Autoencoder Convolucional y Transfer Learning para la Clasificación de Imágenes en el Conjunto de Datos Fashion-MNIST con PyTorch

María Victoria Mascaró de Gárate*
Facultad de Ciencias Económicas (FCE), Universidad Nacional de Córdoba, Ciudad Universitaria, 5000 Córdoba, Argentina
(Dated: 23 de febrero de 2026)

Se entrenó un autoencoder convolucional profundo para aprender representaciones latentes comprimidas de imágenes del conjunto Fashion-MNIST. Posteriormente, el encoder aprendido se reutilizó como extractor de características en una tarea de clasificación supervisada. Se evaluaron distintos esquemas de transferencia de conocimiento: entrenamiento desde cero, *fine-tuning* y encoders congelados (preentrenados y aleatorios). Los modelos fueron implementados en PyTorch y comparados mediante curvas de pérdida, precisión y matrices de confusión sobre conjuntos de entrenamiento y validación.

I. INTRODUCCIÓN

El aprendizaje de representaciones es un componente central del aprendizaje profundo moderno. En particular, el preentrenamiento no supervisado permite extraer estructuras generales de los datos que luego pueden reutilizarse en tareas supervisadas específicas mediante técnicas de transferencia de conocimiento.

El objetivo de este trabajo es analizar el impacto del preentrenamiento de un autoencoder convolucional sobre el desempeño de un modelo de clasificación de imágenes. Para ello, se entrena primero un autoencoder para reconstrucción de imágenes y posteriormente su encoder se reutiliza como extractor de características dentro de un clasificador supervisado.

Se comparan distintos esquemas de entrenamiento que difieren en la inicialización y el grado de actualización del encoder, con el fin de evaluar en qué medida las representaciones aprendidas para reconstrucción resultan útiles para clasificación.

II. TEORÍA Y MÉTODOS

A. Datos y preprocesamiento

Se utilizó el conjunto Fashion-MNIST, compuesto por imágenes en escala de grises de tamaño 28×28 pertenecientes a diez categorías de prendas de vestir.

Los valores de los píxeles se normalizaron con media 0.5 y desviación estándar 0.5, reescalando el rango al intervalo $[-1, 1]$, siguiendo prácticas estándar de preprocesamiento en redes neuronales profundas [1, p. 453]. Esta normalización mejora la estabilidad numérica del entrenamiento y es consistente con el uso de activación Tanh[1, p. 195] en la capa de salida del autoencoder.

B. Entrenamiento y funciones de pérdida

El autoencoder se entrenó minimizando el **Error Cuadrático Medio** (*MSE, Mean Squared Error*) entre la imagen original y la reconstruida.

Los modelos clasificadores se entrenaron mediante **Entropía Cruzada** (*Cross Entropy Loss*). En PyTorch, la función `CrossEntropyLoss` incorpora internamente la operación `LogSoftmax`[1, p. 184–187], por lo que no es necesario aplicarla explícitamente en la capa de salida del modelo.

En todos los experimentos se utilizó el **optimizador Adam**, con una **tasa de aprendizaje** (*learning rate*) de 10^{-3} y un **tamaño de lote** (*batch size*) de 100.

C. Autoencoder convolucional

La arquitectura del encoder está compuesta por tres bloques convolucionales secuenciales. El primer bloque aplica una convolución 2D con 32 filtros de tamaño 3×3 y *padding* 1, seguida de activación ReLU[1, p. 193-194], max pooling 2×2 y dropout $p = 0{,}2$, reduciendo la resolución espacial de $(1, 28, 28)$ a $(32, 14, 14)$.

El segundo bloque aplica una convolución con 64 filtros de tamaño 3×3, seguida de ReLU, max pooling y dropout, reduciendo la representación a $(64, 7, 7)$.

Finalmente, una tercera convolución con $n = 64$ filtros de tamaño 3×3 produce la representación latente del modelo, de dimensión $(64, 7, 7)$, que constituye el **cuello de botella** (*bottleneck*) del autoencoder.

En conjunto, estas transformaciones reducen progresivamente la dimensionalidad espacial mientras incrementan la profundidad de canales, permitiendo al modelo capturar patrones jerárquicos de la estructura visual de las imágenes.

Para un mayor entendimiento del trade-off entre compresión y capacidad expresiva determinado por el tamaño del *bottleneck*, véase [2, pp. 386–387].

