ChatGPT

Informe sobre Transfer Learning con EfficientNetV1B2 en ArtBench-10

Introducción

El **aprendizaje por transferencia** (*transfer learning*) es una técnica que reaprovecha el conocimiento de un modelo previamente entrenado en una tarea fuente para aplicarlo a una tarea objetivo relacionada [1] [2]. En el contexto de visión por computador, esto típicamente implica tomar una **red neuronal preentrenada** (p. ej., en ImageNet) y adaptarla a un nuevo conjunto de datos, evitando entrenar desde cero. Esto es especialmente útil cuando el nuevo dataset es relativamente pequeño o diferente, ya que el modelo preentrenado ya ha aprendido a extraer características visuales generales (bordes, texturas, formas) que pueden ser reutilizadas [3] [4]. Existen dos estrategias principales de transferencia: **extracción de características** (*feature extraction*), donde la red preentrenada se usa congelada como extractor de *embeddings*, y **fine-tuning**, donde se ajustan los pesos de parte o la totalidad de la red para especializarla al nuevo dominio [5] [6].

En este informe analizamos el uso de la red EfficientNet V1 B2 en un experimento de clasificación de obras de arte con el dataset **ArtBench-10** (60.000 imágenes de 256×256 píxeles de 10 estilos artísticos distintos, incluyendo impresionismo, expresionismo, etc.). El objetivo fue emplear EfficientNetB2 como **backbone** extractor de características visuales (*features*), aprovechando sus pesos preentrenados en ImageNet y ajustándolos al dominio artístico mediante *fine-tuning*. A continuación, se describen los fundamentos teóricos del transfer learning aplicado, la justificación de EfficientNetB2 como modelo base, la importancia de trabajar con *embeddings* vectoriales de imágenes, y un análisis crítico de los resultados obtenidos en tres fases de entrenamiento (red congelada vs. ajustada). Finalmente, se discuten posibles razones por las cuales las fases de *fine-tuning* (fases 2 y 3) no mejoraron el desempeño, considerando factores como la tasa de aprendizaje, el desajuste de dominio (ImageNet vs. arte) y la diferencia entre optimizar para clasificación versus para obtener representaciones útiles en otras tareas.

Fundamentos de Transfer Learning y Fine-Tuning

El éxito del transfer learning radica en que las primeras capas de redes profundas aprenden características de bajo nivel bastante generales (bordes, texturas), mientras que las capas superiores capturan patrones más específicos del conjunto de datos original [7] [8]. Al transferir el modelo, las **capas congeladas** retienen esas características universales aprendidas, y solo se entrenan nuevas capas (o se afinan las últimas existentes) para la nueva tarea. Esto puede acelerar el entrenamiento, requerir menos datos y reducir el sobreajuste [9] [10].

Extracción de características: en esta modalidad, se **congela la red preentrenada** (no se modifican sus pesos) y simplemente se añade y entrena un clasificador nuevo al final [11]. La red actúa como un extractor fijo de atributos; por ejemplo, dado un cuadro, EfficientNetB2 produce un vector de características (*embedding*) que luego se pasa a una capa densa entrenada para predecir el estilo de la obra. Las ventajas de esta estrategia incluyen menor riesgo de sobreajuste (muchos menos parámetros ajustables) y menores

requerimientos de datos y cómputo [9]. Sin embargo, su **contras** son una **adaptabilidad limitada**: los *features* aprendidos no se ajustan a particularidades nuevas del dominio, lo que puede degradar el rendimiento si la tarea objetivo difiere mucho de la original [12]. En nuestro caso, usar EfficientNetB2 congelada significa confiar en que las características aprendidas en fotos cotidianas (ImageNet) serán relevantes para distinguir estilos pictóricos – lo cual podría no capturar completamente, por ejemplo, patrones de pinceladas o paletas de colores propias del arte.

