Práctica 1: Red convolutiva

Explicación del conjunto de datos

Hemos obtenido los datos para este proyecto de diferentes fuentes en Kaggle. El conjunto de imágenes incluye diversas clases de vehículos, como aviones, guaguas, motos, coches, helicópteros, globos y barcos. Cada clase cuenta con aproximadamente 800 imágenes en total.

Para garantizar un equilibrio en el número de imágenes por clase y evitar sesgos en el análisis, desarrollamos un código que ajusta la cantidad de imágenes, eliminando las excedentes hasta que todas las clases tengan el mismo número de ejemplos. Esto asegura un conjunto de datos homogéneo y más adecuado para el entrenamiento y la evaluación del modelo. Además, realizamos una limpieza manual para eliminar fotos donde aparecían múltiples vehículos de otras clases en una misma imagen.

Asimismo, implementamos un segundo código para dividir el conjunto de datos en tres subconjuntos: entrenamiento con un 70%, validación con un 15% y prueba con otro 15%. Esta división se realizó de manera uniforme para cada clase, asegurando que la proporción de datos en cada subconjunto sea consistente y representativa. Esto permite evaluar correctamente el rendimiento del modelo en cada etapa del desarrollo.

Etiqueta	**Clase**
0	Aviones
1	Barco
2	Guagua
3	Coche
4	Globo
5	Helicóptero
6	Moto

Pruebas:

En este proyecto, hemos realizado pruebas y ajustes en diversos hiper parámetros para optimizar el rendimiento del modelo. Los hiper parámetros considerados incluyen los siguientes:

- **Número de épocas:** Determina cuántas veces el modelo recorrerá el conjunto de datos de entrenamiento.
- **Tasa de aprendizaje:** Controla el tamaño de los pasos que el modelo da al actualizar los pesos durante el entrenamiento.
- **Tamaño del mini lote:** Define el número de muestras que se procesan juntas antes de actualizar los pesos.
- **Número de capas convolucionales y completamente conectadas (fully connected):** Influye directamente en la capacidad del modelo para extraer y procesar características relevantes.
- **Número de filtros por capa convolucional:** Regula la cantidad de detectores de características aprendidos en cada capa convolucional.
- **Número de neuronas por capa completamente conectada:** Afecta la capacidad del modelo para combinar las características extraídas y realizar predicciones precisas.
- **Funciones de activación:** Determinan cómo se transforman las salidas de cada capa y son clave para la introducción de no linealidad en el modelo.

Estos hiper parámetros se han ajustado cuidadosamente mediante pruebas sistemáticas para identificar la combinación óptima que maximice el rendimiento del modelo en términos de precisión, velocidad de convergencia y capacidad de generalización.

Para comenzar, es importante destacar que, en cada una de las pruebas realizadas, hemos variado los siguientes parámetros clave mediante **tuning**:

- **Tasa de aprendizaje (learning rate):** Probamos valores que oscilan entre **0.005** y **0.00001**, evaluando cómo afectan la convergencia y el rendimiento del modelo.
- **Número de épocas:** Configuramos el entrenamiento para **15** y **30 épocas**, analizando su impacto en la capacidad del modelo para aprender de los datos.
- **Tamaño de los lotes (batch size):** Probamos tamaños de lotes de **32** y **64**, observando su influencia en la estabilidad del entrenamiento y la velocidad de convergencia.

Se probaron todas las combinaciones posibles, generando gráficas de pérdida y precisión tanto para el entrenamiento como para la validación. De todas estas combinaciones, seleccionamos las **cinco mejores configuraciones** basándonos en los resultados de validación y, finalmente, elegimos la que ofreció la mejor **exactitud en validación** para realizar las pruebas finales. Posteriormente, se aplicó esta configuración al conjunto de datos de prueba (test), generando la **matriz de confusión** y obteniendo su **exactitud**.

Adicionalmente, implementamos técnicas para mejorar la estabilidad y generalización del modelo:

- **Dropout:** Utilizado para prevenir el sobreajuste, desactivando aleatoriamente un porcentaje de las neuronas durante el entrenamiento.
- **MaxPooling:** Incorporado en las capas convolucionales para reducir dimensionalidad y extraer características relevantes de manera más eficiente.

- **Batch Normalization:** Aplicada tanto a las capas convolucionales como a las capas completamente conectadas (fully connected) para normalizar la activación de las neuronas durante el entrenamiento, acelerando la convergencia.

