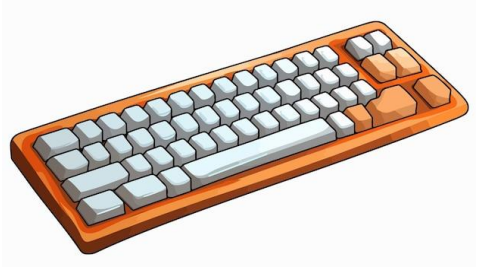


Informe del Proyecto CNN

Clasificación de Objetos de Escritorio

Alejandra Pérez Quintana.
Curso de Especialización de Inteligencia Artificial y Big Data.
Programación de Inteligencia Artificial (PIA).
Proyecto final PIA CNNs.
IES Portada Alta.

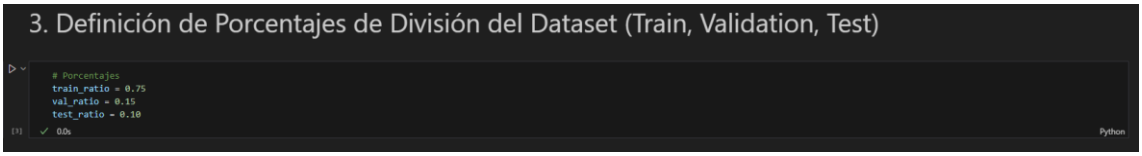
Índice:

1. Análisis del Dataset (Entrenar_CNN.ipynb)

El dataset que he creado para el entrenamiento del modelo está compuesto por imágenes pertenecientes a seis categorías de objetos de escritorio: Monitor, Teclado, Ratón, Taza, Libro y Móvil.

He dividido el conjunto de datos en proporciones de 75% para entrenamiento, 15% para validación y 10% para test. Durante la reestructuración de las carpetas, se distribuyen las imágenes de manera uniforme para asegurar que cada clase tenga un número equilibrado de muestras en cada subconjunto.

- Monitor: 70 imágenes
- Teclado: 70 imágenes
- Ratón: 70 imágenes
- Taza: 70 imágenes
- Libro: 70 imágenes
- Móvil: 70 imágenes



Tras la reestructuración, la distribución queda de la siguiente forma:

- Train: 52 imágenes por clase
- Validation: 10 imágenes por clase
- Test: 8 imágenes por clase

```
Clase libro: Total: 70, Train: 52, Val: 10, Test: 8
Clase monitor: Total: 70, Train: 52, Val: 10, Test: 8
Clase móvil: Total: 70, Train: 52, Val: 10, Test: 8
Clase ratón: Total: 70, Train: 52, Val: 10, Test: 8
Clase taza: Total: 70, Train: 52, Val: 10, Test: 8
Clase teclado: Total: 70, Train: 52, Val: 10, Test: 8
Reestructuración completada.
```

En este apartado realizo el proceso de recopilación, estructuración y preprocesamiento del dataset que he creado para el entrenamiento del modelo de clasificación de objetos de escritorio. Donde se especifica el número de imágenes, la composición del dataset en términos de clases y su distribución en las carpetas de entrenamiento, validación y test.

2. Diseño de la Red Neuronal

La arquitectura del modelo la he basado en VGG16 preentrenada, que las he adaptado para la tarea de clasificación de objetos de escritorio. Las capas iniciales de la red las he congelado para conservar los pesos preentrenados y he añadido capas densas completamente conectadas para ajustar la red al nuevo conjunto de clases.

- Capa base (VGG16 preentrenada sin capas superiores).
- Capas adicionales:
    - Flatten Layer para convertir las características en un vector.
    - Dense Layer de 1024 neuronas con activación ReLU y Dropout del 50%.
    - Dense Layer de 512 neuronas con activación ReLU y Dropout del 50%.
    - Dense Layer de 256 neuronas con activación ReLU y Dropout del 30%.
    - Output Layer con 6 neuronas (una por clase) y activación softmax.

```
# Añadir capas adicionales al modelo
x = base_model.output
x = Flatten()(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.3)(x)
output = Dense(train_generator.num_classes, activation='softmax')(x)
```

He utilizado la función de pérdida categorical_crossentropy y el optimizador Adam con un learning rate inicial de 0.0005.