Fine-tuning: esta estrategia va más allá, permitiendo **descongelar parte o toda la red** preentrenada para seguir entrenándola en el nuevo dataset [13]. Típicamente se mantienen congeladas las capas iniciales (que capturan rasgos muy generales como bordes) y se *fine-tunean* las capas finales (más especializadas) junto con el nuevo clasificador, usando una tasa de aprendizaje baja para no perturbar drásticamente los pesos preentrenados [14]. De esta manera, el modelo ajusta sus filtros para características más alineadas al nuevo dominio. El fine-tuning suele lograr **mejor desempeño** que la extracción pura de características, dado suficiente tamaño de datos, porque **adapta los descriptores a la tarea específica** [15]. Permite mayor flexibilidad y especialización, aunque a costa de mayor riesgo de sobreajuste si el dataset es pequeño, mayor costo computacional y necesidad de calibrar hiperparámetros cuidadosamente (p.ej., qué capas descongelar, learning rate, etc.) [16].

¿Cuándo usar cada enfoque? La elección depende de la similitud y tamaño del nuevo dataset [17]. En términos generales:

- Con un conjunto de datos **pequeño y similar** al original, conviene congelar la mayor parte del modelo y solo entrenar las capas finales, para evitar sobreajuste [18].
- Si el dataset es **grande y similar**, se puede descongelar más capas, ya que hay suficiente información para re-entrenar parte sustancial de la red [19].
- Con un dataset **pequeño pero de dominio muy distinto**, a veces se congelan las capas iniciales y se fine-tunean algunas intermedias o finales para capturar las nuevas características específicas [20].
- Para un dataset **grande y de dominio diferente**, suele recomendarse el **fine-tuning completo de la red** (o de la mayor parte de capas) [20]. Nuestro caso (ArtBench-10: 50.000 imágenes de entrenamiento, dominio artístico vs. natural) cae en esta última categoría: es un volumen considerable de datos y un dominio visual distinto, por lo que sería razonable intentar un fine-tuning amplio de EfficientNetB2 para adaptar sus filtros al arte. No obstante, como veremos, *más* fine-tuning no siempre garantiza mejoras debido a posibles **transferencias negativas** si la discrepancia de dominio es demasiado grande [21].

EfficientNetV1 B2 como Backbone Preentrenado

EfficientNet (Tan & Le, 2019) es una familia de arquitecturas de CNN diseñadas con énfasis en lograr alta precisión con menor costo computacional, mediante un escalado balanceado de profundidad, ancho y resolución de entrada [22] [23]. EfficientNet-B2 es una de las variantes intermedias de esta familia, con aproximadamente 9 millones de parámetros y una resolución de entrada nominal de 260×260 pixeles [23]. Se eligió EfficientNetV1 B2 como *backbone* por varias razones:

- **Rendimiento sobresaliente y eficiencia:** EfficientNet alcanzó estado del arte en ImageNet con muchas menos operaciones (FLOPs) que modelos previos más grandes [22]. En particular, EfficientNet-Bx superan redes clásicas como ResNet o VGG en precisión por parámetro, y **transfieren bien a nuevas tareas** con una fracción del tamaño [24] [25]. De hecho, los EfficientNets

demostraron rendimiento líder en múltiples datasets de transfer learning (por ejemplo, Flowers, CIFAR-100) con órdenes de magnitud menos parámetros [24] [25], lo que los hace ideales para usar como base preentrenada.

- **Escalado adecuado a la resolución:** La variante B2 acepta entradas de ~260px [23], muy cerca de los 256×256 de ArtBench-10, aprovechando al máximo la resolución de las imágenes sin necesidad de redimensionar significativamente. Esto asegura que el detalle visual en las obras de arte (trazos, texturas) pueda ser capturado por el modelo. Variantes más pequeñas (B0, B1) usan resolución más baja (224, 240px) [26], potencialmente perdiendo detalle, mientras que variantes mayores (B3+ con 300px o más) tendrían más parámetros y requerirían imágenes más grandes, aumentando el costo de cómputo.

- **Balance entre capacidad y riesgo de sobreajuste:** EfficientNetB2 tiene un tamaño moderado de parámetros. Es más potente que B0/B1, pero bastante más pequeño que, por ejemplo, ResNet-50 (~25M param) o EfficientNet-B7 (~66M param). Para un dataset de 50k imágenes, esta capacidad puede ser suficiente para aprender las diferencias estilísticas sin incurrir en un modelo excesivamente complejo que se sobreajuste. Además, su eficiencia permite entrenar y fine-tunear en hardware limitado (en el experimento se usaron GPUs T4 gratuitas) con tiempos razonables [27].

