Clasificación de Especies de Hongos Mediante Deep Learning y Transfer Learning

Diego Alfaro

10 de noviembre de 2025

Resumen

Este reporte presenta un estudio comparativo entre dos arquitecturas de redes neuronales convolucionales (CNN) para la clasificación de especies de hongos. Se implementó un modelo baseline entrenado desde cero y un modelo basado en transfer learning utilizando EfficientNet-B0 preentrenado con ImageNet. Los resultados demuestran que el transfer learning logra una mejora significativa en la precisión de clasificación, alcanzando un 99.07 % en el conjunto de prueba, comparado con el 15.70 % del modelo baseline.

Índice

1. Introducción

La clasificación automática de especies de hongos es un problema desafiante en el campo de la visión por computadora debido a la alta variabilidad intra-clase y las similitudes inter-clase entre diferentes especies. Este proyecto evalúa dos enfoques de aprendizaje profundo para abordar este problema: un modelo CNN diseñado desde cero y un modelo basado en transfer learning.

1.1. Objetivos

- Desarrollar e implementar dos arquitecturas de CNN para clasificación de hongos
- Comparar el rendimiento entre entrenamiento desde cero y transfer learning
- Evaluar la efectividad de los pesos preentrenados de ImageNet en este dominio específico
- Analizar las diferencias en tiempo de entrenamiento y precisión final

2. Dataset

2.1. Descripción del Dataset

El dataset utilizado proviene de Kaggle y contiene imágenes de diferentes especies de hongos. Las características principales son:

- **Fuente**: Kaggle - Mushroom Species Classification Dataset
- **Número de clases**: 169 especies diferentes de hongos
- **Tamaño de imágenes**: Imágenes de alta resolución redimensionadas a 224x224 píxeles
- **División del dataset**:
  - Conjunto de entrenamiento: 80 % de los datos
  - Conjunto de validación: 10 % de los datos
  - Conjunto de prueba: 10 % de los datos

2.2. Preprocesamiento de Datos

Se aplicaron las siguientes transformaciones:
Conjunto de entrenamiento (Data Augmentation):

- Redimensionamiento a 256x256 píxeles
- Recorte aleatorio a 224x224 píxeles
- Volteo horizontal aleatorio (probabilidad = 0.5)
- Rotación aleatoria (hasta $\pm 15°$)

- Normalización con media y desviación estándar de ImageNet

Conjuntos de validación y prueba:

- Redimensionamiento a 256x256 píxeles
- Recorte central a 224x224 píxeles
- Normalización con media y desviación estándar de ImageNet

3. Arquitecturas de Modelos

3.1. Modelo Baseline - CNN desde Cero

3.1.1. Descripción de la Arquitectura

Se diseñó una CNN con 5 bloques convolucionales progresivos:

Bloque	Entrada	Filtros	Salida	Operaciones
Entrada	224x224	-	-	Imagen RGB
Bloque 1	224x224	32	112x112	Conv + BN + ReLU + MaxPool
Bloque 2	112x112	64	56x56	Conv + BN + ReLU + MaxPool
Bloque 3	56x56	128	28x28	Conv + BN + ReLU + MaxPool
Bloque 4	28x28	256	14x14	Conv + BN + ReLU + MaxPool
Bloque 5	14x14	512	7x7	Conv + BN + ReLU + MaxPool

Cuadro 1: Arquitectura del modelo baseline CNN

Características del clasificador:

- Adaptive Average Pooling a 1x1
- Capa completamente conectada: 512 $\rightarrow$ 256 (Dropout 0.5)
- Activación ReLU
- Capa de salida: 256 $\rightarrow$ 182 clases (Dropout 0.3)

3.1.2. Parámetros del Modelo

- **Total de parámetros**: 3,803,798
- **Parámetros entrenables**: 3,803,798
- **Inicialización**: Aleatoria (Xavier/Kaiming)

3.2. Modelo Transfer Learning - EfficientNet-B0

3.2.1. Descripción de la Arquitectura

Se utilizó EfficientNet-B0, una arquitectura eficiente diseñada mediante búsqueda de arquitectura neuronal (NAS):

- **Arquitectura base**: EfficientNet-B0
- **Pesos preentrenados**: ImageNet (1000 clases)
- **Modificaciones**: Reemplazo de la capa de clasificación final para 182 clases
- **Estrategia de fine-tuning**: Todas las capas entrenables desde el inicio

3.2.2. Parámetros del Modelo

- **Total de parámetros**: 4,153,614
- **Parámetros entrenables**: 4,153,614
- **Inicialización**: Pesos de ImageNet + capa final aleatoria

4. Configuración del Entrenamiento

4.1. Hiperparámetros Comunes

Hiperparámetro	Valor
Épocas	15
Tamaño de batch	256
Learning rate inicial	0.001
Optimizador	AdamW
Weight decay	0.01
Scheduler	ReduceLROnPlateau
Factor de reducción	0.5

Cuadro 2: Hiperparámetros de entrenamiento

4.2. Función de Pérdida

Se utilizó `CrossEntropyLoss` con pesos balanceados por clase para manejar el desbalance en el dataset.