Estas optimizaciones nos permitieron mejorar el rendimiento general del modelo y garantizar su robustez durante el entrenamiento y las pruebas.

Prueba 1:

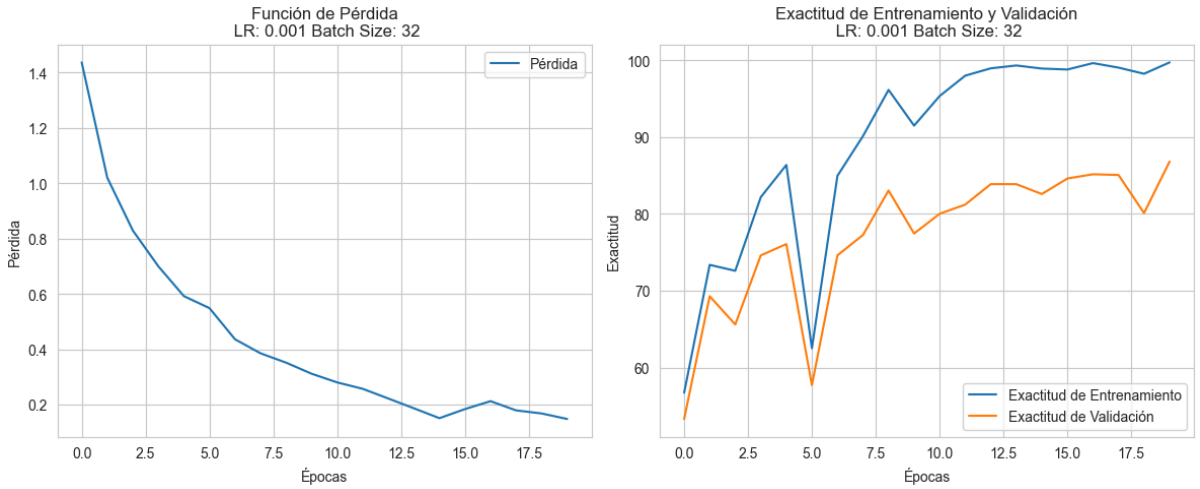
Empleamos una arquitectura compuesta por **tres capas convolucionales** seguidas de **tres capas completamente conectadas (fully connected)** con funciones de activación **ReLU**. Como optimizador, utilizamos **Adam**

```
self.conv1   =   nn.Conv2d(in_channels=3,   out_channels=32,
kernel_size=3, stride=1, padding=1)
self.conv2   =   nn.Conv2d(in_channels=32,   out_channels=64,
kernel_size=3, stride=1, padding=1)
self.conv3   =   nn.Conv2d(in_channels=64,   out_channels=128,
kernel_size=3, stride=1, padding=1)
```

```
self.fc1 = nn.Linear(128 * 12 * 12, 256)
self.fc2 = nn.Linear(256, 128)
self.fc3 = nn.Linear(128, 7)
```

Las cinco mejores configuraciones de parámetros que obtuvimos fueron:

LR=0.001, Batch=32, Epochs=30, Exactitud validación=86.79%
LR=0.001, Batch=64, Epochs=30, Exactitud validación=85.50%
LR=0.001, Batch=32, Epochs=15, Exactitud validación=84.59%
LR=0.0005, Batch=64, Epochs=30, Exactitud validación=84.59%
LR=0.0005, Batch=32, Epochs=30, Exactitud validación=83.76%

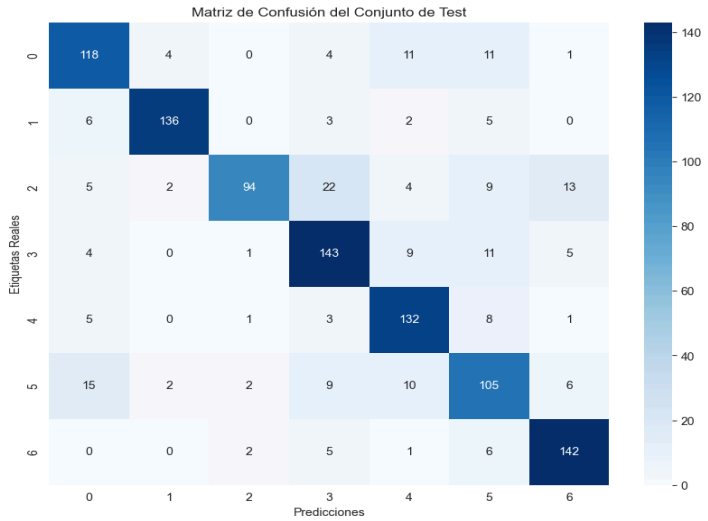


En las gráficas se observan los siguientes aspectos clave:

En la gráfica de la izquierda la pérdida disminuye de manera consistente a medida que avanzan las épocas, indicando que el modelo está aprendiendo y ajustándose a los datos de entrenamiento de forma efectiva. Hacia las últimas épocas, la pérdida se estabiliza, sugiriendo una buena convergencia.