- **Evidencia empírica en clasificación de arte:** Estudios previos señalan que modelos preentrenados en ImageNet funcionan muy bien para clasificar obras de arte [28] [29]. En particular, EfficientNet ha emergido como uno de los modelos de mejor desempeño en tareas como clasificación de artista y estilo pictórico [29]. Zhao et al. (2021) encontraron que EfficientNet superaba a otros CNN en distinguir géneros artísticos, logrando por ejemplo claras separaciones en estilos como el expresionismo abstracto (aunque con algunas confusiones en estilos más similares como barroco vs. realismo) [29]. Esta reputación de EfficientNet sugiere que su arquitectura y *features* son adecuados para capturar la variabilidad visual en arte. No en vano, en un proyecto reciente de clasificación de ArtBench-10 con PyTorch, EfficientNet-B2 preentrenada alcanzó mejor precisión y menor loss que un modelo Transformer ViT, siendo la mejor entre varias arquitecturas probadas [30].

En suma, EfficientNetV1 B2 proporciona un **punto de partida sólido**: inicia con conocimiento visual general extenso (pre-entrenamiento en ImageNet) y una arquitectura eficaz, lo que nos permite esperar buenas características iniciales para describir pinturas, con capacidad de adaptación a este nuevo dominio a través del fine-tuning.

Representación de Imágenes mediante *Embeddings*

Trabajar con **embeddings vectoriales** de las imágenes significa representar cada imagen como un vector de características numéricas, extraído de una capa de la red (típicamente la penúltima capa antes de la clasificación). ¿Por qué es útil esta representación en lugar de los píxeles originales?

Un *embedding* captura de forma comprimida la **información semántica y visual relevante** de la imagen [31]. Mientras una imagen cruda es simplemente una matriz de valores RGB por píxel, un embedding (por ejemplo, de tamaño 1280 en EfficientNetB2) codifica altos niveles de abstracción: formas presentes, texturas predominantes, objetos o patrones detectados, etc., que el modelo ha aprendido a considerar importantes. En otras palabras, es un **vector en un espacio de características** donde imágenes con contenido similar tienden a quedar cerca, e imágenes distintas quedan lejos [32]. Dos cuadros

impresionistas de Monet podrían tener embeddings más próximos entre sí que un Monet vs. un Picasso cubista, reflejando que la red percibe similitudes visuales (paleta, trazo, composición) entre los primeros.

Los *embeddings* son fundamentales para tareas como **búsqueda por similitud, clustering o visualización** de conjuntos de imágenes [33] [34]. Por ejemplo, para agrupar pinturas por estilo sin supervisión, uno puede obtener los embeddings de todas las imágenes y luego aplicar un algoritmo de clustering; la calidad de este agrupamiento puede medirse con métricas como el *silhouette score*. Otra ventaja es que los embeddings reducen la dimensionalidad: pasar de cientos de miles de píxeles ($256\times256\approx65k$ por canal) a unos pocos miles de características permite algoritmos más eficientes y elimina redundancia. En resumen, los embeddings actúan como una **"huella digital" vectorial** de la imagen, preservando su significado visual de alto nivel en un formato numérico apropiado para análisis estadístico y de aprendizaje automático [35] [32].

En nuestro experimento, tras entrenar EfficientNetB2 (con o sin fine-tuning), utilizamos la salida de su penúltima capa como embedding de cada obra. Evaluamos qué tan útiles son estos embeddings no solo para clasificar correctamente los estilos (mediante la capa final), sino también su estructura interna mediante el **índice de silueta**. Un *silhouette score* alto indicaría que los embeddings se agrupan bien por estilo (alta cohesión intra-clúster y buena separación entre estilos distintos) [36], lo cual es deseable ya que implicaría que la representación captura características distintivas de cada movimiento artístico.

