Universitat Oberta de Catalunya

uoc.edu

Clasificación de Imágenes Mediante Transfer Learning de Redes Neuronales

Autor: Eleana Liscar Rey Quijada

Nombre Tutor/a de TF: Josep Mª Carmona Leyva

Profesor/a responsable de la asignatura: Joan M. Nuñez Do Rio

MU Ingeniería Informática Área de Inteligencia Artificial

Fecha Entrega: 14/01/2025

Índice

Introducción y contexto del problema

- El aumento exponencial de datos e información a la que se tiene acceso y que acaba almacenada en los dispositivos de uso personal resalta la necesidad de modelos de redes neuronales avanzados y eficientes para clasificar y organizar imágenes.

- Este trabajo propone optimizar un modelo de redes neuronales, como MobileViT, para ofrecer una solución eficiente que se adapte a las limitaciones de recursos de estos dispositivos, mejorando la experiencia de gestión de contenido digital de los usuarios.

Objetivos

Objetivo general:

- Aplicar un mecanismo basado en redes neuronales sobre un conjunto de imágenes reales para resolver un problema de clasificación de imágenes.

Objetivos específicos:

- Adaptar modelos preentrenados a tareas específicas: MobileVit
- Comprobar si este modelo basado en MobileViT mejora otros resultados anteriores de clasificación en conjuntos de datos públicos, utilizando métricas como precisión y pérdida.
- Analizar los tiempos requeridos para entrenar este modelo y evaluar su viabilidad en dispositivos con recursos limitados.
- Evaluar si se necesitan menos datos de aprendizaje para alcanzar un rendimiento competitivo en comparación con otras arquitecturas, para reducir la dependencia de datos extensivos.

Estado del Arte

Evolución de las Redes Neuronales

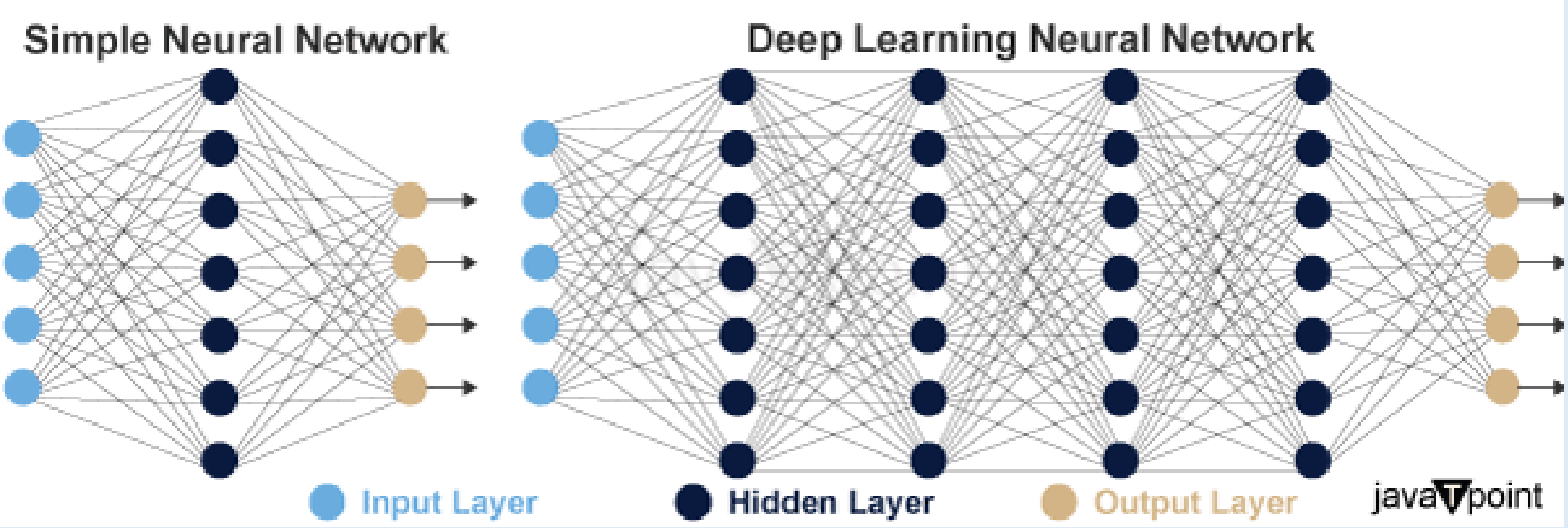


Figura 5 - Comparación entre SNN y DNN.(*DNN Machine Learning - Javatpoint*, s. f.)

CNN – AlexNet (Krizhevsky et al., 2012)

