Optimización de la Clasificación de Beans mediante Congelamiento Selectivo en ResNet18

Pedro Ortiz Villanueva
Universidad Politécnica de Madrid
Madrid, España
pedro.ortiz@upm.es

***Resumen*—Este estudio desafía la creencia común de que entrenar todos los parámetros de un modelo preentrenado conduce necesariamente a mejores resultados. Utilizando una única arquitectura ResNet18, investigamos cómo el congelamiento estratégico de diferentes bloques puede mantener o incluso mejorar el rendimiento mientras se reduce significativamente el número de parámetros entrenables y los recursos computacionales necesarios. El objetivo es demostrar que una configuración más eficiente de los parámetros entrenables puede lograr resultados comparables o superiores, optimizando así el uso de recursos computacionales sin comprometer el rendimiento.**

***Index Terms*—Deep Learning, Transfer Learning, ResNet18, Congelamiento Selectivo, Optimización de Recursos, Clasificación de Imágenes**

I. Introducción

Si bien históricamente existió una tendencia a utilizar todos los parámetros disponibles en modelos preentrenados [3], actualmente es bien conocida la importancia crítica de la eficiencia computacional. Este cambio de paradigma está impulsado por la necesidad de democratizar el acceso a los modelos de aprendizaje profundo, haciéndolos más accesibles y prácticos para su implementación en servicios del mundo real [4].

Esta tendencia hacia la democratización se refleja en el desarrollo de diversas técnicas de optimización como:

- **LoRA** (Low-Rank Adaptation) [6], que permite el fine-tuning eficiente mediante la factorización de matrices
- **Destilación de conocimiento** [9] para crear modelos más compactos
- **Cuantización de pesos** [7], [8] para reducir los requisitos de memoria y acelerar la inferencia

En este contexto, el estudio utiliza la ResNet18 [2] como caso de estudio para demostrar que, mediante el congelamiento selectivo de capas, podemos reducir significativamente la cantidad de parámetros entrenables mientras mantenemos o mejoramos el rendimiento del modelo.

II. Estado del Arte

La optimización de recursos sin comprometer el rendimiento se ha convertido en un factor crucial para hacer que las soluciones de inteligencia artificial sean verdaderamente escalables y accesibles para todo el mundo, desde pequeñas empresas hasta aplicaciones de gran escala. En los últimos años, se han desarrollado diversas técnicas para optimizar el entrenamiento y despliegue de modelos de deep learning, siendo el transfer learning una de las más exitosas [3].

III. Metodología

III-A. Dataset

El estudio utiliza el dataset "Beans"proporcionado por AI-Lab-Makerere a través de HuggingFace, que consiste en imágenes de plantas de frijol para clasificación. Los datos se procesan mediante un pipeline de transformaciones que incluye:

- Redimensionamiento de imágenes a 224x224 píxeles
- Normalización utilizando los valores estándar de ImageNet [5]:
  - mean=[0.485, 0.456, 0.406]
  - std=[0.229, 0.224, 0.225]
- Transformación a tensores para su procesamiento en PyTorch

III-B. Arquitectura del Modelo

Se utiliza una ResNet18 pre-entrenada en ImageNet (IMAGENET1K_v1) como modelo base, modificada para la tarea de clasificación específica:

- **Modelo base**: ResNet18 con pesos pre-entrenados
- **Capa de clasificación**: Adaptada para 3 clases
- **Estrategia de congelamiento**: Implementación de congelamiento selectivo por bloques

III-C. Proceso de Entrenamiento

La configuración base del entrenamiento incluye:

Parámetro	Valor
Batch size	16
Learning rate	1e-4
Max epochs	30
Workers	4

Cuadro I
Configuración base del entrenamiento

La implementación del proceso de entrenamiento se realizó mediante un framework de alto nivel basado en PyTorch Lightning, que facilita la reproducibilidad y escalabilidad del experimento. La estrategia de entrenamiento incorpora diversos mecanismos de control y optimización:

- **Mecanismos de Control**: Se implementaron callbacks para la persistencia selectiva de modelos basada en el rendimiento de validación, junto con un sistema de early stopping para prevenir el sobreajuste. Ambos mecanismos operan sobre la métrica de pérdida en validación con un período de paciencia establecido.
- **Optimización de Recursos**: Se empleó un esquema de precisión mixta para optimizar el uso de memoria y velocidad de procesamiento, particularmente relevante en el entorno de ejecución basado en GPU (Nvidia T4).
- **Monitorización y Registro**: La instrumentación del entrenamiento incluye un sistema de logging basado en TensorBoard, que facilita el seguimiento de métricas y la visualización de resultados. El registro de métricas se realiza con una granularidad temporal apropiada para capturar la dinámica del entrenamiento sin comprometer el rendimiento.

El framework de entrenamiento se configuró con determinismo computacional para garantizar la reproducibilidad de los resultados, un aspecto crucial en la investigación científica. La limitación del número máximo de épocas, combinada con el mecanismo de early stopping, proporciona un equilibrio entre la exploración del espacio de soluciones y la prevención del sobreajuste.

IV. Experimentos y Resultados

IV-A. Análisis Experimental

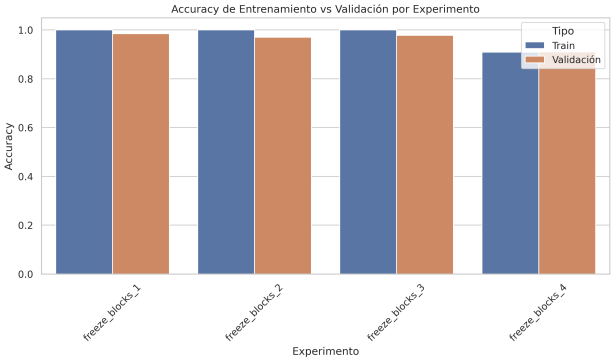


Figura 1. Accuracies de Entrenamiento y Validación para diferentes configuraciones de congelamiento

IV-A1. Rendimiento del Modelo: Puesto que se trata de un problema de clasificación sencillo, hasta la congelación de la mayoría del modelo rinde excepcionalmente bien. Esto se ha hecho para remarcar aún más el hecho de que no necesitamos hacer full-finetunings ni grandes entrenamientos para obtener resultados notables. Incluso se puede ver que a partir de la congelación de los dos últimos bloques el rendimiento extra es residual.

La visualización del loss durante el entrenamiento es bastante ilustrativa. Puesto que se trata de un problema de clasificación sencilla, el modelo converge rápidamente. Además, se puede ver cómo el loss durante la validación es bastante estable

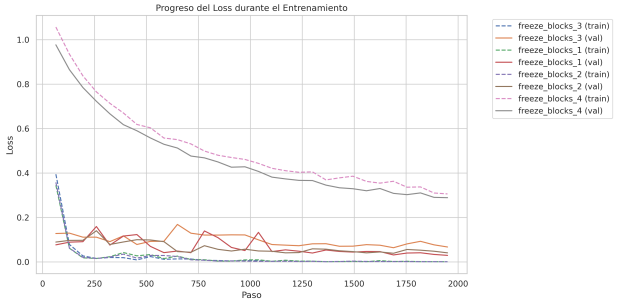


Figura 2. Evolución del Loss Durante el Entrenamiento

y se mantiene parejo al del training, dándonos a entender que no estamos sufriendo de overfitting.

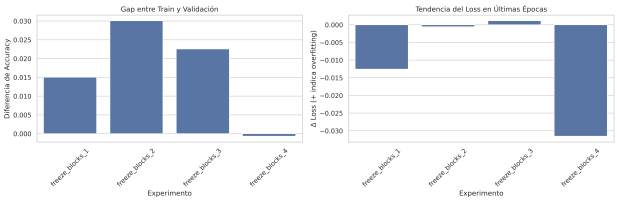


Figura 3. Estudio de Overfitting

IV-A2. Análisis de Generalización: La gráfica de análisis de overfitting muestra claramente que no hay un overfitting latente en el modelo. El que más indicios de generalización muestra es el modelo de 4 bloques congelados puesto que tiene que lidiar con todo un ”forward pass”de 4 bloques de ResNet18 sin ningún parámetro entrenable.