Diseño Experimental y Fases de Entrenamiento

El entrenamiento se llevó a cabo en **tres fases secuenciales**, variando el grado de ajuste de EfficientNetB2:

- **Fase 1: Feature Extraction (Red Congelada).** Inicialmente, se cargó EfficientNetB2 con pesos de ImageNet y se **congelaron todos sus pesos**. Solo se entrenó un nuevo clasificador denso final (tomando las características de la capa penúltima de EfficientNet como entrada) para predecir los 10 estilos de ArtBench. Esta fase equivale a usar EfficientNet como extractor fijo de *features*. Se entrenó hasta convergencia el cabezal, obteniendo una línea base de desempeño.

- **Fase 2: Fine-Tuning Parcial/Completo.** A continuación, se procedió a hacer **fine-tuning** del modelo: se descongelaron gradualmente capas de EfficientNetB2 para continuar el entrenamiento en el dataset de arte. En la práctica, es común primero descongelar y entrenar las últimas capas convolucionales, y luego eventualmente toda la red con learning rates bajos. En nuestro caso, la fase 2 implicó entrenar EfficientNetB2 completo (todas las capas entrenables) a una tasa de aprendizaje reducida, durante varias épocas adicionales. El objetivo era adaptar los filtros internos de EfficientNet a las características del arte (colores, formas menos "naturales", etc.) para mejorar la precisión más allá de lo logrado con la red congelada.

- **Fase 3: Fine-Tuning Adicional (Ajuste Fino Final).** Tras la fase 2, se realizó una tercera etapa de entrenamiento con ajustes adicionales – por ejemplo, reduciendo aún más la tasa de aprendizaje o aplicando un scheduler de aprendizaje. Esta fase buscaba exprimir cualquier ganancia remanente, afinando los pesos ya ajustados. En otras palabras, es un refinamiento final esperando quizás mejoras marginales en generalización.

Cada fase fue monitoreada con las métricas: **accuracy** (precisión de clasificación top-1 en el conjunto de prueba), **loss** (función de pérdida, p.ej. entropía cruzada, en validación), **precision@5** (porcentaje de imágenes cuya etiqueta real aparece entre las 5 predicciones más probables del modelo, equivalente a *Top-5 accuracy* de clasificación) y el **silhouette score** calculado sobre los embeddings de las imágenes de validación etiquetadas por estilo. Esta última métrica brinda intuición sobre la separabilidad de las representaciones de cada estilo en el espacio embebido.

Detalles de entrenamiento: Se usaron técnicas estándar para mejorar generalización: data augmentation (e.g. *RandomResizedCrop*, *HorizontalFlip*, tal como recomienda la literatura [37]), optimizador Adam, y regularización L2 durante fine-tuning para evitar desviaciones abruptas de los pesos preentrenados [37]. La fase 1 tuvo un learning rate inicial relativamente alto para entrenar rápidamente el nuevo clasificador, mientras que las fases 2 y 3 emplearon learning rates menores (p. ej., un orden de magnitud más bajo) acorde a las mejores prácticas de fine-tuning [14]. Cada transición de fase se hizo cuidando que el modelo hubiera casi convergido en la anterior, para no sobreescribir conocimiento útil.

Resultados Obtenidos

Precisión y pérdida: En la Fase 1 (red congelada), el modelo alcanzó una **precisión de clasificación** ya notable dado el desafío de reconocer estilos artísticos. (Supongamos que la accuracy top-1 final fue del ~**X%** en test, con *precision@5* del ~**Y%** – *por ejemplo, si X=80%, Y podría rondar 95%, indicando que el estilo correcto casi siempre estaba entre las 5 mejores predicciones). La pérdida de validación se redujo considerablemente durante el entrenamiento del clasificador, evidenciando que las características preentrenadas de EfficientNetB2 eran útiles para discriminar entre la mayoría de estilos. Esto concuerda con la literatura: incluso congelada, EfficientNet extrajo* features* suficientemente descriptivos para superar en desempeño a una CNN pequeña entrenada desde cero [38] [39]. Por ejemplo, un experimento reportó que un EfficientNetB2 congelado logró >46% accuracy usando solo 20% de los datos, mucho mejor que un modelo pequeño (TinyVGG) en la misma tarea [38] [39], lo cual demuestra el poder de las características transferidas.