El decoder expande progresivamente la representación latente mediante dos capas de convolución traspuesta con factor de escala 2, seguidas de una convolución final que recupera el canal original de la imagen.

La última capa del decoder utiliza activación Tanh, coherente con la normalización de las imágenes al rango $[-1, 1]$.

El encoder aprendido define un espacio latente que posteriormente se reutiliza como extractor de características para la tarea de clasificación.

Para una descripción detallada de los hiperparámetros de las capas convolucionales en PyTorch, véase la documentación oficial [3].

Asimismo, para el análisis teórico de la aritmética de convoluciones (efecto de *stride*, *padding* y tamaño de kernel sobre las dimensiones espaciales), véase [4].

D. Clasificación y transferencia de conocimiento

El clasificador está compuesto por el encoder convolucional seguido de un perceptrón multicapa compuesto por una capa densa oculta con activación ReLU y dropout, y una capa densa de salida que produce las predicciones de clase.

Se evaluaron cuatro esquemas de entrenamiento:

1. Entrenamiento completo desde cero (encoder y clasificador aleatorios).
2. *Fine-tuning* con encoder preentrenado.
3. Encoder aleatorio congelado.
4. Encoder preentrenado congelado (*transfer learning*).

El **aprendizaje por transferencia** (*transfer learning*) consiste en reutilizar representaciones aprendidas en un problema o conjunto de datos fuente para mejorar el aprendizaje en una nueva tarea relacionada, especialmente cuando los datos disponibles son limitados. Una de las estrategias más comunes de *transfer learning* es el ***fine-tuning***, que adapta un modelo preentrenado reemplazando su capa de salida y ajustando sus parámetros al nuevo dominio, generalmente con tasas de aprendizaje diferenciadas. Este enfoque permite aprovechar características generales previamente aprendidas y especializarlas para la tarea objetivo. [5, pp. 622–628] [1, pp. 536–540]

Los cuatro esquemas experimentales pueden interpretarse dentro del marco de *transfer learning* y fine-tuning. El entrenamiento completo sin preentrenamiento corresponde al enfoque clásico de aprendizaje desde cero, sin transferencia de conocimiento previa. El entrenamiento completo con encoder preentrenado implementa fine-tuning, donde se reutilizan representaciones aprendidas y se ajustan conjuntamente con el clasificador para adaptarlas a la nueva tarea. Entrenar solo el clasificador con encoder aleatorio congelado evalúa la capacidad de un extractor de características no adaptado, sin transferencia efectiva de conocimiento. Finalmente, entrenar solo el clasificador con encoder preentrenado congelado representa transferencia de características fija, donde el conocimiento aprendido previamente se reutiliza sin modificación durante la tarea supervisada.

Resumiendo, estos esquemas permiten evaluar el efecto del preentrenamiento y del grado de adaptabilidad del encoder a la tarea supervisada.

III. RESULTADOS

Desde el punto de vista cuantitativo, los modelos que permiten entrenar el encoder durante la tarea supervisada —entrenamiento desde cero (Exp.1) y *fine-tuning* (Exp.2)— alcanzan el mejor desempeño global. En ambos casos la Loss en el conjunto de entrenamiento desciende por debajo de 0.1, mientras que la Loss en el conjunto de validación alcanza un mínimo cercano a 0.22 aproximadamente entre las épocas 10 y 20. El Accuracy de entrenamiento se aproxima a 1, y el Accuracy en el conjunto de validación se estabiliza alrededor de 0.92–0.93. Sin embargo, ambos modelos muestran evidencia clara de **sobreajuste** (*overfitting*), ya que pasada la época 20 la Loss en el conjunto de validación comienza a aumentar, mientras que la Loss en el conjunto de entrenamiento continúa disminuyendo.

El modelo con encoder preentrenado congelado (Exp.4) presenta un desempeño intermedio, con Loss de validación en el rango 0.27–0.30 y Accuracy cercano a 0.90. En contraste, el modelo con encoder aleatorio congelado (Exp.3) obtiene el peor rendimiento, con Loss de validación alrededor de 0.4 y Accuracy cercano a 0.85.

En términos de dinámica de entrenamiento, los modelos que entrenan el encoder muestran mayor capacidad de ajuste a la tarea de clasificación, pero también mayor propensión al sobreajuste. Por el contrario, los modelos con encoder congelado presentan curvas de entrenamiento y validación más estables y sin divergencias evidentes en el rango de épocas analizado, aunque con menor desempeño final. La ausencia de sobreajuste abre la posibilidad de que estos modelos puedan beneficiarse de un entrenamiento más prolongado. Extender la cantidad de épocas permitiría evaluar si su desempeño continúa mejorando progresivamente o si, por el contrario, ya han alcanzado una etapa de estancamiento en el aprendizaje.

