noviembre de 2025)

En este trabajo se implementaron y analizaron un autoencoder convolucional profundo para la reconstrucción de imágenes del conjunto Fashion-MNIST y un clasificador convolucional profundo para su reconocimiento. Se exploraron múltiples arquitecturas de autoencoders, variando hiperparámetros como el tamaño de las capas y la tasa de aprendizaje, seleccionando el modelo con mejor desempeño. El codificador resultante se reutilizó como bloque inicial de un clasificador, sobre el cual también se evaluaron distintas configuraciones y niveles de dropout. Se compararon varios esquemas de entrenamiento: clasificador con codificador pre-entrenado, entrenamiento completo sin pre-entrenamiento, *fine-tuning* con codificador pre-entrenado y entrenamiento del clasificador sin pre-entrenamiento del codificador. Los resultados muestran que el pre-entrenamiento puede mejorar la estabilidad y acelerar la convergencia, aunque el *fine-tuning* alcanza el mejor rendimiento global.

I. INTRODUCCIÓN

Los autoencoders convolucionales se han consolidado como una herramienta central en el aprendizaje no supervisado, gracias a su capacidad para aprender representaciones latentes compactas que preservan la estructura esencial de los datos, especialmente en dominios con organización espacial como las imágenes [1]. En este trabajo se utiliza el conjunto de datos Fashion-MNIST que incluye imágenes de 28×28 píxeles en escala de grises de prendas de vestir distribuidas en diez categorías [3].

El objetivo de este trabajo es evaluar cómo distintas configuraciones influyen en la calidad de reconstrucción y en la extracción de características. A partir de estas representaciones, se estudia además el impacto de reutilizar el codificador como parte de una red clasificadora, comparando tres estrategias de entrenamiento: preentrenamiento, *fine-tuning* y entrenamiento desde cero. Este análisis permite examinar cómo cada enfoque afecta la precisión, la estabilidad del aprendizaje y la capacidad de generalización del clasificador.

II. TEORÍA

Un autoencoder convolucional es un modelo de aprendizaje no supervisado que utiliza capas convolucionales para aprender representaciones compactas y estructuralmente significativas de datos con organización espacial, como imágenes. El codificador aplica filtros convolucionales que extraen características locales y reducen progresivamente la dimensionalidad, mientras que el decodificador invierte este proceso mediante convoluciones transpuestas para reconstruir la entrada original [2].

En este trabajo se implementó un autoencoder convolucional profundo compuesto por dos módulos: un módulo codificador y un módulo decodificador. Con el objetivo de evaluar cómo la complejidad del modelo influye en el rendimiento del mismo, se realizaron tres experimentos variando el tamaño y la cantidad de capas intermedias de cada módulo. Estas modificaciones requirieron ajustar el tamaño y las características de los kernels, junto con los parámetros de stride y padding.

	Exp	in_chans	out_chans	kernel_size	stride	padding
Conv 1	1	1	32	3	1	1
	2	1	32	3	1	1
	3	1	16	3	1	1
Conv 2	1	32	64	3	1	1
	2	32	64	3	1	1
	3	16	32	3	1	1
Conv 3	1	–	–	–	–	–
	2	64	128	3	1	1
	3	–	–	–	–	–

	Exp	in_chans	out_chans	kernel_size	stride	padding
ConvT 1	1	64	32	2	2	0
	2	128	64	3	2	0
	3	32	16	2	2	0
ConvT 2	1	32	1	2	2	0
	2	64	32	2	2	0
	3	16	1	2	2	0
ConvT 3	1	–	–	–	–	–
	2	32	1	2	2	0
	3	–	–	–	–	–

Tabla 1. Hiperparámetros del codificador (arriba) y del decodificador (abajo) utilizados en los experimentos 1, 2 y 3.

La Tabla 1 presenta los parámetros específicos de los codificadores y decodificadores empleados en los experimentos 1, 2 y 3. En todas las variantes se aplicó ReLU después de cada convolución para incorporar no linealidad, *MaxPool* 2×2 en el codificador para reducir progresivamente la resolución espacial y un *dropout* de 0.2 como técnica de regularización. En la etapa final del decodificador se utilizó la función de activación *sigmoid*.

En esta primera etapa, todos los modelos se entrenaron con el Error Cuadrático Medio (MSE) como función de costo y el optimizador Adam configurado con una tasa de aprendizaje de 1×10^{-3} y un *batch size* de 100. La configuración con mejor desempeño se utilizó luego en una instancia siguiente, donde se evaluaron tres valores de *learning rate* (1×10^{-4}, 5×10^{-4} y 1×10^{-3}).