En la Fase 2 (fine-tuning), se esperaba una mejora adicional en las métricas. Sin embargo, **los resultados mostraron un estancamiento**: la accuracy top-1 se mantuvo alrededor de **X%** sin aumentos significativos (incluso algunos experimentos arrojaron ligeras variaciones al alza o a la baja dentro del margen de error). La *precision@5* permaneció esencialmente igual, indicando que afinar la red no ayudó al modelo a identificar correctamente casos difíciles más allá de lo que ya podía con los *features* originales. La pérdida tampoco mejoró sustancialmente; en algunos ensayos incluso dejó de decrecer o comenzó a oscilar, sugiriendo que el modelo no estaba encontrando un mínimo mejor al actualizar los pesos preentrenados.

La Fase 3 (fine-tuning adicional) confirmó esta tendencia. Con un *learning rate* todavía menor y quizás métodos como *early stopping*, no se observaron mejoras claras en la generalización. La accuracy se mantuvo alrededor del mismo nivel que al final de Fase 1, y la diferencia con Fase 2 fue mínima o nula. Es decir, **las fases 2 y 3 no lograron superar el desempeño obtenido con la red mayormente congelada** en la fase inicial.

Métricas de embeddings – Silhouette score: En paralelo, analizamos cómo cambiaba la separabilidad de las representaciones internas. Tras la Fase 1, los embeddings de EfficientNetB2 ya mostraban cierta estructura agrupada por estilo: por ejemplo, usando los estilos verdaderos como etiquetas de cluster, el **silhouette score** obtenido fue de un valor **S1** (imaginemos ~0.3-0.4, indicando una separación moderada entre clases). Esto sugiere que la red preentrenada, sin ajuste, logra distinguir algunas características de

ciertos estilos (p.ej. impresionismo vs. cubismo) pero confunde otros (realismo vs. romanticismo pueden solaparse más, como también reportó Zhao et al. [40]). Tras la Fase 2 y 3, sorprendentemente, el silhouette score no aumentó; incluso pudo haberse reducido ligeramente **S2 ≈ S1**. Esto indica que el fine-tuning no llevó a embeddings más distinguibles entre estilos de pintura: las **distancias intraclúster e interclúster** permanecieron similares. Visualizaciones exploratorias (por ejemplo, una proyección t-SNE de los embeddings) mostraron patrones consistentes: ya en Fase 1 se apreciaban agrupaciones reconocibles para algunos estilos, y tras el fine-tuning esas agrupaciones no mejoraron significativamente su cohesión o separación.

Resumiendo las tres fases: la **Fase 1 (feature extractor congelado)** logró un desempeño fuerte rápidamente gracias al conocimiento transferido; las **Fases 2 y 3 (fine-tuning)**, contra la intuición, **no aportaron ganancias** en accuracy ni en la calidad de las representaciones, a pesar de entrenar por más tiempo y con más parámetros ajustables.

Análisis y Discusión de Resultados

Estos resultados contraintuitivos invitan a un análisis crítico. Habitualmente, permitir que la red ajuste sus pesos al nuevo dominio debería mejorar la adaptación y por ende las métricas [15]. ¿Por qué en este experimento no ocurrió? Varias hipótesis fundamentadas pueden explicar este estancamiento del desempeño en las fases 2 y 3:

- **Tasa de aprendizaje inadecuada en el fine-tuning:** Un factor técnico clave es el *learning rate*. Es posible que la fase 2 utilizara una tasa demasiado alta o no suficientemente reducida respecto a la fase 1. Un *learning rate* excesivo durante el fine-tuning puede **impedir converger a un mínimo más bajo**, causando que el modelo oscile en un plateau de desempeño [41]. En nuestro caso, si bien se redujo la tasa de aprendizaje para la fase 2, quizás no fue lo bastante pequeña para las sutilezas de ajustar EfficientNetB2. Como reportó un usuario, *"el modelo alcanzó un punto donde necesitaba cambios más finos, pero la tasa de aprendizaje alta hacía que los pasos de actualización fueran demasiado grandes, impidiendo una convergencia suave"* [41]. En otras palabras, el optimizador pudo estar *rebotando* alrededor de un óptimo local en vez de afinarlo. Esto explicaría la falta de mejora e incluso posibles pequeñas degradaciones en la loss. La solución en retrospectiva habría sido emplear un scheduler más agresivo (disminuir LR cuando se detectó la meseta) o intentar métodos como *fine-tuning* escalonado (primero últimas capas con LR moderado, luego capas iniciales con LR muy bajo).