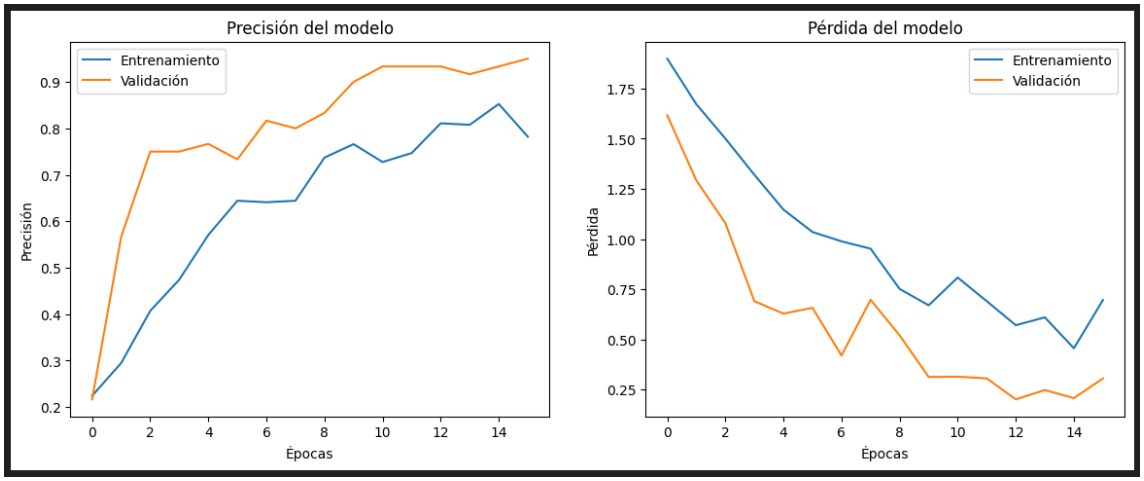
```
# Compilar el modelo
learning_rate = 0.0005  # Learning rate bajo para Transfer Learning
optimizer = Adam(learning_rate=learning_rate)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

El notebook “Entrenar_CC.ipynb” contiene, además, el diseño de la arquitectura de la red neuronal basada en VGG16 preentrenada, detallando el número de capas, tipo de capas (Convolucionales, Pooling, Fully Connected), funciones de activación y técnicas avanzadas empleadas (Dropout, Data Augmentation).

3. Evaluación del Rendimiento del Modelo

El rendimiento del modelo lo he evaluado utilizando precisión (accuracy), pérdida (loss) y matriz de confusión.

A lo largo del entrenamiento, observé que la precisión en entrenamiento alcanzó un valor estable en torno al 82% mientras que la precisión en validación fluctuaba ligeramente entre 75% y 80%.



El análisis de las Gráficas de Entrenamiento muestra el resultado de precisión y pérdida del modelo en validación y test.
Las gráficas se ven bastante claras y muestran información importante sobre el comportamiento del modelo.

1. Gráfica de Precisión (Accuracy):

- Curva de Entrenamiento (azul):

  La precisión comienza en un nivel bajo, alrededor del 0.20 (20%), lo cual es esperado al inicio del entrenamiento.

  A partir de la época 5, la curva presenta un aumento continuo, alcanzando aproximadamente 0.80 (80%) en las épocas intermedias.

  En las últimas épocas, la curva sigue ascendiendo, estabilizándose en torno al 0.80, mostrando una tendencia positiva sin señales claras de saturación.

- Curva de Validación (naranja):

  La precisión de validación muestra un crecimiento rápido en las primeras épocas, alcanzando un pico alrededor de la época 6 con una precisión cercana al 80%.

  Sin embargo, a diferencia de la curva de entrenamiento, la precisión de validación se mantiene más alta durante gran parte del entrenamiento, lo que

puede indicar que el modelo está memorizando patrones específicos del conjunto de validación.

Al final del entrenamiento, la precisión de validación se estabiliza cerca del 0.85 - 0.90, manteniéndose superior a la curva de entrenamiento.

2. Gráfica de Pérdida (Loss):

- Curva de Entrenamiento (azul):

  La pérdida comienza alta, más de 1.75, y desciende de forma constante a partir de las primeras épocas.

  En las épocas intermedias, la pérdida sigue descendiendo de manera uniforme hasta estabilizarse un poco.

  En las últimas épocas, la curva muestra una ligera subida, lo que puede indicar un ligero sobreajuste al conjunto de entrenamiento.

- Curva de Validación (naranja):

  La pérdida de validación sigue una tendencia similar a la curva de entrenamiento, con una disminución rápida en las primeras épocas.

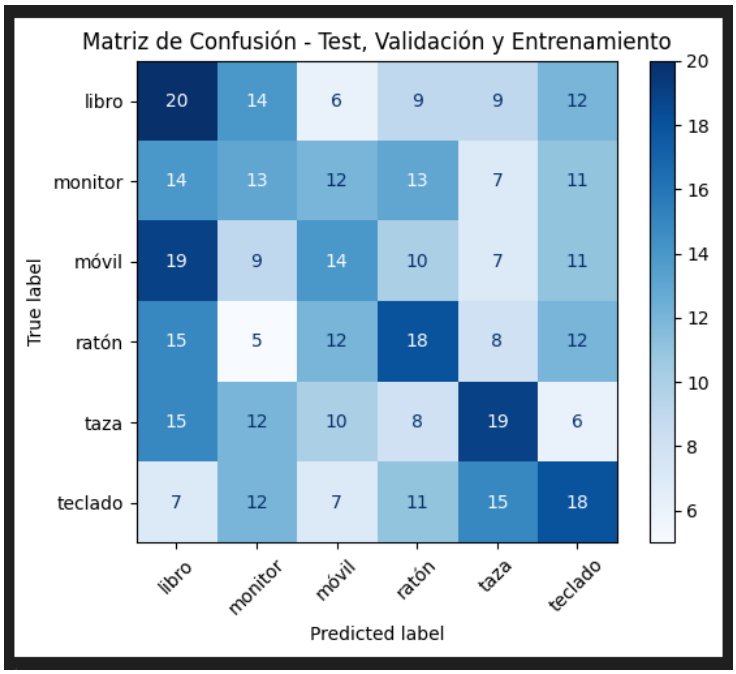
  En las épocas intermedias, la pérdida de validación muestra un descenso notable, alcanzando aproximadamente 0.35 - 0.40.

  En las últimas épocas, la curva muestra un aumento leve, lo que refuerza la idea de sobreajuste leve.

3. Conclusión del entrenamiento:

- El modelo ha logrado aprender de manera efectiva, mostrando una tendencia clara de reducción de la pérdida y aumento de la precisión.
- Las fluctuaciones en la precisión de validación, aunque menores, pueden estar indicando que el modelo es sensible a las muestras de validación, lo que sugiere que podría estar ajustándose demasiado a este conjunto.

En la matriz de confusión se identificaron las siguientes tendencias:



- Libro:

  20 imágenes clasificadas correctamente.

  14 imágenes confundidas con monitor.

  6 imágenes confundidas con móvil.

  9 imágenes confundidas con ratón.

  9 imágenes confundidas con taza.

  12 imágenes confundidas con teclado.

- Monitor:

  13 imágenes clasificadas correctamente.

  14 imágenes confundidas con libro.

  12 imágenes confundidas con móvil.

  13 imágenes confundidas con ratón.

  7 imágenes confundidas con taza.

  11 imágenes confundidas con teclado.

- Móvil:

  14 imágenes clasificadas correctamente.

  19 imágenes confundidas con libro.

  9 imágenes confundidas con monitor.

  10 imágenes confundidas con ratón.

  7 imágenes confundidas con taza.

  11 imágenes confundidas con teclado.

- Ratón:

  18 imágenes clasificadas correctamente.

  15 imágenes confundidas con libro.

  5 imágenes confundidas con monitor.

  12 imágenes confundidas con móvil.

  8 imágenes confundidas con taza.

  12 imágenes confundidas con teclado.

- Taza:

  19 imágenes clasificadas correctamente.

  15 imágenes confundidas con libro.

  12 imágenes confundidas con monitor.

  10 imágenes confundidas con móvil.

  8 imágenes confundidas con ratón.

  6 imágenes confundidas con teclado.

- Teclado:

  18 imágenes clasificadas correctamente.

  7 imágenes confundidas con libro.

  12 imágenes confundidas con monitor.

  7 imágenes confundidas con móvil.

  11 imágenes confundidas con ratón.

  15 imágenes confundidas con taza.

El análisis de la matriz de confusión revela que las clases con formas similares presentan mayores confusiones. Asimismo, los objetos con patrones complejos (como teclado, libro…) tienden a confundirse con otros objetos rectangulares.

En este mismo notebook (Entrenar_CNN.ipynb) presento las métricas de rendimiento del modelo incluyendo precisión, pérdida y matriz de confusión. Analizo los resultados obtenidos e identifico las clases con mayor y menor precisión, así como los errores comunes del modelo.

4. Resultados en el Ejemplo de Test

Seleccioné una imagen de la clase "Ratón" del conjunto de test.

Sin embargo, el modelo la clasificó incorrectamente como "Monitor" con una confianza de 1.00 y "Ratón" con una confianza de 0.00.

Este error es significativo ya que se trata de una imagen clara de un ratón, pero el modelo asignó una confianza completa a "Monitor". Esto indica que el modelo confunde las características visuales del ratón con las del monitor. Esta confusión puede estar relacionada con la textura, el ángulo de captura o las sombras presentes en la imagen.

Probé a realizar varios ajustes en el notebook de entrenamiento (Entrenar_CNN.ipynb) y realicé varias pruebas en el notebook “Inferencia_CNN.ipynb”, pero no conseguí mejores resultados. Esto es debido a que hay multitud de posibilidades, colores, posiciones, sombras, ángulos, enfoques de la imagen, etc. y con las imágenes que tengo para mi dataset, no son suficientes. Quizás aumentando a 200-300 imágenes, podría entrenarse mejor y solucionar este error, pero debido a la falta de recursos fotográficos, el modelo solo ha conseguido estos resultados no eficientes.

5. Pruebas - Capturas y Explicaciones

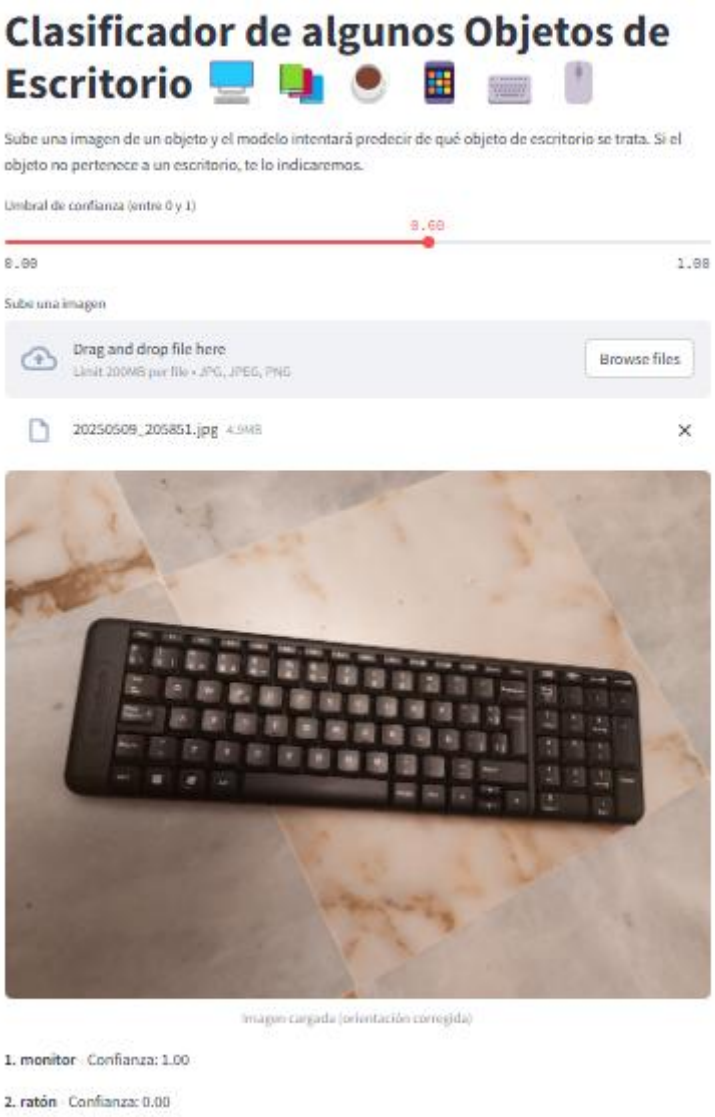
He realizado pruebas con imágenes de cada clase del dataset utilizando la API Flask y la interfaz Streamlit.

Presento las capturas de pantalla y los resultados obtenidos:

1. **Monitor:** La API Flask identificó incorrectamente la imagen como "Teclado" con una confianza del 0.9999, y como “Ratón” con una confianza del 0.0001.