Las matrices de confusión presentadas en la Figura 1, evaluadas sobre el conjunto de validación tras 50 épocas, están normalizadas por clase real, de modo que cada fila suma 1 y representa la distribución de predicciones condicionada al rótulo verdadero. Esto permite interpretar los valores como probabilidades de clasificación por clase.

Todas las matrices muestran un patrón de error consistente entre configuraciones: La mayor confusión se produce entre las clases *T-Shirt* y *Shirt*, lo cual resulta esperable dada su similitud visual.

Comparativamente, el modelo con *fine-tuning* presenta la mayor concentración de probabilidad sobre la diagonal principal, indicando el mejor desempeño global. Asi-

mismo, dado que los modelos que entrenan el encoder presentan sobreajuste a partir de las 10–20 épocas, es razonable esperar que su desempeño pudiera mejorar si se evaluaran en el punto de mejor validación mediante *early stopping*.

IV. DISCUSIÓN Y CONCLUSIONES

El autoencoder convolucional aprende representaciones comprimidas que preservan la estructura visual de los artículos de Zalando del dataset Fashion-MNIST. Al reutilizar el encoder como extractor de características, el clasificador converge más rápidamente.

Esto confirma que el aprendizaje no supervisado permite capturar regularidades estructurales de los datos que resultan útiles para tareas supervisadas posteriores. Sin embargo, el congelamiento total del encoder limita su capacidad de adaptación a la tarea de clasificación, lo que explica el menor desempeño observado respecto de los modelos que permiten ajustar toda la red.

Los resultados también muestran que el preentrenamiento no garantiza automáticamente el mejor rendimiento. Su efectividad depende del grado de alineación entre la tarea de preentrenamiento y la tarea final. En este caso, el autoencoder fue optimizado para reconstrucción de imágenes, lo que favorece la preservación de información visual general, pero no necesariamente la separabilidad entre clases.

En conjunto, los experimentos indican que el preentrenamiento proporciona representaciones informativas y estables, pero *la adaptación supervisada del encoder resulta fundamental* cuando el objetivo es maximizar la capacidad discriminativa del modelo.

Estos hallazgos son consistentes con el enfoque dominante en aprendizaje profundo moderno, donde primero se aprenden representaciones generales de los datos y luego se especializan mediante ajuste supervisado para tareas específicas.

El *transfer learning* resulta especialmente útil en datasets pequeños, donde reutilizar representaciones aprendidas mejora la estabilidad y acelera la convergencia. Entrenar desde cero suele recomendarse solo con grandes volúmenes de datos etiquetados, del orden de miles de ejemplos por clase [2, p. 60].

Este principio es ampliamente explotado en modelos modernos de visión artificial y procesamiento del lenguaje, donde se entrena primero una representación general y luego se la especializa para tareas concretas.

En síntesis, el preentrenamiento no garantiza automáticamente el mejor modelo, pero puede constituir una estrategia valiosa dentro de un enfoque más amplio de diseño y entrenamiento de redes neuronales profundas. Su utilidad práctica depende del equilibrio entre la calidad de las representaciones aprendidas previamente y la necesidad de adaptarlas a la tarea específica de interés.

* mvictoriamascaro@gmail.com

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] Valliappa Lakshmanan, Martin Görner, and Ryan Gillard. *Practical Machine Learning for Computer Vision*. O'Reilly Media, 2021.

[3] PyTorch Documentation. torch.nn.conv2d. `https://docs.pytorch.org/docs/stable/generated/torch.nn.Conv2d.html`, 2024. Accessed: 2026-02-22.

[4] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. `https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md`, 2018. Accessed: 2026-02-22.

[5] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023.

[6] Rajalingappaa Shanmugamani. *Deep Learning for Computer Vision*. Packt Publishing, 2018.

Modelo	Train Loss	Valid Loss	Train Acc.	Valid Acc.	Gap Acc. (%)
Fine-tuning	0.0172	0.3154	0.9955	0.9276	6.79
Scratch	0.0248	0.3067	0.9922	0.9260	6.62
Pretrained frozen	0.1502	0.2913	0.9420	0.9041	3.79
Random frozen	0.3752	0.4125	0.8575	0.8469	1.06

Tabla I: Resultados de los modelos de clasificación tras 50 épocas de entrenamiento. Se reportan Loss y Accuracy, sobre los conjuntos de entrenamiento y validación. El *gap de accuracy* corresponde a la diferencia entre precisión de entrenamiento y validación, utilizado como indicador aproximado del grado de sobreajuste.