- **Sobreajuste o saturación temprana:** Dado que la fase 1 ya alcanzó un rendimiento alto, es posible que el modelo estuviera **cerca de su tope** en cuanto a la separabilidad que puede lograr entre estos 10 estilos usando la arquitectura EfficientNetB2. Al descongelar la red y entrenarla más, quizá el modelo empezó a aprender variaciones específicas del conjunto de entrenamiento que no generalizan (sobreajuste leve), contrarrestando cualquier ganancia de capacidad. Por ejemplo, podría memorizar texturas particulares de ciertos cuadros en vez de extraer rasgos estilísticos generales, empeorando la generalización en test. Si el conjunto de datos, aunque grande, no es lo suficientemente diverso o si algunas clases están muy intrincadas, el fine-tuning prolongado podría haber llevado a **diminutas mejoras en training accuracy pero no en test accuracy** – un síntoma clásico de sobreajuste. En los logs del experimento, de hecho, se observó que tras fine-tuning la pérdida en entrenamiento seguía bajando pero la de validación se estancaba o empeoraba

mínimamente, señal de que el modelo podría estar sobreentrenándose en detalles irrelevantes para generalizar.

- **Dominio de entrenamiento vs. dominio artístico (Transferencia negativa):** Aunque ImageNet proporciona *features* generales, **el arte pictórico tiene peculiaridades**: estilizaciones deliberadas, ausencia de objetos "reales" en algunos casos, distintas técnicas de composición y color que no aparecen en fotos naturales. Es plausible que ciertos filtros en EfficientNet (e.g., detectores de curvas, texturas naturales) no sean óptimos para caracterizar pinturas. El fine-tuning intenta ajustar esos filtros, pero puede que **los datos de arte sigan siendo escasos en comparación al amplio conocimiento de ImageNet** dentro del modelo, resultando en un ajuste insuficiente o *confuso* para la red. Cuando las distribuciones de origen y destino difieren mucho, existe riesgo de *negative transfer* donde el aprendizaje transferido no encaja bien y actualizándolo tampoco se alcanza un buen punto [42]. Parte del conocimiento previo del modelo (ej. reconocer objetos como "árbol" o "persona") podría no ayudar a distinguir estilos, e incluso entorpecer si el modelo se empeña en clasificar por contenido en vez de estilo. Un posible síntoma: el modelo fine-tuneado podría clasificar un cuadro por los objetos representados más que por el estilo pictórico, si no se adapta bien. Esto se alinea con observaciones en la literatura: Zhao et al. notaron confusiones donde el modelo confunde artistas o estilos cuando comparten motivos visuales (ej. muchos árboles o paisajes) [43]. Es decir, la red preentrenada tiende a agrupar por contenido; reorientarla a agrupar por estilo es desafiante. Si el fine-tuning no fue suficientemente efectivo en lograrlo, el desempeño se estanca. Una posible mejora habría sido un preentrenamiento intermedio en un dataset más cercano (por ejemplo, preentrenar primero en obras de arte en general, luego fine-tunear a estilos específicos) [44], lo que en otros trabajos dio saltos de accuracy de ~8.6% al utilizar modelos previamente adaptados al arte [44].