Las pruebas iniciales mostraron que la curva de pérdida

de validación quedaba por debajo de la de entrenamiento, probablemente debido a la diferencia de tamaño de los conjuntos. Por ello, para obtener una evaluación más consistente del autoencoder, se utilizó un subconjunto fijo de 10.000 muestras del entrenamiento y el conjunto de validación de Fashion-MNIST por defecto, también de 10.000 muestras, de modo que ambas curvas se calcularan sobre particiones equivalentes.

En segunda instancia se implementó un clasificador convolucional profundo compuesto por dos módulos: un decodificador (obtenido del autoencoder previamente entrenado) y un clasificador. Para ello se analizaron distintas arquitecturas, variando tanto el número de capas ocultas como la cantidad de neuronas por capa. Las configuraciones evaluadas se presentan en la Tabla 2.

Exp	in_features	n1	n2
1	$64 \times 7 \times 7$	64	32
2	$64 \times 7 \times 7$	128	64
3	$64 \times 7 \times 7$	512	256

Exp	in_features	n1	n2	n3
1	$64 \times 7 \times 7$	128	64	32
2	$64 \times 7 \times 7$	256	128	64
3	$64 \times 7 \times 7$	512	256	128

Tabla 2. Arquitecturas evaluadas para el clasificador de dos capas (arriba) y tres capas (abajo).

Todos los experimentos del clasificador se realizaron utilizando la entropía cruzada como función de pérdida y el optimizador Adam con un *learning rate* de 1×10^{-3}, aplicando descenso por gradiente únicamente sobre los parámetros del módulo clasificador, con un *dropout* de 0.2 y un *batch size* de 100. Posteriormente, se seleccionó la arquitectura con mejor desempeño y se probaron tres valores de *dropout*. Para evaluar el clasificador se emplearon los conjuntos de entrenamiento y validación provistos por defecto por Fashion-MNIST.

Por último, con el objetivo de evaluar la efectividad del pre-entrenamiento se realizaron tres experimentos adicionales utilizando la red clasificadora que obtuvo los mejores resultados, considerando los siguientes esquemas: (1) entrenamiento completo sin pre-entrenamiento, (2) entrenamiento completo (*fine-tuning*) con el codificador pre-entrenado y (3) entrenamiento únicamente del clasificador, sin pre-entrenamiento del codificador.

III. RESULTADOS

En esta sección se presentan los resultados obtenidos a partir de los experimentos realizados tanto con el autoencoder como con el clasificador convolucional profundo, considerando distintos métodos de pre-entrenamiento.

Parte 1: Autoencoder Convolucional Profundo

1. Análisis de Arquitecturas

Los valores de *loss* obtenidos sobre el conjunto de validación para los experimentos 1, 2 y 3, correspondientes a la Tabla 1, luego de 100 épocas de entrenamiento fueron de 0.0025, 0.0040 y 0.0042 respectivamente.

En la Figura 1 puede observarse la evolución de la curva de pérdida para cada arquitectura. De las tres variantes, se seleccionó la correspondiente al experimento 1, ya que alcanzó el menor valor de pérdida, con tiempos de entrenamiento comparables a las otras configuraciones.

Asimismo, se observa que a partir de la época 60 no hay mejoras sustanciales en ninguna de las arquitecturas, por este motivo, y con el fin de evitar cómputo innecesario, para la evaluación de los distintos valores de *learning rate* se consideraron únicamente 60 épocas de entrenamiento.

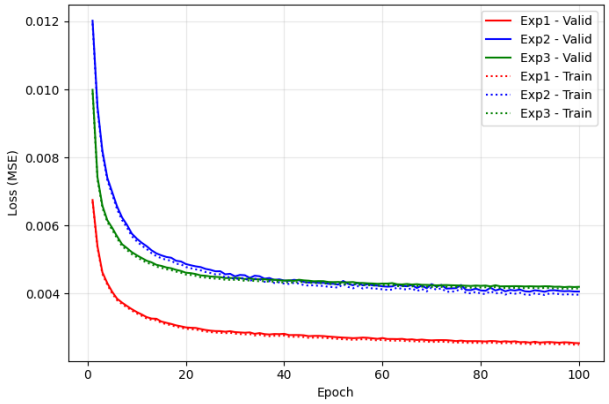

Figura 1. Curvas de *loss* sobre el conjunto de validación para distintas arquitecturas de autoencoders convolucionales.