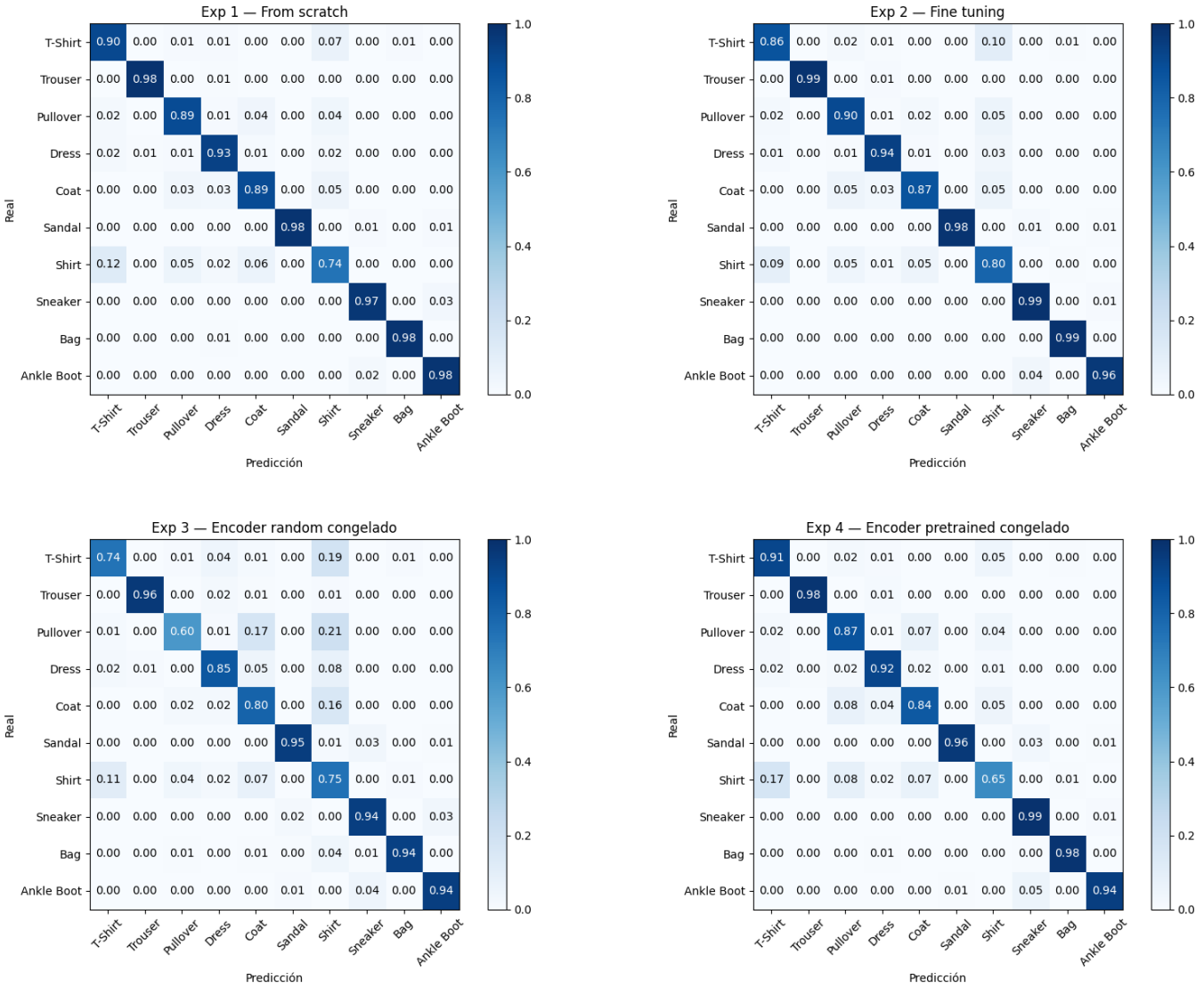

Figura 1: **Matrices de confusión normalizadas para los cuatro experimentos analizados.** Cada panel corresponde a un experimento distinto evaluado sobre el conjunto de validación tras 50 épocas. Las matrices están normalizadas por clase real (filas), de modo que cada fila suma 1 e indica la distribución de predicciones condicionada al rótulo verdadero. La mayor concentración sobre la diagonal indica mejor desempeño de clasificación.