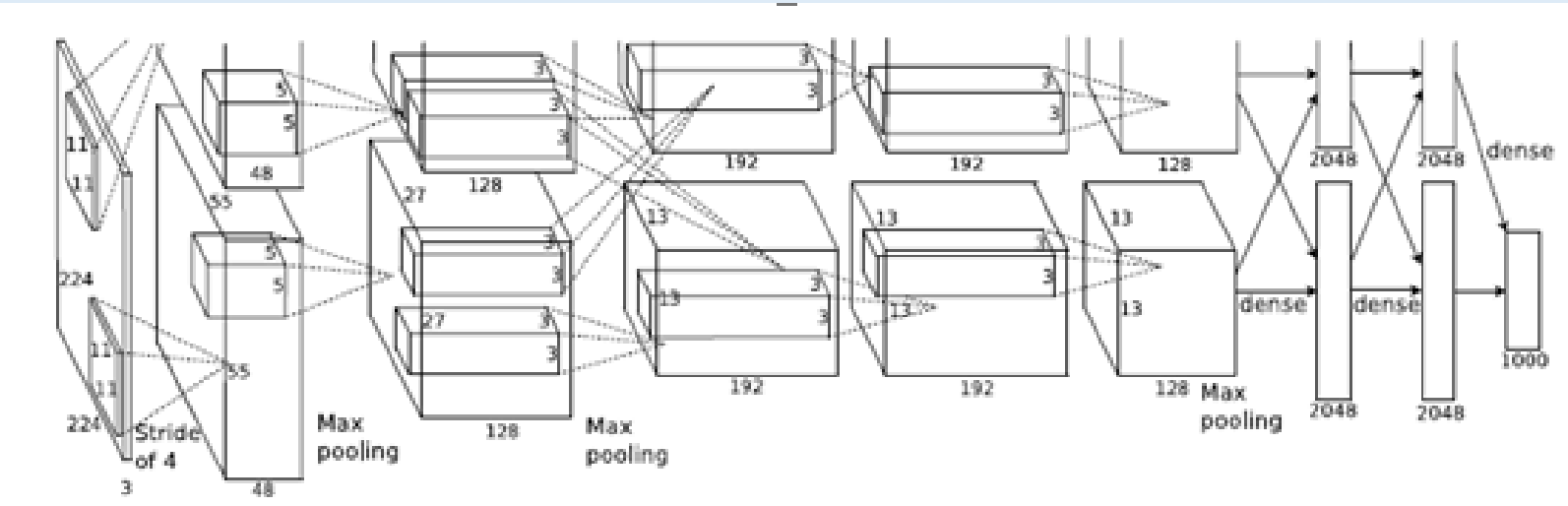
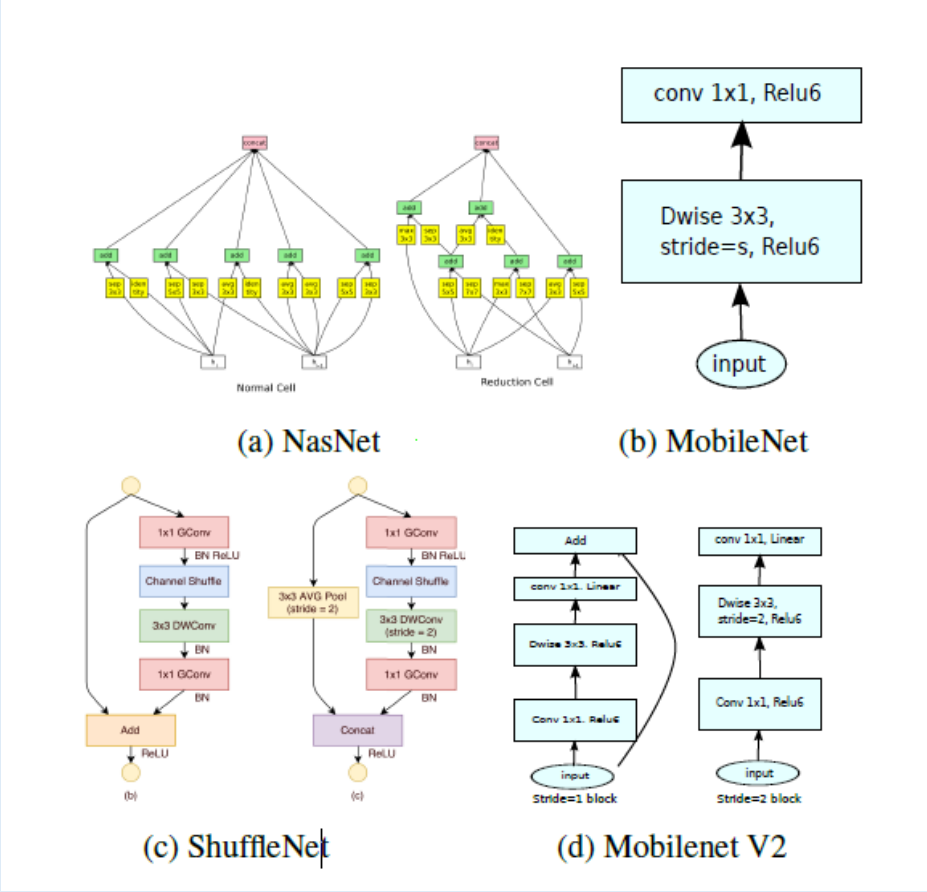