2. Evaluación del learning rate

Los valores de *loss* obtenidos sobre el conjunto de validación para los valores de *learning rate* 1×10^{-4}, 5×10^{-4} y 1×10^{-3} luego de 60 épocas de entrenamiento fueron de 0.0052, 0.0036 y 0.0032 respectivamente (véase Figura 2).

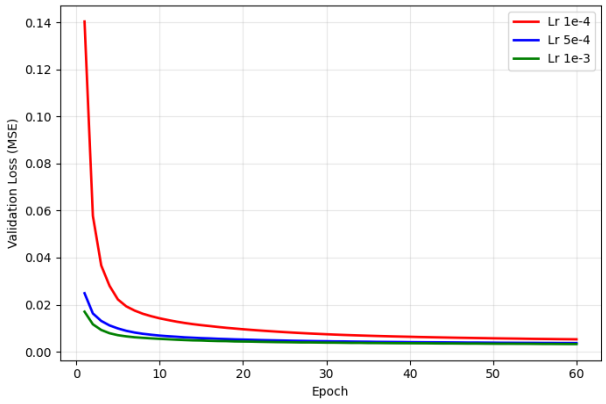

Figura 2. Curvas de *loss* sobre el conjunto de validación para distintos valores de *learning rate* del autoencoder convolucional.

En el gráfico de la Figura 2 se observa que la curva correspondiente al valor 1×10^{-4} presenta una caída inicial abrupta pero luego se estabiliza en un valor final superior al de los otros dos modelos, lo que indica que este paso de actualización es demasiado pequeño. En contraste, los *learning rates* de 5×10^{-4} y 1×10^{-3}, parten de valores iniciales más bajos y alcanzan valores de *loss* menores.

Dado que los tiempos de ejecución fueron muy similares en los tres casos, el valor 1×10^{-3} resulta ser la alternativa más adecuada, ya que combina una convergencia rápida, estable y con el menor valor final de *loss* alcanzado entre las opciones evaluadas.

Parte 2: Clasificador Convolucional Profundo

1. Análisis de Arquitecturas

Para seleccionar la arquitectura con mejor desempeño, se analizaron primero los resultados correspondientes a redes de dos capas, luego los de tres capas, y finalmente se compararon los mejores casos de cada grupo.

Los valores de *loss* obtenidos sobre el conjunto de validación para los experimentos 1, 2 y 3 de dos capas luego de 100 épocas de entrenamiento fueron de 0.2885, 0.2608 y 0.2615 respectivamente (Figura 3). En general, se observan picos que indican cierta inestabilidad en la convergencia. Sin embargo, la curva azul presenta un comportamiento más estable, con menos fluctuaciones y valores de *loss* consistentemente inferiores a los de las otras curvas.

A partir de la época 40 se observa un incremento del *overfitting*, evidenciado por la creciente separación entre las curvas de entrenamiento y validación (destacadas por las líneas verticales azules). Por esta razón, en los experimentos posteriores destinados a evaluar el dropout y analizar el impacto del pre-entrenamiento se decidió utilizar únicamente 40 épocas de entrenamiento.

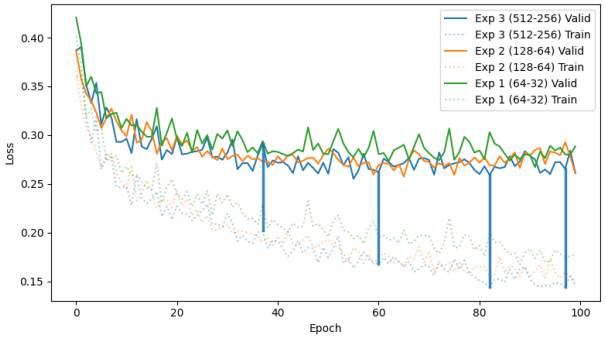

Figura 3. Curvas de *loss* sobre los conjuntos de entrenamiento y validación de los autoencoders convolucionales de dos capas.

En cuanto a la precisión, las arquitecturas de dos capas alcanzaron valores de 0.9022, 0.9091 y 0.9111 en los experimentos 1, 2 y 3, respectivamente. Dado que el experimento 3 obtuvo la mayor precisión y el menor *loss*, con un costo computacional comparable al de los otros dos casos, se seleccionó como la configuración óptima.