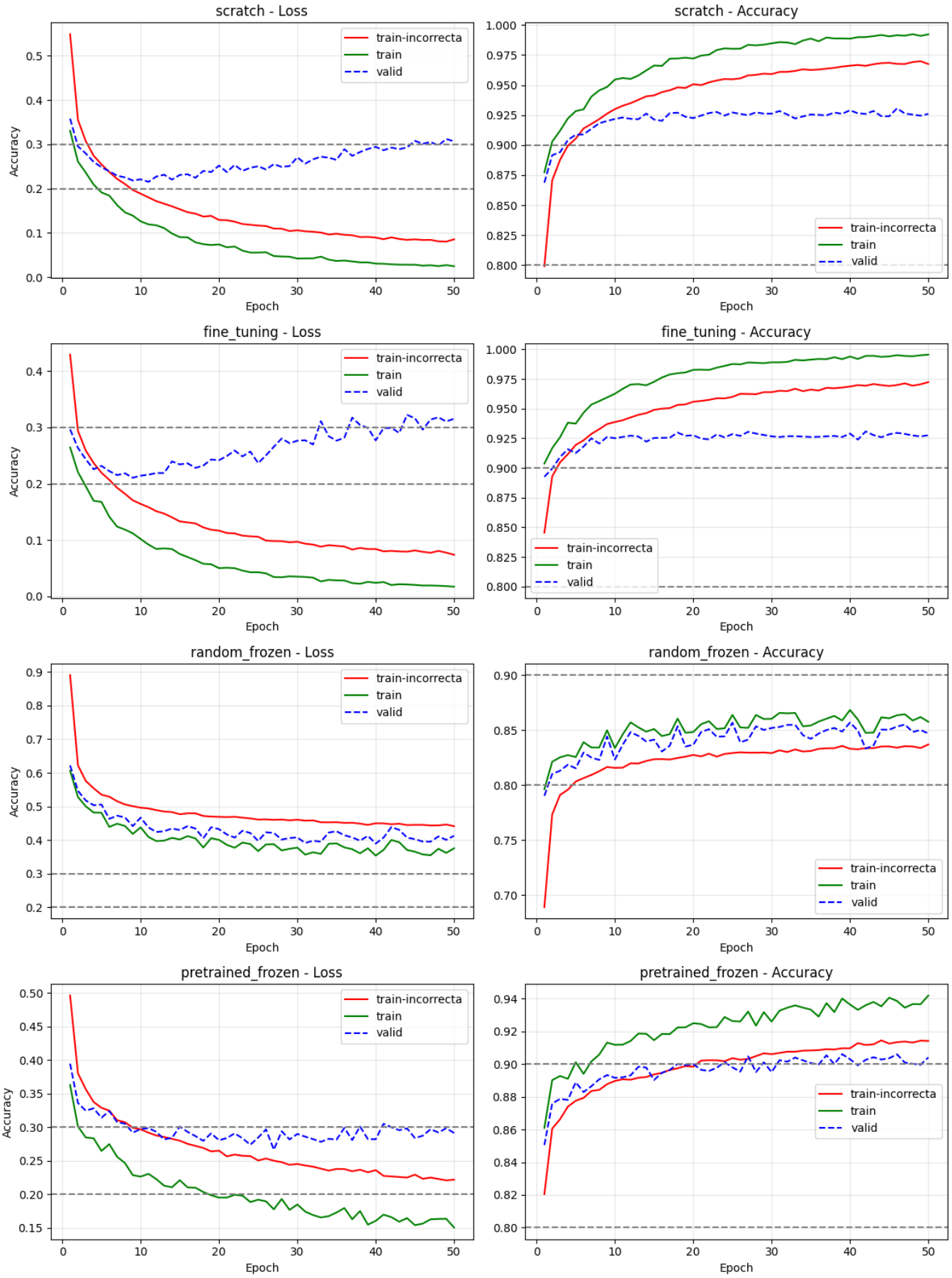

Figura 2: **Evolución de la función de pérdida (Cross Entropy Loss) y precisión (Accuracy).** Se muestran las curvas correspondientes a los cuatro experimentos de clasificación. Se comparan tres curvas: entrenamiento activo (rojo), entrenamiento sin actualización de pesos (verde) y validación (azul discontinua).