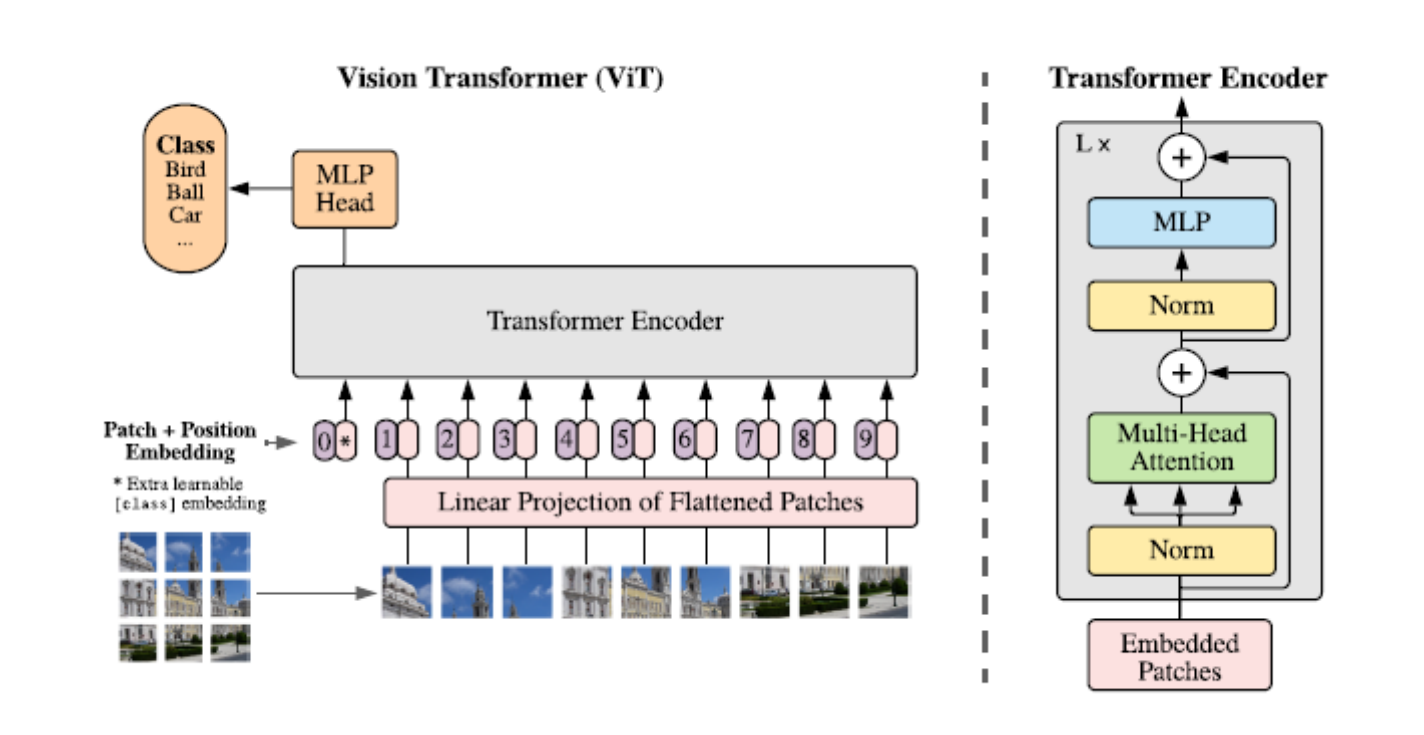
Figura 4 – ImageNet CNN (Krizhevsky et al., 2012)

La Figura de arriba, describe la arquitectura de la red neuronal convolucional (CNN) presentada por Krizhevsky et al. (2012), utilizada en su modelo AlexNet para clasificación de imágenes y fue un avance clave en 2012 para mejorar la eficiencia en tareas de visión por computadora, aprovechando la computación distribuida en GPUs para acelerar el entrenamiento y la inferencia.

Comparación de Bloques convolucionales de diferentes Arquitecturas. (Sandler et al., 2019)