Los valores de *loss* obtenidos sobre el conjunto de validación para los experimentos 1, 2 y 3 luego de 100 épocas de entrenamiento fueron de 0.2669, 0.2850 y 0.2662 respectivamente (Figura 4). Aunque se observa *overfitting* a partir de la época 40, la separación entre las curvas de entrenamiento y validación es similar en los tres casos. Esto contrasta con la arquitectura de dos capas, donde la cantidad de neuronas generaba separaciones de distinta magnitud. En cambio, la arquitectura de tres capas muestra un comportamiento más uniforme ante variaciones en la cantidad de neuronas, lo que sugiere que dichos cambios no afectan significativamente la convergencia.

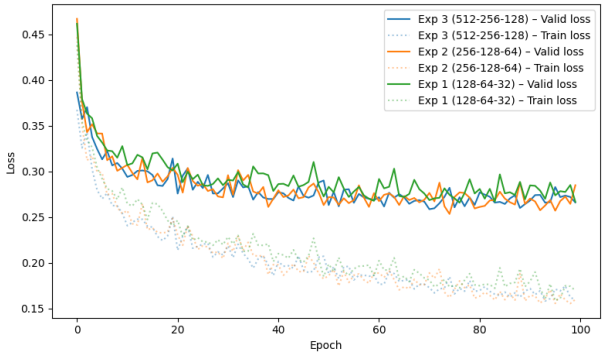

Figura 4. Curvas de *loss* sobre los conjuntos de entrenamiento y validación de los autoencoders convolucionales de tres capas.

Los valores de precisión obtenidos sobre el conjunto de validación para los experimentos 1, 2 y 3 luego de 100 épocas fueron de 0.9092, 0.9053 y 0.9109 respectivamente. Dado que la *loss* mostró un comportamiento similar en los tres casos, la configuración óptima se seleccionó en función de la precisión, eligiendo el experimento 3.

Finalmente, la Figura 5 compara la precisión de los mejores casos de ambas arquitecturas. La red de dos capas alcanza una precisión ligeramente superior y presenta un menor costo computacional debido a su menor profundidad, lo que se traduce en tiempos de entrenamiento más reducidos. Por este motivo, se optó por esta arquitectura.

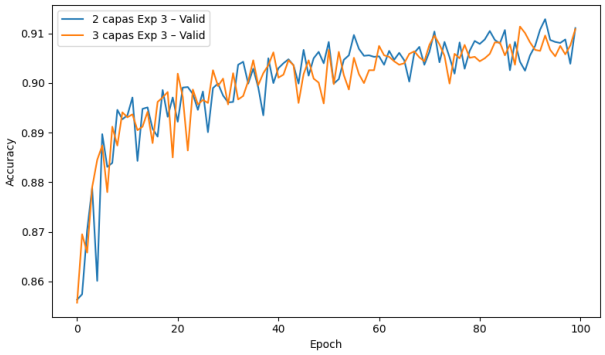

Figura 5. Curvas de *accuracy* sobre el conjunto de validación del clasificador convolucional para distintas arquitecturas.

2. Evaluación del dropout

Los valores de precisión obtenidos sobre el conjunto de validación para los valores de *dropout* de 0.2, 0.4 y 0.6 luego de 40 épocas de entrenamiento fueron de 0.902, 0.8883 y 0.8868 respectivamente.

Como se observa en la Figura 6, el *dropout* de 0.2 presenta una convergencia más estable que los valores superiores. Esto es esperable, ya que *dropouts* demasiado altos tienden a eliminar excesivamente la información durante el entrenamiento, dificultando el aprendizaje y generando oscilaciones más pronunciadas en la curva de validación. En función de estos resultados, se seleccionó el valor de 0.2 como la opción más adecuada.

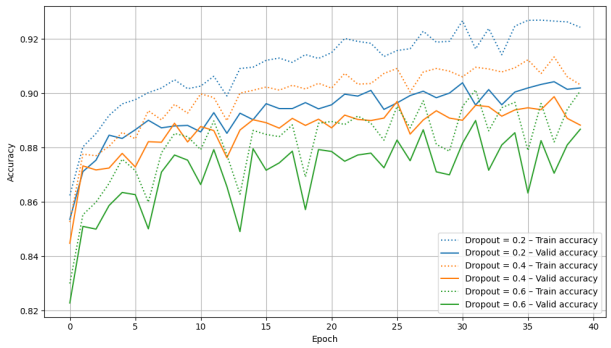

Figura 6. Curvas de *accuracy* sobre el conjunto de validación del clasificador convolucional para distintos valores de *dropout*.

