UNIVERSIDAD NACIONAL DE COLOMBIA

Detección de Imágenes: Real vs. AI

Clasificación Binaria con Transfer Learning (ResNet50)

Samuel Castro, Juan David Ramírez Ortiz

12 de diciembre de 2025

Proyecto de Deep Learning

¿Qué problema resolvemos?

El Desafío: Deepfakes y Generación AI

La proliferación de modelos generativos permite crear imágenes sintéticas indistinguibles a simple vista. Es necesario automatizar la distinción entre contenido auténtico y generado.

Impacto:

- Prevención de desinformación (Fake News).
- Validación de identidad y seguridad digital.
- Filtrado de contenido en redes sociales.

Solución Propuesta

Estrategia:

- **Tarea:** Clasificación Binaria de Imágenes.
- **Clases:** `REAL` (1) vs. `FAKE` (0).
- **Modelo:** Red Neuronal Convolucional (CNN).
- **Técnica:** Transfer Learning (Fine-tuning parcial).

Figura 1: Izquierda:AI Generated. Derecha:Real (CIFAKE).

Dataset y Preprocesamiento

Estructura del Dato:

- **Resolución Original:** 32×32 píxeles (RGB).
- **Resolución de Entrada (Upscaling):** 128×128 píxeles.
- **Normalización:** Rescale 1./255.

Set	**Cantidad**	**Clases**
Entrenamiento (Train)	100,000 imágenes	Balanced (50/50)
Prueba (Test)	20,000 imágenes	Balanced (50/50)
Total	**120,000 imágenes**	

CIFAR-10: Fundamentos y Justificación

¿Qué es?

Un dataset estándar de 60,000 imágenes a color clasificadas en **10 categorías** (aviones, gatos, coches, etc.).

¿Por qué se eligió este dataset?

- **Benchmark Universal:** Permite comparar métricas de rendimiento contra el .estado del arte".
- **Complejidad Balanceada:** Más desafiante que MNIST (dígitos), pero manejable para prototipado rápido.

Sobre la Baja Resolución (32×32 px):

- Diseñado para la **eficiencia computacional**.

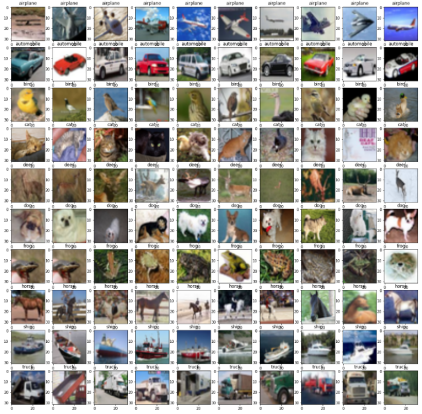


Figura 2: Muestra de las 10 clases del dataset CIFAR-10.

Arquitectura y Entrenamiento

Configuración del Modelo:

- **Base:** ResNet50 (Weights: ImageNet).
- **Congelamiento:** Todas menos las últimas 10 capas.
- **Cabezal (Head):**
  - GlobalAveragePooling2D
  - Dense (256, ReLU) + Dropout (0.5)
  - Salida: Dense (1, Sigmoid)
- **Optimizador:** Adam ($lr = 1e - 5$).
- **Loss:** Binary Crossentropy.

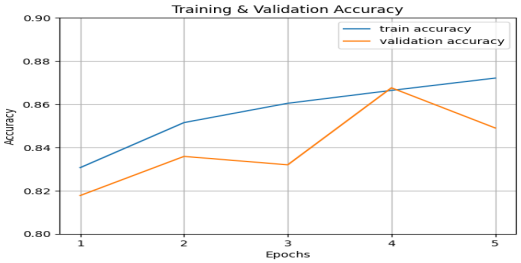
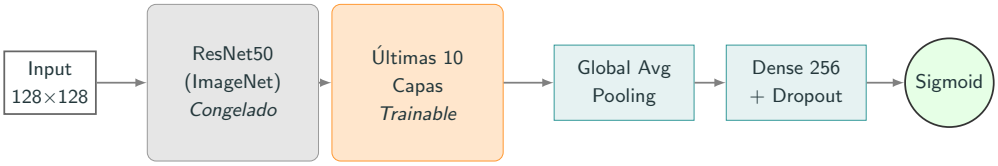


Figura 3: Curvas de Aprendizaje (5 Épocas). Max Val Acc: $\sim 86{,}7\,\%$

Arquitectura de la Red



- **Gris:** Extractor de características congelado.
- **Naranja:** Ajuste fino (Fine-tuning) al dominio sintético.
- **Verde:** Clasificador denso personalizado.