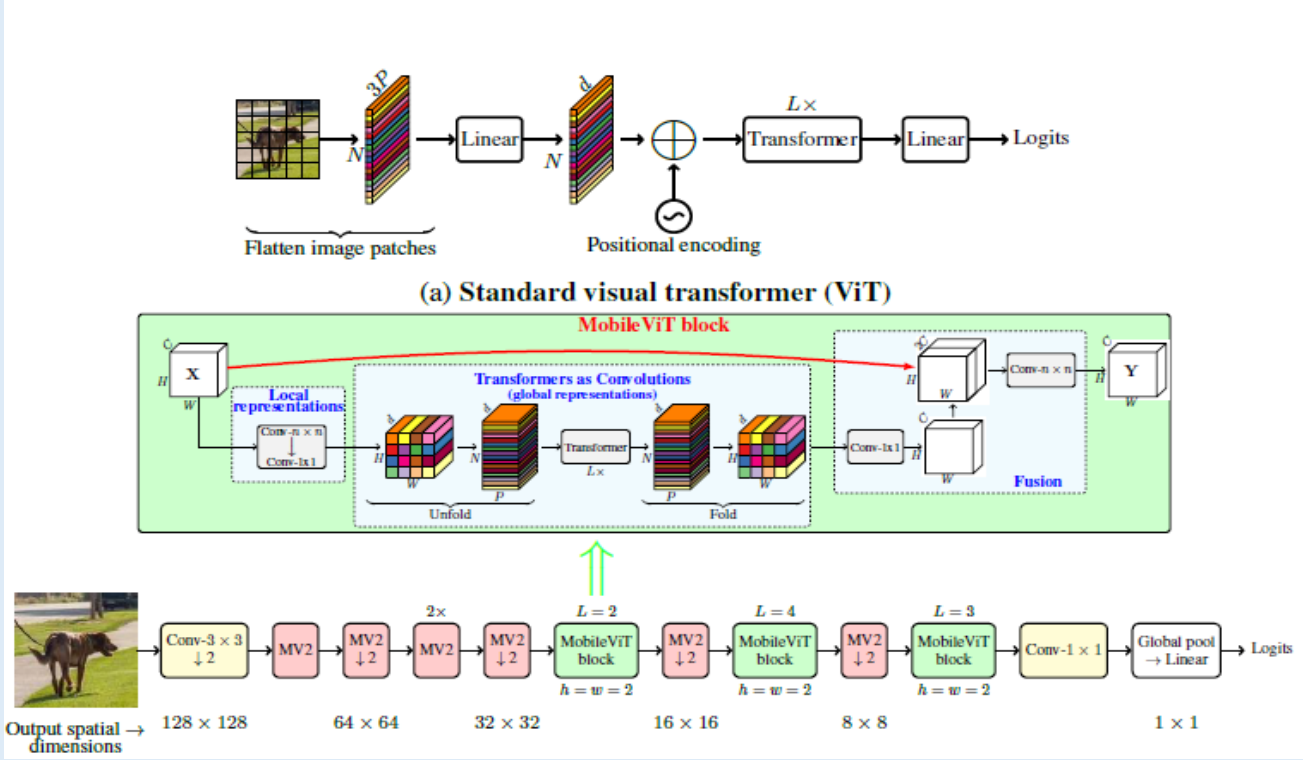


Vision Transformers



Arquitecturas Transformers y Vision Transformers.(Dosovitskiy et al., 2021)

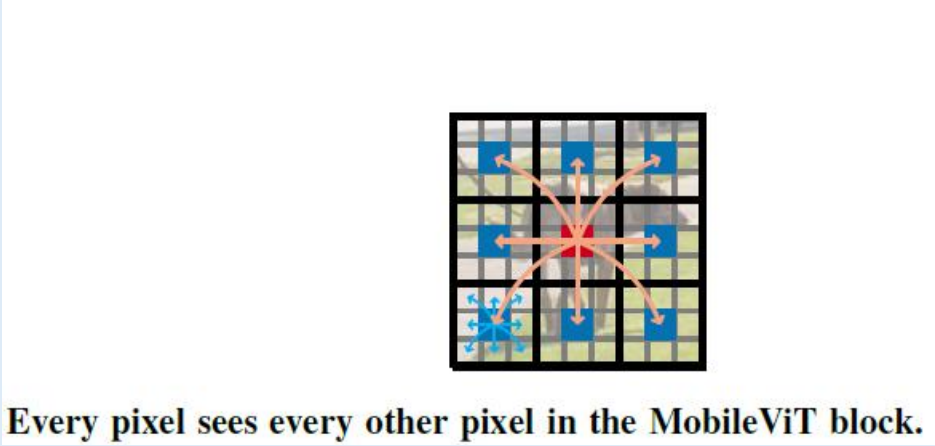
MobileViT



Propiedades del MobileViT .(Mehta & Rastegari, 2022a) en comparación con modelos CNN ligeros.

- MobileViT demuestra que es más capaz de adaptarse a diferentes tareas manteniendo una alta precisión en comparación con modelos ligeros de CNN.
- Se enumeran los parámetros de la red para diferentes extractores de características utilizados en el SSDLite, que es una red de detección de objetos.
- La comparación se realiza sobre el dataset MS-COCO, destacando que MobileViT logra resultados superiores a ViT.

Mobile ViT



Every pixel sees every other pixel in the MobileViT block.

MobileViT integra relaciones locales y globales para procesar imágenes completas, maximizando su capacidad de comprensión visual.

Funcionamiento del bloque MobileViT

- Conexión entre píxeles
- Codificación previa con convoluciones
- Ventaja de los transformadores
- Representación visual

Comparación de Modelos

MobileViT frente a arquitecturas tradicionales

- **MobileViT vs. MobileNet**

MobileViT supera en precisión (78.4% vs. -75%) con parámetros similares.

- **MobileViT vs. Vision Transformers (ViT)**

MobileViT es más ligero y eficiente en dispositivos móviles.

- **Relevancia en la práctica**

MobileViT combina precisión y eficiencia para aplicaciones reales.

Materiales y metodología

- Entorno de trabajo
  - Hardware:
    - Portátil personal: Procesador: Intel Core , 16 GB DDR4, 512 GB, Intel Iris Xe Graphics integrada, SO Windows 11 Pro de 64 bits.
    - Procesamiento: Google Colab Pro con acceso a GPUs NVIDIA o TPUs y hasta 334 GB de RAM.
    - Almacenamiento: Google Drive con 100 GB.

  - Software:
    - Entorno de desarrollo: Google Colab.
    - Lenguaje de programación Python 3.
    - Framework PyTorch, que facilita la implementación y personalización de modelos de aprendizaje profundo.
    - Librerías auxiliares: torchvision, matplotlib, numpy, y scikit-learn.

Conjuntos de datos utilizados

Tiny-ImageNet-200

- Contiene 200 clases con 500 imágenes de entrenamiento por clase, 50 de validación y 50 de prueba.
- Las imágenes son de baja resolución (64x64 píxeles).

Dataset propio de clasificación binaria, que llamamos “text_recognition”

- Se combinaron los dataset públicos MSRA-TD500 y Flickr8k y se extrajeron 200 muestras aleatorias de cada conjunto.
- Se generó un conjunto de datos con las clases text y no_text.
- Se le aplicó redimensionamiento a 224x224 pixeles

MSRA-TD500

- Contiene 500 imágenes.
- 300 imágenes que se utilizan para el conjunto de entrenamiento y 200 imágenes se reservan para pruebas.
- Las imágenes incluyen texto tanto en chino como en inglés.