4.3. Hardware Utilizado

- **GPU**: NVIDIA Tesla T4
- **Plataforma**: Kaggle Notebooks
- **Framework**: PyTorch 2.x

5. Resultados

5.1. Métricas de Entrenamiento

5.1.1. Modelo Baseline

Métrica	Entrenamiento	Validación	Prueba
Pérdida final	3.8147	4.2891	-
Precisión máxima (%)	23.45	15.93	15.70
Tiempo de entrenamiento	45.3 min	-	-
Tiempo promedio por época	2.3 min	-	-

Cuadro 3: Resultados del modelo baseline

5.1.2. Modelo Transfer Learning

Métrica	Entrenamiento	Validación	Prueba
Pérdida final	0.0124	0.0453	0.0421
Precisión máxima (%)	99.87	99.23	99.07
Tiempo de entrenamiento	52.8 min	-	-
Tiempo promedio por época	2.6 min	-	-

Cuadro 4: Resultados del modelo con transfer learning

5.2. Comparación de Modelos

Característica	Baseline CNN	Transfer Learning
Arquitectura	CNN 5 capas	EfficientNet-B0
Parámetros	3,803,798	4,153,614
Pesos preentrenados	No	Sí (ImageNet)
Tiempo de entrenamiento	45.3 min	52.8 min
Precisión en validación	15.93 %	99.23 %
Precisión en prueba	15.70 %	99.07 %
Mejora relativa	**Baseline**	**+83.37 %**

Cuadro 5: Comparación exhaustiva entre modelos

5.3. Análisis de Convergencia

Modelo Baseline:

- Presentó dificultades para converger efectivamente
- Alta pérdida de validación (4.29) indica sobreajuste severo
- Gap significativo entre precisión de entrenamiento (23.45 %) y validación (15.93 %)

- Incapaz de capturar patrones complejos debido a la capacidad limitada del modelo

Modelo Transfer Learning:

- Convergencia rápida desde las primeras épocas
- Pérdida de validación muy baja (0.045) con mínimo sobreajuste
- Gap mínimo entre entrenamiento (99.87 %) y validación (99.23 %)
- Excelente generalización gracias a features preentrenadas de ImageNet

6. Análisis y Discusión

6.1. Ventajas del Transfer Learning

1. **Mejora dramática en precisión**: El modelo con transfer learning logró una precisión de 99.07 %, representando una mejora de 6.3x sobre el baseline (15.70 %)
2. **Mejor generalización**: Mínima diferencia entre precisión de entrenamiento y validación
3. **Convergencia más rápida**: Alcanzó alta precisión en pocas épocas
4. **Robustez**: Features preentrenadas capturan patrones visuales generales aplicables a hongos

6.2. Limitaciones del Modelo Baseline

1. **Insuficiente capacidad del modelo**: 182 clases requieren mayor complejidad
2. **Dataset limitado**: No suficientes datos para entrenar desde cero efectivamente
3. **Sobreajuste severo**: Alta varianza entre entrenamiento y validación
4. **Baja precisión final**: 15.70 % es apenas mejor que el azar aleatorio (0.55 %)

6.3. Importancia de los Pesos Preentrenados

Los pesos de ImageNet proporcionaron:

- Detectores de bordes y texturas de bajo nivel
- Representaciones de formas y patrones de nivel medio
- Features complejas adaptables al dominio de hongos
- Inicialización óptima que acelera la convergencia

6.4. Eficiencia Computacional

A pesar de tener más parámetros (4.15M vs 3.80M), el modelo de transfer learning:

- Requirió solo 7.5 minutos más de entrenamiento
- Logró resultados significativamente superiores
- Ofrece mejor relación costo-beneficio en términos de precisión por tiempo

7. Conclusiones

7.1. Principales Hallazgos

1. El transfer learning con EfficientNet-B0 es altamente efectivo para clasificación de hongos, alcanzando 99.07 % de precisión
2. El modelo baseline demuestra que entrenar desde cero con un dataset limitado es ineficiente para tareas de clasificación multiclase complejas
3. La diferencia de 83.37 puntos porcentuales valida el uso de modelos preentrenados en dominios especializados
4. EfficientNet-B0 proporciona un excelente balance entre precisión y eficiencia computacional

7.2. Trabajo Futuro

- Explorar arquitecturas más grandes (EfficientNet-B1 a B7)
- Implementar técnicas de ensemble para mejorar robustez
- Realizar análisis de interpretabilidad con Grad-CAM
- Evaluar el modelo en condiciones de campo con imágenes de menor calidad
- Incorporar metadata adicional (ubicación, estación del año)

7.3. Recomendaciones

Para proyectos similares de clasificación de imágenes:

1. Priorizar transfer learning sobre entrenamiento desde cero
2. Seleccionar arquitecturas eficientes como EfficientNet
3. Utilizar data augmentation extensivo
4. Implementar estrategias de fine-tuning gradual si es necesario
5. Monitorear métricas de sobreajuste durante el entrenamiento

8. Referencias

- Tan, M., & Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019.
- Deng, J., et al. (2009). ImageNet: A Large-Scale Hierarchical Image Database. CVPR 2009.
- Dataset: Mushroom Species Classification - Kaggle
- Framework: PyTorch 2.x - `https://pytorch.org`
- Biblioteca timm: PyTorch Image Models - `https://github.com/rwightman/pytorch-image-mod`