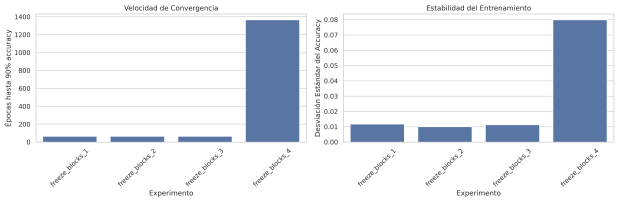


Figura 4. Comparativa de Convergencia

Como es natural, cuanto más parámetros más rápido va a converger el modelo. Le va a costar muy poco encontrar un óptimo, todo lo contrario al modelo de 4 bloques congelados cuya convergencia es más lenta. La desviación estándar del 4 es más alta, sus steps son más abruptos.

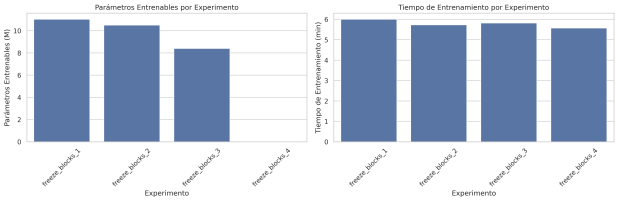
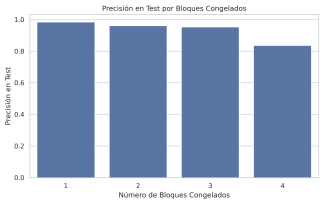


Figura 5. Parámetros vs Tiempo de Entrenamiento

IV-A3. Eficiencia Computacional: Esto ejemplifica muy bien cómo el experimento al ser pequeño, se da la dicotomía entre los tiempos similares de entrenamiento y el número de parámetros entrenables. Puesto que el cuello de botella no está en el training, sino en el batch size y los dataloaders. Esto hace que usar una GPU para problemas pequeños no sea una buena inversión de recursos.



Resultados Detallados

Configuración	Accuracy
Bloques 1	0.9844
Bloques 2	0.9609
Bloques 3	0.9531
Bloques 4	0.8359

Figura 6. Accuracy en Test

IV-A4. Resultados Finales: Es de esperar que el modelo de 4 bloques congelados tenga el menor accuracy puesto que es el que menos parámetros entrenables tiene. Además, se puede ver cómo el modelo de 1 bloque congelado rinde casi igual que el de 2 y 3, nos da a pensar que no compensa usar más parámetros entrenables, solo los necesarios.

V. Conclusiones

Los resultados obtenidos en este estudio respaldan firmemente nuestra hipótesis inicial sobre la optimización de recursos mediante el congelamiento selectivo de capas en arquitecturas preentrenadas. A través de una serie de experimentos sistemáticos con ResNet18, hemos demostrado que no es necesario entrenar todos los parámetros del modelo para obtener un rendimiento óptimo en tareas de clasificación específicas.

El análisis detallado de diferentes configuraciones de congelamiento reveló que podemos mantener un accuracy bueno para este caso de uso (superior al 95 %) incluso cuando congelamos hasta tres bloques del modelo. Este hallazgo es particularmente significativo desde la perspectiva de la eficiencia computacional y la optimización de recursos.

Referencias

[1] Autor, A. (Año) "Título del artículo," Nombre de la revista, vol. X, no. Y, pp. ZZ-ZZ.

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2016). "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[3] Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). "How transferable are features in deep neural networks?" In Advances in neural information processing systems (pp. 3320-3328).

[4] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861.

[5] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). "Imagenet: A large-scale hierarchical image database." In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255).

[6] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2021). "LoRA: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685.

[7] Nagel, M., & Kutz, D. (2021). "Low-bit Neural Network Quantization: A Review." arXiv preprint arXiv:2106.11720.

[8] Zhao, Y., et al. (2022). "AWQ: Adaptive Weight Quantization for Efficient Inference." arXiv preprint arXiv:2206.05246.

[9] Gou, J., et al. (2021). "Knowledge Distillation: A Survey." arXiv preprint arXiv:2006.05525.