Flickr8k

- Compuesto por 8,000 imágenes acompañadas de descripciones textuales generadas por humanos.
- Estas imágenes son diversas y capturan una amplia gama de actividades, objetos y escenas.
- Se utilizó para extraer muestras para utilizarlo como clase de imágenes que no contienen texto.

Ajustes de Hiperparámetros y Técnicas Aplicadas

Optimización: Se hace uso del optimizador Adam con una tasa de aprendizaje baja ajustada a 0.0001 para reducir el riesgo de destruir la información adquirida durante el preentrenamiento.

La normalización de datos se evaluó basada en:

Estadísticas de ImageNet:

- Media: [0.485, 0.456, 0.406] (promedio de valores RGB).
- Desviación estándar: [0.229, 0.224, 0.225].

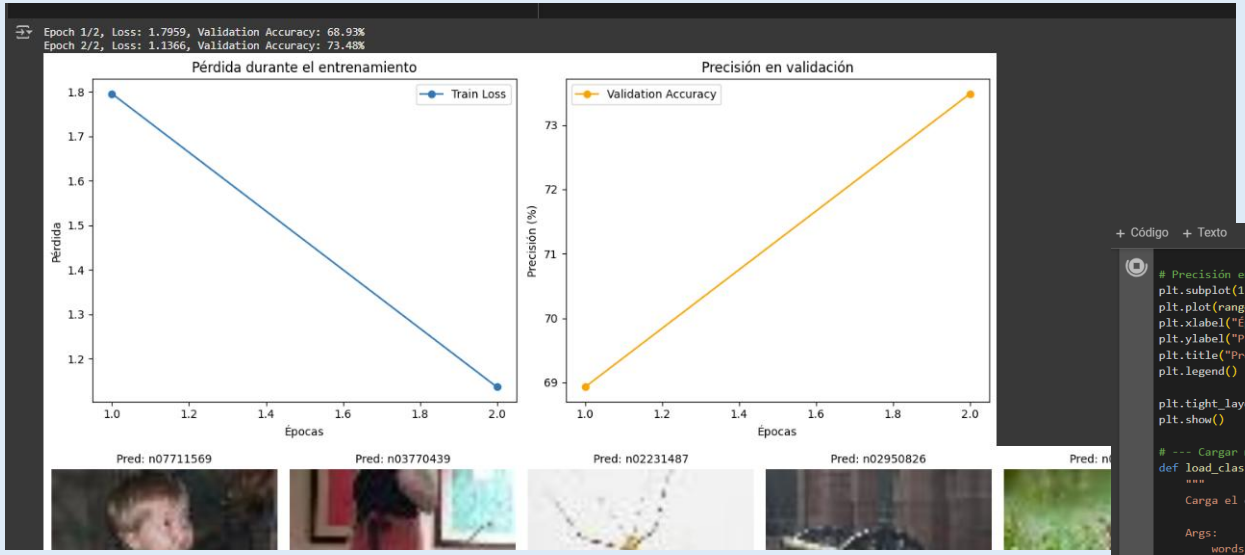
Estándar (0.5,), (0.5,).

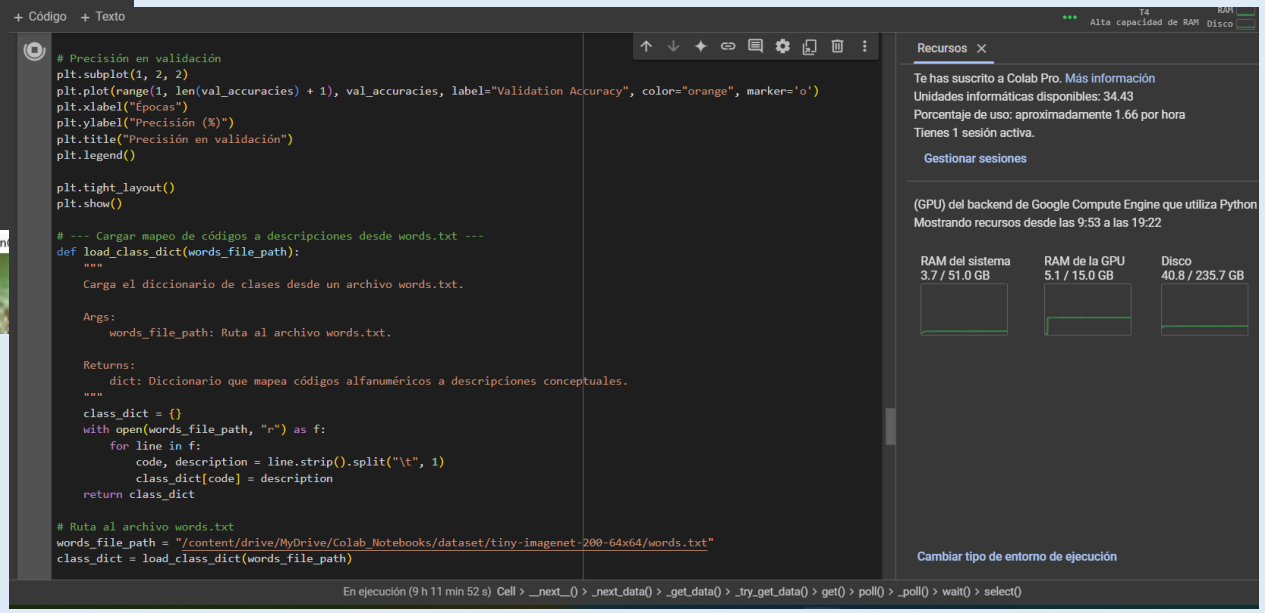
Data augmentation: Se realizan transformaciones como rotaciones para mejorar la generalización.