Evaluación de la efectividad del pre-entrenamiento

1. Entrenamiento completo sin pre-entrenamiento

Tras 40 épocas de entrenamiento, el modelo alcanza una precisión elevada del 92,66 % en el conjunto de validación. Sin embargo, al observar el gráfico de la función de pérdida (Figura 7), se aprecia un marcado sobreajuste desde etapas tempranas.

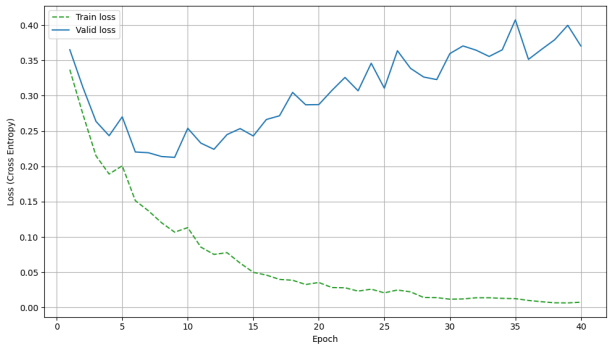

Figura 7. Curvas de *loss* sobre los conjuntos de entrenamiento y validación del caso 1.

Observamos que, aproximadamente a partir de la décima época, la pérdida de entrenamiento continúa disminuyendo mientras que la pérdida de validación comienza a incrementarse, alcanzando valores de 0.9983 y 0.3704 en la época 40. Este comportamiento evidencia que el modelo comienza a memorizar el conjunto de entrenamiento y, en consecuencia, pierde capacidad de generalización.

2. Entrenamiento completo (o *fine-tunning*) con codificador pre-entrenado

El valor final de precisión alcanzado en este experimento luego de 40 épocas de entrenamiento es del 92,7 %, apenas superior al obtenido en el caso 1. Considerando que el *fine-tuning* completo implica un costo computacional mayor, esta estrategia no resulta ventajosa.

Como se observa en la Figura 8, el sobreajuste aparece desde las primeras épocas: la loss de validación aumenta rápidamente mientras que la de entrenamiento desciende hasta valores muy bajos, alcanzando los valores 0.9990 y 0.4018 hacia la época 40.

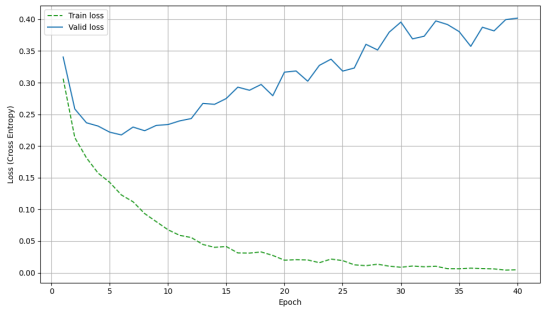

Figura 8. Curvas de *loss* sobre los conjuntos de entrenamiento y validación del caso 2.

La Figura 9 muestra la matriz de confusión correspondiente al caso dos sobre el conjunto de validación.

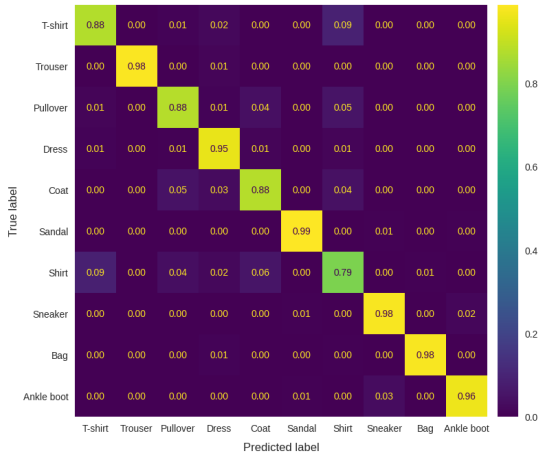

Figura 9. Matriz de confusión del experimento *fine-tuning* con codificador pre-entrenado sobre el conjunto de validación.

En esta matriz se observa que las categorías T-shirt, Shirt y Coat son las que presentan mayor nivel de confusión, lo cual se aprecia en los tonos más claros fuera de la diagonal. Este patrón de errores es coherente con el sobreajuste previamente observado y refleja la dificultad del modelo para distinguir adecuadamente entre clases visualmente similares. Este comportamiento probablemente esté asociado a la simplicidad del conjunto Fashion-MNIST, que facilita que el modelo memorice las muestras de entrenamiento en lugar de aprender representaciones más generalizables.