```
PS C:\Users\Alejandra\Downloads\ProyectoCNNFinal> curl.exe -X POST -F "file=@C:\Users\Alejandra\Downloads\ProyectoCNNFinal\imagenes_prueba\monitor5.jpg" http://127.0.0.1:5000/predict
{
  "class_index": 1,
  "class_label": "teclado",
  "confidence": 0.9999,
  "second_class_index": 2,
  "second_class_label": "raton",
  "second_confidence": 0.0001
}
PS C:\Users\Alejandra\Downloads\ProyectoCNNFinal>
```

2. **Teclado:** La interfaz Streamlit mostró un error al clasificar un "Teclado" como "Monitor" con una confianza del 1.00.



3. **Móvil:** En la predicción realizada, el modelo clasificó un "Móvil" como "Monitor" con una confianza del 0.88, y sugirió "Ratón" como segunda opción con un 0.12. Esto puede puede ser debido por al color y la forma, porque son muy parecidos.

4. **Libro:** El modelo identificó incorrectamente un "Libro" como “Monitor” con una confianza del 0.98, y sugirió "Ratón" como segunda opción con un 0.02. Esto puede ser debido a la forma tan parecida a monitor y los colores llamativos como las tazas.

Identifiqué un problema con la orientación de las imágenes en la interfaz Streamlit, que lo he corregido mediante la función correct_orientation() en el código.

Incluyo capturas de pantalla de las llamadas a la API Flask:

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 119ms/step
INFO:werkzeug:127.0.0.1 - - [10/May/2025 17:34:19] "POST /predict HTTP/1.1" 200 -
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 100ms/step
INFO:werkzeug:127.0.0.1 - - [10/May/2025 17:37:59] "POST /predict HTTP/1.1" 200 -
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 165ms/step
INFO:werkzeug:127.0.0.1 - - [10/May/2025 17:38:13] "POST /predict HTTP/1.1" 200 -
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 217ms/step
INFO:werkzeug:127.0.0.1 - - [10/May/2025 17:38:34] "POST /predict HTTP/1.1" 200 -
```