División balanceada: Separación del 80% para entrenamiento y 20% para validación.

Entrenamiento y evaluación

Primer experimento con el modelo MobileViT y el dataset tiny-imagenet-200





```
# Precisión en validación
plt.subplot(1, 2, 2)
plt.plot(range(1, len(val_accuracies) + 1), val_accuracies, label="Validation Accuracy", color="orange", marker='o')
plt.xlabel("Épocas")
plt.ylabel("Precisión (%)")
plt.title("Precisión en validación")
plt.legend()

plt.tight_layout()
plt.show()

# --- Cargar mapeo de códigos a descripciones desde words.txt ---
def load_class_dict(words_file_path):
    """
    Carga el diccionario de clases desde un archivo words.txt.

    Args:
        words_file_path: Ruta al archivo words.txt.

    Returns:
        dict: Diccionario que mapea códigos alfanuméricos a descripciones conceptuales.
    """
    class_dict = {}
    with open(words_file_path, "r") as f:
        for line in f:
            code, description = line.strip().split("\t", 1)
            class_dict[code] = description
    return class_dict

# Ruta al archivo words.txt
words_file_path = "/content/drive/MyDrive/Colab_Notebooks/dataset/tiny-imagenet-200-64x64/words.txt"
class_dict = load_class_dict(words_file_path)
```

Recursos

Te has suscrito a Colab Pro. Más información
Unidades informáticas disponibles: 34.43
Porcentaje de uso: aproximadamente 1.66 por hora
Tienes 1 sesión activa.
Gestionar sesiones

(GPU) del backend de Google Compute Engine que utiliza Python
Mostrando recursos desde las 9:53 a las 19:22

RAM del sistema 3.7 / 51.0 GB
RAM de la GPU 5.1 / 15.0 GB
Disco 40.8 / 235.7 GB

Cambiar tipo de entorno de ejecución

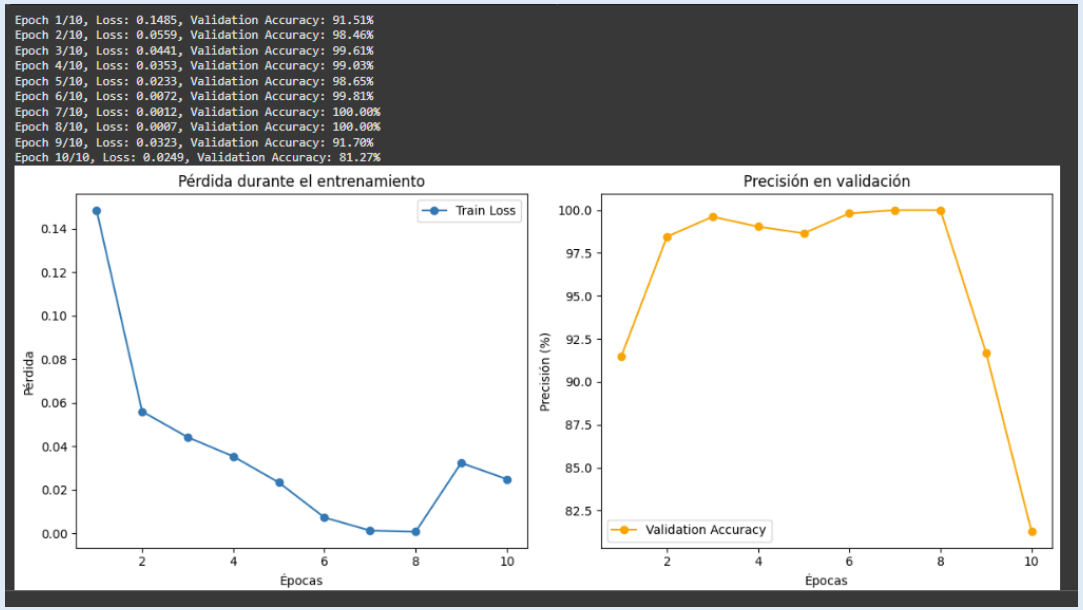
Resultados de perdida 1.1366 y precisión 73.48%.