3. Entrenamiento solo del clasificador sin pre-entrenamiento del codificador

Los valores de precisión obtenidos sobre los conjuntos de entrenamiento y validación luego de 40 épocas fueron de 0.8862 y 0.8699 respectivamente. En este caso no se observa sobreajuste ya que ambas curvas avanzan de manera paralela a lo largo de todas las épocas (Figura 10). Esto refleja un comportamiento estable y una buena capacidad de generalización del modelo.

Si bien la precisión final en validación (86,99 %) es menor que la obtenida en los experimentos 1 y 2, este enfoque demuestra ser menos propenso a memorizar los datos. Al ajustar únicamente los parámetros del codificador durante el entrenamiento, el modelo reduce su capacidad de adaptarse a los ejemplos del conjunto *train*. Por ello, este método constituye una alternativa eficiente cuando se priorizan la estabilidad y la reducción del sobreajuste.

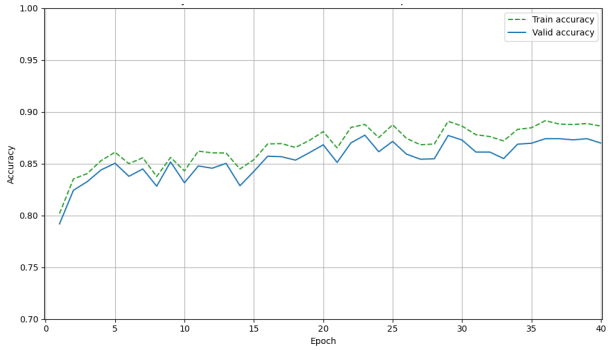

Figura 10. Curvas de *accuracy* sobre los conjuntos de entrenamiento y validación del caso 3.

IV. CONCLUSIONES

A lo largo de las distintas variantes evaluadas, tanto para el autoencoder como para el clasificador, pudimos observar que incrementar la profundidad de los mismos no garantiza mejoras de rendimiento. Por el contrario, redes más complejas implican un costo computacional mayor y tienden a sobreajustar con mayor facilidad. En consecuencia, cuando una arquitectura más simple alcanza niveles de desempeño comparables, resulta preferible priorizar esa opción.

En cuanto a la efectividad del pre-entrenamiento, los experimentos muestran que tanto el entrenamiento completo (caso 1) como el *fine-tuning* con codificador pre-entrenado (caso 2) alcanzan precisiones similares del 92.66 % y 92.7 % respectivamente, pero ambos presentan un gran sobreajuste desde etapas tempranas del entrenamiento. El *fine-tuning*, además, exige mayores recursos computacionales y no se traduce en una mejora en la capacidad de generalización, por lo que su costo adicional no parece justificado para este dataset.

Por su parte, el entrenamiento solo del módulo clasificador prácticamente no muestra sobreajuste dado que las curvas de entrenamiento y validación evolucionan de manera paralela. Sin embargo, esta estabilidad se logra a costa de una precisión menor del 86,99 %. Un rendimiento similar se observa al pre-entrenar el autoencoder completo y luego el decoder, donde se alcanza una precisión del 90,02 %, con curvas estables y pocos picos. En este sentido, los métodos 0 y 3 muestran un mejor equilibrio entre estabilidad y costo computacional, aunque con precisiones inferiores a los métodos 1 y 2.

En conclusión, los resultados muestran que tanto la complejidad arquitectónica como las decisiones de pre-entrenamiento influyen directamente en la precisión final, el grado de sobreajuste y la estabilidad del entrenamiento. Dado que Fashion-MNIST es un dataset con imágenes relativamente simples, modelos excesivamente profundos no aportan mejoras significativas sino que, por el contrario, tienden a sobreajustar. En términos de desempeño, si el objetivo es maximizar la precisión, los casos 1 y 2 alcanzan los mejores valores mientras que, si se prioriza un modelo más estable y con menor riesgo de sobreajuste, los casos 0 y 3 ofrecen un equilibrio más adecuado. En cualquier caso, es recomendable evitar arquitecturas innecesariamente profundas y limitar el número de épocas para reducir tanto el sobreajuste como el consumo de recursos computacionales.

V. REFERENCIAS

* florenciabrunello@unc.edu.ar

[1] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

[2] Nielsen, M. A. (2016). *Deep learning*. Determination Press.

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