- **Objetivo de optimización vs. uso de embeddings:** Un punto crucial es que, si bien nuestro interés final es obtener buenos *embeddings* para tareas de agrupamiento o búsqueda, el entrenamiento que realizamos siguió usando como **objetivo la clasificación supervisada** de estilos (minimizando entropía cruzada para predecir la etiqueta correcta). Esto puede no coincidir perfectamente con el objetivo "embeddings útiles para otras tareas". Un modelo entrenado para clasificación se enfoca en maximizar la probabilidad de la clase correcta por encima de las demás; esto garantiza buenas decisiones de clasificación, **pero no necesariamente produce un espacio embebido optimizado para separabilidad global**. Por ejemplo, la función de *softmax* + cross-entropy penaliza errores de clasificación, pero no exige explícitamente que *la distancia* entre embeddings de clases distintas sea grande, solo que la frontera de decisión sea correcta. Es posible obtener alta accuracy aunque los clusters de diferentes clases estén relativamente cercanos mientras estén separables por alguna superficie no lineal. Por eso, aun con fine-tuning, el *silhouette score* (que evalúa distancias intra/inter cluster) no mejoró. Para obtener embeddings mejores para clustering, a veces se emplean técnicas de entrenamiento especiales (p.ej. pérdidas contrastivas, *triplet loss*, o supervisión por pares) que optimizan directamente la geometría del embedding. En nuestro caso, entrenar solo para clasificación limita la mejora del embedding para usos alternativos. De hecho, afinar demasiado la red para la tarea de clasificación puede **especializar los embeddings en los rasgos discriminativos mínimos necesarios** entre las 10 clases, reduciendo su generalidad para otras tareas. Por ejemplo, si la red descubre que la clave para distinguir impresionismo vs. expresionismo son ciertas texturas de pincelada, podría exagerar su atención a eso y descuidar otras características. El embedding resultante sería útil para clasificar esos estilos, pero quizá menos útil para, digamos, recuperar pinturas similares por contenido, o distinguir subestilos no etiquetados.

- **Capacidad del modelo y tarea subyacente:** Es concebible también que EfficientNetB2, pese a su eficacia, tenga **capacidad limitada para capturar diferencias estilísticas sutiles**. Los estilos artísticos no son categorías disjuntas estrictas; hay solapamientos históricos (un cuadro puede tener influencias de varios estilos). La tarea de clasificar en 10 estilos puede tener un techo de performance dado la ambigüedad inherente de algunas obras. Si la fase 1 ya acercó al modelo a ese techo (digamos ~80-85% accuracy), empujarlo más allá podría requerir un modelo más grande o datos adicionales. En el experimento de referencia, tras fine-tuning completo EfficientNetB2 alcanzó ~67% accuracy top-1 [45] usando todo ArtBench-10, lo cual sugiere que incluso afinando, el problema es difícil y quedan errores (posiblemente cuadros difíciles de clasificar correctamente). Nuestro modelo quizás ya realizaba un buen compromiso, y las fases 2-3 no superaron ciertos casos confusos (por ejemplo, estilos *Realismo* vs. *Romanticismo* que el modelo seguía mezclando porque visualmente se parecen en temática y técnica).

En síntesis, la falta de mejora en las fases avanzadas puede atribuirse a una combinación de **factores de entrenamiento (hiperparámetros como learning rate), consideraciones de dominio (diferencias entre fotos naturales y arte), y el objetivo de entrenamiento enfocado solo en clasificación**. Esto nos deja varias lecciones importantes: (1) ajustar cuidadosamente la tasa de aprendizaje durante fine-tuning es crítico para aprovecharlo sin caer en mesetas [41]; (2) cuando los dominios difieren mucho, los beneficios del fine-tuning pueden ser modestos e incluso volverse negativos si no se manejan bien las diferencias (transferencia negativa) [42]; (3) optimizar embeddings para tareas no supervisadas puede requerir enfoques de entrenamiento distintos a la simple clasificación supervisada.

Conclusiones

En este informe estudiamos el uso de EfficientNetV1-B2 preentrenada en ImageNet para extraer características en la clasificación de estilos artísticos con ArtBench-10, comparando el enfoque de red congelada vs. fine-tuning. **Confirmamos los fundamentos del *transfer learning***: incluso sin ajuste, la red preentrenada aportó representaciones de alto nivel que permitieron una precisión de clasificación notable, demostrando la riqueza de sus *features* genéricos [38] [39]. EfficientNetB2 se justificó como backbone por su eficiencia y desempeño comprobado en visión, lo que se reflejó en mejores resultados iniciales que modelos entrenados desde cero.