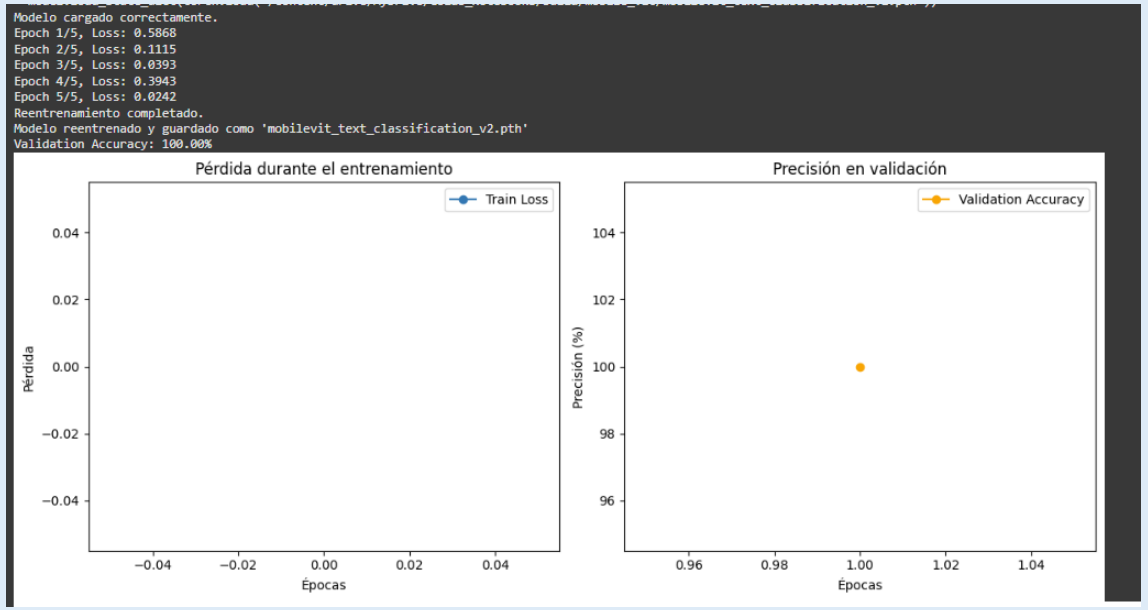
Intentamos entrenar en 10 épocas y finalizamos manualmente ejecución tras 9 horas en proceso y un consumo de mas de 40 unidades informáticas.

Entrenamiento y evaluación

Segundo experimento con el modelo MobileViT y el dataset text_recognition

- Obtenemos una precisión del 81,27 % en un primer entrenamiento de 10 épocas y 100% al reentrenarlo con el mismo set.

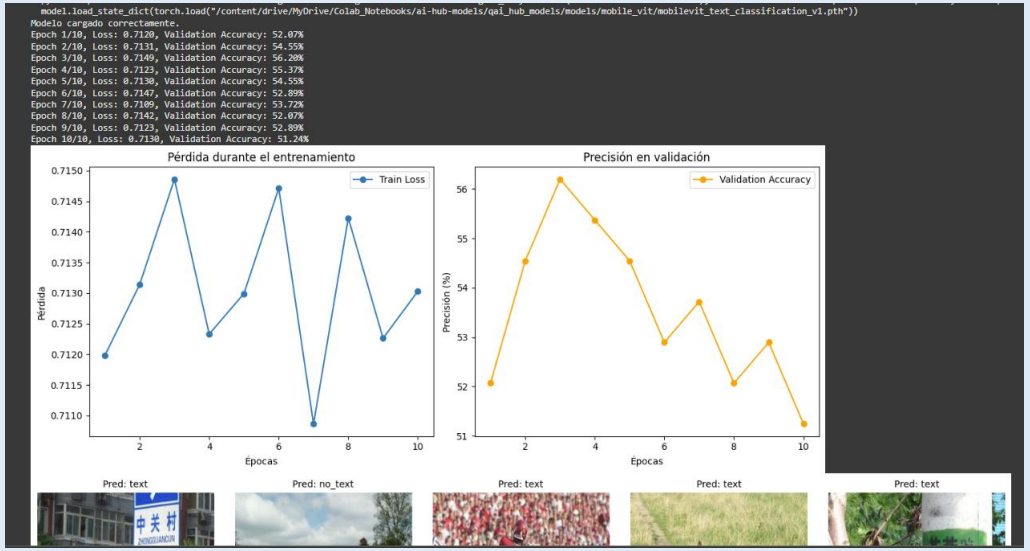


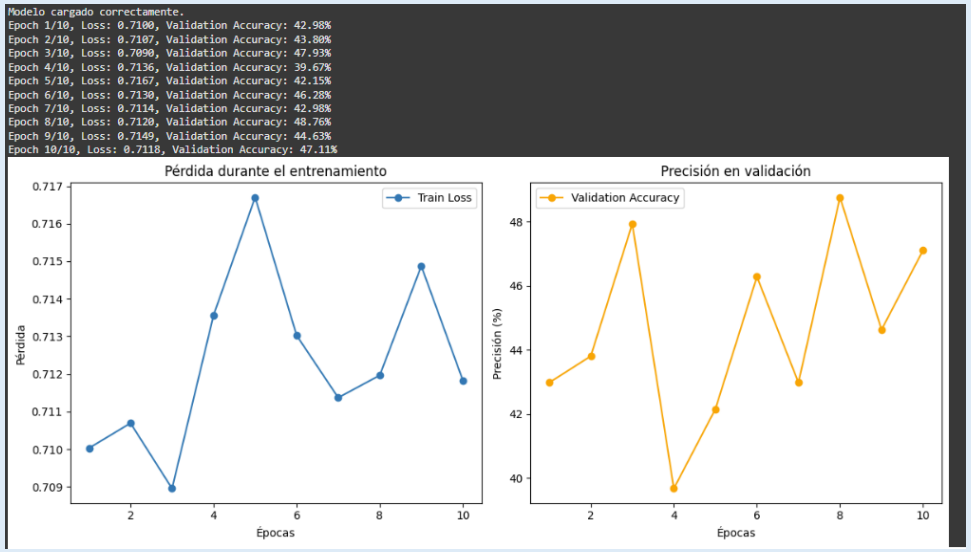


Entrenamiento y evaluación

Tercer experimento con el modelo con el modelo MobileViT y el dataset

text_recognition tras balancear las clases y aplicar auentacion de datos:





Precisión del modelo al 1 entrenamiento: 51.24%

Precisión del modelo al 2do entrenamiento: 47.1%

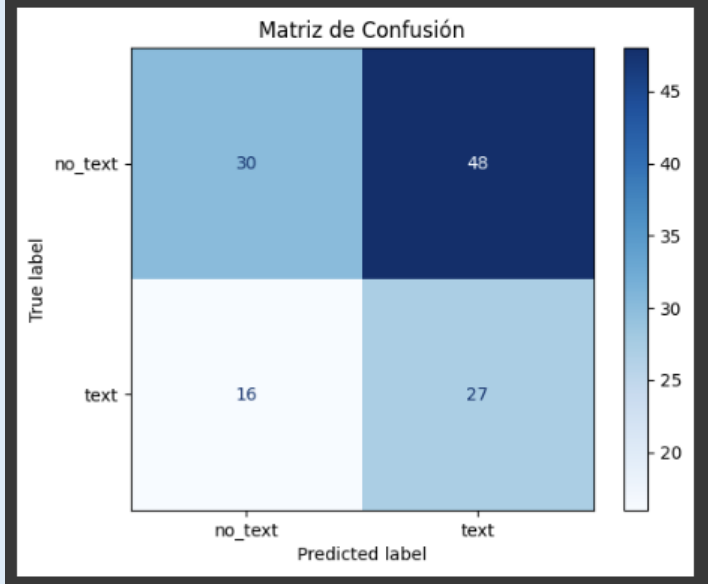
Matriz de Confusión

Precisión de 47.11% en el set de validación tras 10 épocas.

```python
# Se usa una matriz de confusión para visualizar cómo el modelo
# clasifica las imágenes

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Matriz de confusión
cm = confusion_matrix(all_labels, all_predictions)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classes)
disp.plot(cmap=plt.cm.Blues)
plt.title("Matriz de Confusión")
plt.show()
```



Correctas como 'no text'

30 imágenes correctamente clasificadas.

Correctas como 'text'

27 imágenes correctamente clasificadas.

Falsos positivos

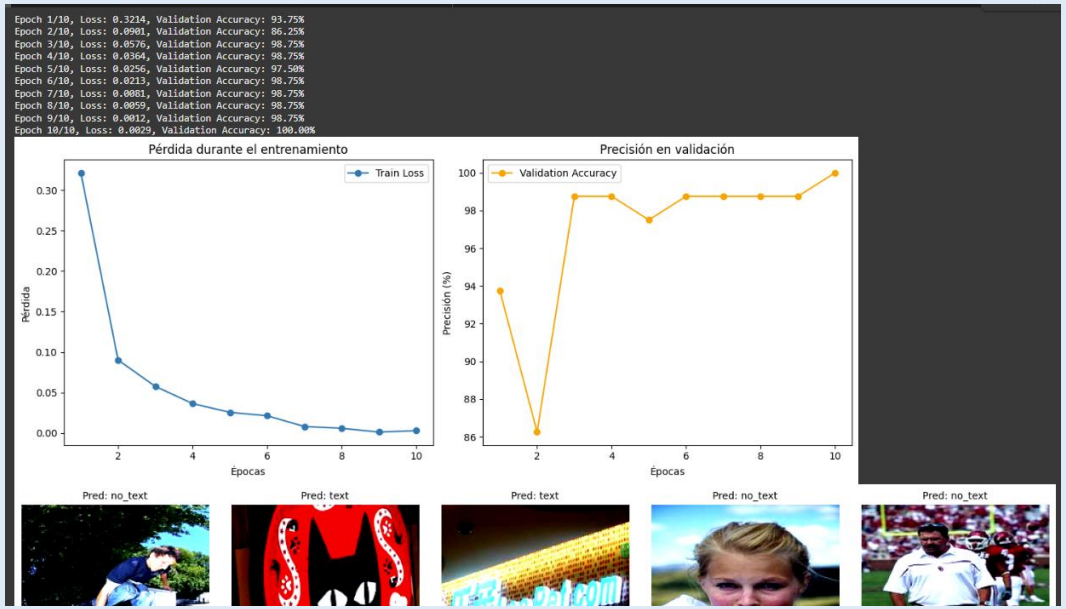
48 imágenes clasificadas incorrectamente como 'text’.

Falsos negativos

16 imágenes clasificadas incorrectamente como 'no text'.

Entrenamiento y evaluación

Cuarto experimento con el modelo Mobile ViT y el dataset text_recognition, optimizando la normalización



- Obtenemos malos resultados con:
  - Perdida: 0.0029
  - Precisión: 100.00%

Conclusiones y Trabajos futuros

- El modelo ajustado de MobileViT ha mostrado un bajo rendimiento, con un número significativo de errores en las predicciones en el conjunto de validación, aunque se trabajó en el balance del dataset, se aplicaron técnicas de aumento de datos y se ajustaron hiperparámetros como la tasa de aprendizaje con el objetivo de mejorar la capacidad del modelo para generalizar y reducir estos errores, pero no conseguimos buenos resultados.

- Para las futuras iteraciones del proyecto, sería interesante probar otros conjuntos de datos, otros modelos mas avanzados y otros ajustes de transformación e hiperparámetros.