Optimización del Modelo

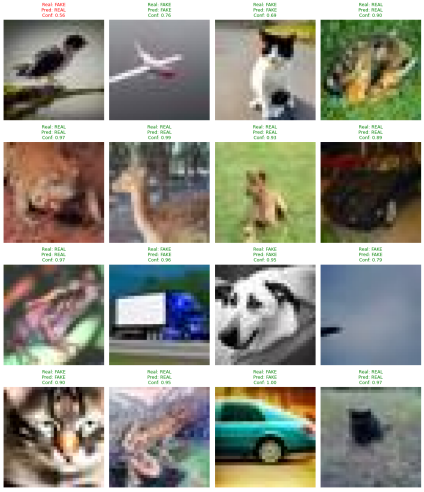
Comparativa estimada para despliegue en producción:

Formato	**Peso Aprox.**	**Inferencia**	**Caso de Uso**
Keras (.h5/.keras)	$\sim$ 95 MB	Lento	Entrenamiento
TorchScript / ONNX	$\sim$ 90 MB	Medio	Servidor Cloud
LiteRT (TFLite)	$\sim$ 25 **MB**	**Rápido**	**Móvil / Edge**

Cuadro 1: Reducción de tamaño mediante cuantización (Float16/Int8).

Resultados Visuales de Inferencia

Visualización de predicciones en el set de prueba (Grid Aleatorio):



Script de Inferencia Local

Flujo de Ejecución:

1. **Carga (Argparse):** Recibe la ruta de la imagen y del modelo '.tflite' por línea de comandos.
2. **Preprocesamiento:**
   - Redimensiona a 128×128.
   - Normaliza píxeles a rango $[0, 1]$.
   - Añade dimensión Batch $(1, 128, 128, 3)$.
3. **Motor LiteRT:** Usa `tf.lite.Interpreter` (sin cargar Keras completo) para máxima velocidad.
4. **Post-procesamiento:** Decodifica la salida Sigmoid y genera una gráfica comparativa con `matplotlib`.

Lógica Clave (Python):

```python
# 1. Preprocesamiento (Identico al Training)
image = image.resize((128, 128))
input_data = np.array(image) / 255.0

# 2. Inferencia TFLite
interpreter.allocate_tensors()
interpreter.set_tensor(idx, input_data)
interpreter.invoke() # Ejecutar
output = interpreter.get_tensor(idx)

# 3. Clasificacion Binaria
# Threshold: 0.5
if output[0][0] > 0.5:
    label = "REAL"   # Confianza = output
else:
    label = "FAKE"   # Confianza = 1 - output
```

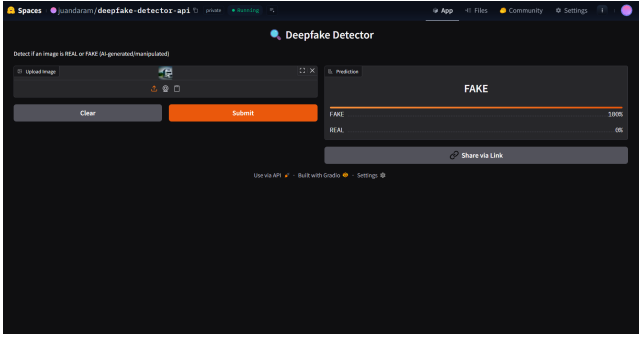
Comando de uso:

```
$ python src/local_predict/inference_script.py imgs/fake/5.jpg
```

Despliegue (Demo)

El modelo es accesible vía web para pruebas rápidas:

- **Plataforma:** HuggingFace Spaces.
- **Interfaz:** Gradio (Subir imagen $\rightarrow$ Probabilidad Real/Fake).



Resultados y Limitaciones

Métricas Finales (Época 5)

- **Train Accuracy:** 87.25 %
- **Val Accuracy:** 84.90 % (Pico en 86.76 %)
- **Train Loss:** 0.3026
- Convergencia estable en pocas épocas.

Limitaciones

- **Resolución:** El upscale de 32 a 128px puede introducir artefactos no deseados.
- **Tiempo:** Se entrenó solo por 5 épocas; podría mejorar con más tiempo.
- **Generalización:** Necesario probar con imágenes de alta calidad (no CIFAR/thumbnails).

Conclusiones

- **Eficacia de ResNet50:** A pesar de usar imágenes pequeñas escaladas, el modelo pre-entrenado logra casi un 87 % de precisión rápidamente.
- **Fine-Tuning:** Descongelar las últimas 10 capas fue crucial para adaptar las características de ImageNet al dominio sintético.
- **Escalabilidad:** Con 120,000 imágenes, el dataset es robusto, pero el modelo se beneficiaría de Data Augmentation más agresivo para reducir el overfitting leve.