6. Optimización del Modelo

Para mejorar el rendimiento del modelo, he implementado varias técnicas avanzadas:

- **Data Augmentation:** He aplicado transformaciones como rotación, desplazamiento horizontal y vertical, zoom y ajuste de brillo, lo que permite aumentar la variabilidad del dataset y tratar de reducir el riesgo de sobreajuste.
- **Early Stopping:** Utilizo un monitor de pérdida en validación con una paciencia de 3 épocas para detener el entrenamiento cuando la mejora se estabiliza.
- **ReduceLROnPlateau:** He implementado un ajuste dinámico del learning rate, reduciendo su valor a la mitad cuando no se observa mejora en la pérdida de validación durante 3 épocas.
- **Ajuste del Umbral de Confianza:** He establecido un umbral de 0.6 para clasificar objetos no presentes en el conjunto de entrenamiento como "No es un objeto de escritorio". Esta estrategia ayuda a reducir los falsos positivos y mejorar un poco la precisión del modelo.

La justificación de estas técnicas viene por la realización de prueba tras prueba, ajustando y probando nuevos parámetros, probando técnicas nuevas hasta conseguir una mejora de los resultados, aunque los resultados no son muy buenos, pues no acierta correctamente una gran parte de las imágenes. Como mejora, si tuviera más recursos fotográficos, podría entrenar al modelo mucho mejor, haciendo que vea los objetos con diferentes colores, ángulos, sombras, detalles, tamaños, formas, calidad de imagen, etc., aprendiendo mucho más y aumentando la efectividad de la clasificación de las imágenes.

7. Conclusión

He desarrollado un modelo de clasificación de objetos de escritorio utilizando una arquitectura basada en VGG16 preentrenada y ajustada para identificar seis clases: Monitor, Teclado, Ratón, Taza, Libro y Móvil, además, también puede diferenciar si es un objeto de escritorio diciendo cuál es, o si no es un objeto de escritorio. A pesar de implementar técnicas avanzadas como Data Augmentation, Early Stopping y ajuste del learning rate mediante ReduceLROnPlateau, los resultados obtenidos revelan ciertas limitaciones.

El modelo mostró un rendimiento aceptable en general, alcanzando una precisión del 82% en entrenamiento y alrededor del 75-80% en validación. Sin embargo, se observan confusiones significativas entre clases visualmente similares evidenciando que el modelo aún presenta dificultades para diferenciar objetos con formas, colores y texturas similares.

En el ejemplo de test, el modelo cometió un error notable al clasificar una imagen clara de un ratón como "Monitor" con una confianza del 1.00, evidenciando una

sobreconfianza en la clase incorrecta. Esta situación indica que el modelo podría estar sobreajustado a ciertos patrones visuales o que el dataset no es lo suficientemente representativo para cubrir todas las variabilidades posibles en cada clase (se podría solucionar añadiendo muchos más recursos fotográficos).

En resumen, los resultados obtenidos evidencian que el modelo es capaz de clasificar correctamente algunos objetos de escritorio, pero aún necesita bastantes mejoras en la diferenciación de objetos con características visuales similares.