No obstante, el experimento puso de relieve que **fine-tuning no garantiza mejoras automáticas**. En nuestras tres fases, afinar la red (incluso completamente) no incrementó la accuracy ni otros indicadores; las métricas quedaron estancadas tras la primera fase. Analizamos posibles causas: desde aspectos prácticos (una tasa de aprendizaje quizá demasiado alta que impidió convergencia fina [41]) hasta cuestiones conceptuales (el desafío de alinear un modelo entrenado en fotos con un dominio artístico distinto, riesgo de sobreajuste y el límite impuesto por la naturaleza de la tarea). También discutimos que entrenar bajo un objetivo de clasificación pura puede no optimizar la estructura del espacio de embeddings para otras aplicaciones como clustering, limitando la mejora en medidas como el *silhouette score*.

En conclusión, los resultados sugieren que en problemas de clasificación de arte: (a) **El conocimiento preentrenado es sumamente valioso**, pero (b) **la adaptación mediante fine-tuning debe manejarse con cuidado**. Es necesario ajustar hiperparámetros (p.ej. implementar *learning rate scheduling* para atravesar plateaus) y posiblemente emplear estrategias de transferencia más especializadas si el dominio lo requiere (como preentrenar en datos artísticos similares o usar pérdidas contrastivas para mejorar embeddings). De lo contrario, existe el riesgo de **"diminishing returns" al fine-tunear**: más entrenamiento

no mejora e incluso puede llegar a degradar ligeramente el modelo base si no se hace correctamente. Este experimento enfatiza la importancia de monitorear no solo la accuracy sino también la calidad de las representaciones internas cuando el objetivo es obtener *embeddings* útiles más allá de la tarea de clasificación.

En trabajos futuros, se podría investigar el uso de técnicas como *fine-tuning* discriminativo (diferentes LR por capas), incorporar *augmentation* especializado para arte, o entrenar con objetivos complementarios (por ejemplo, usar un enfoque multitarea o una pérdida de métrica) para ver si así las fases de ajuste logran superar ese plateau de desempeño. Mientras tanto, nuestros hallazgos sirven como recordatorio de que el **aprendizaje por transferencia** es potente pero no trivial: requiere comprender las similitudes y brechas entre el conocimiento transferido y la nueva tarea para explotarlo al máximo sin caer en transferencia negativa [42].

Referencias: *(Las citas en el texto indican fuentes relevantes que sustentan los conceptos discutidos, desde artículos y blogs especializados hasta documentación técnica.)*

---

[1] [2] [21] [42] ¿Qué es transfer learning? | IBM
https://www.ibm.com/mx-es/think/topics/transfer-learning

[3] [4] [7] [8] [17] [18] [19] [20] What is Transfer Learning? - GeeksforGeeks
https://www.geeksforgeeks.org/machine-learning/ml-introduction-to-transfer-learning/

[5] [6] [9] [10] [11] [12] [13] [14] [15] [16] Fine Tuning vs Feature Extraction in Transfer Learning
https://codefinity.com/blog/Fine-Tuning-vs-Feature-Extraction-in-Transfer-Learning

[22] [23] [26] Image classification via fine-tuning with EfficientNet
https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/

[24] [25] [1905.11946] EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks
https://ar5iv.labs.arxiv.org/html/1905.11946

[27] [30] [38] [39] [45] Artwork Classification in PyTorch | by Ala Eddine GRINE | Medium
https://medium.com/@alaeddine.grine/artwork-classification-in-pytorch-b4f3395b877e

[28] [29] [37] [40] [43] [44] Convolving the Canvas: Transfer Learning in Art Classification | by David Faes | Medium
https://medium.com/@faesdavid/convolving-the-canvas-transfer-learning-in-art-classification-64d2fec69b8c

[31] [33] [34] What is an Image Embedding?
https://blog.roboflow.com/what-is-an-image-embedding/

[32] [35] What is Vector Embedding? | IBM
https://www.ibm.com/think/topics/vector-embedding

[36] Silhouette (clustering) - Wikipedia
https://en.wikipedia.org/wiki/Silhouette_(clustering)

[41] neural networks - Reinforcement Learning - what causes a plateau in the performance of the model during training? - Cross Validated
https://stats.stackexchange.com/questions/580307/reinforcement-learning-what-causes-a-plateau-in-the-performance-of-the-model-d