En la gráfica de la derecha la exactitud del entrenamiento aumenta rápidamente en las primeras épocas y luego se estabiliza, superando el 90%. La exactitud de validación también mejora inicialmente, pero muestra una ligera fluctuación en las últimas épocas. Esto podría indicar cierto grado de sobreajuste, ya que la brecha entre las curvas de entrenamiento y validación se amplía ligeramente.

Y testeamos el mejor LR=0.001, Batch=32, Epochs=30 que nos dio una exactitud en test de 80.71%



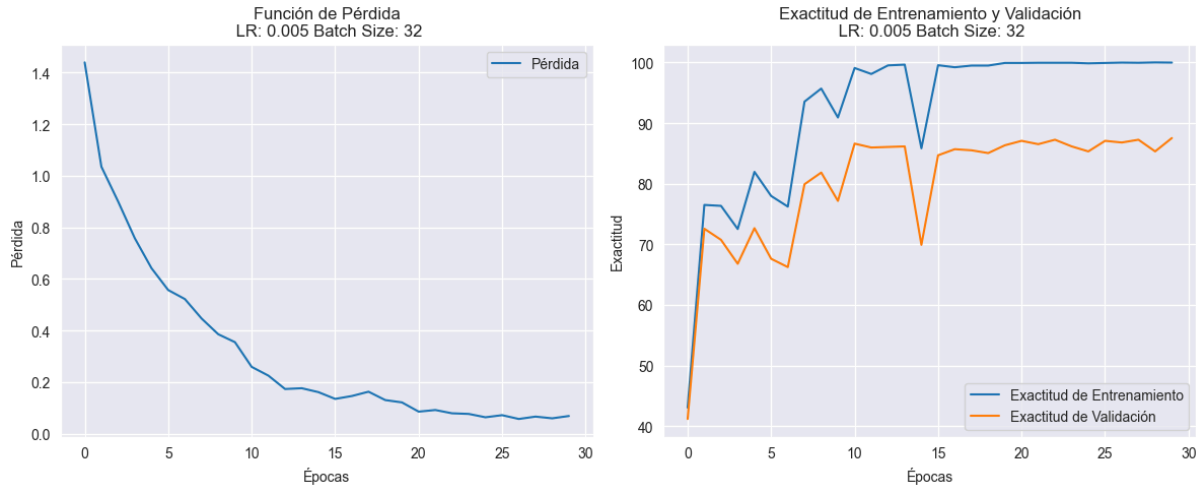
En la matriz de confusión, observamos como la diagonal principal tiene los valores más altos, lo que indica que el modelo clasifica correctamente la mayoría de las instancias para cada clase. Algunas clases tienen confusiones significativas con otras: vemos como la clase Guagua (2) tiene varias instancias clasificadas como Coche (3) y Moto (6), y la clase Helicóptero (5) muestra confusiones con la clase Aviones (0) y, en menor medida, con otras.

Prueba 2:

En esta segunda prueba, mantuvimos la configuración inicial e incorporamos una variación en la tasa de aprendizaje (**learning rate variation**) que reduce el learning rate actual a la mitad cada **10 épocas**. Esta estrategia permitió una mejor convergencia, ajustándose dinámicamente al ritmo de aprendizaje del modelo a medida que avanzaba el entrenamiento.

Las cinco mejores configuraciones de parámetros que obtuvimos fueron:

LR=0.005, Batch=32, Epochs=30, Exactitud validación=87.52%
LR=0.005, Batch=32, Epochs=15, Exactitud validación=86.97%
LR=0.001, Batch=32, Epochs=30, Exactitud validación=86.15%
LR=0.0005, Batch=32, Epochs=15, Exactitud validación=85.50%
LR=0.001, Batch=32, Epochs=15, Exactitud validación=85.41%

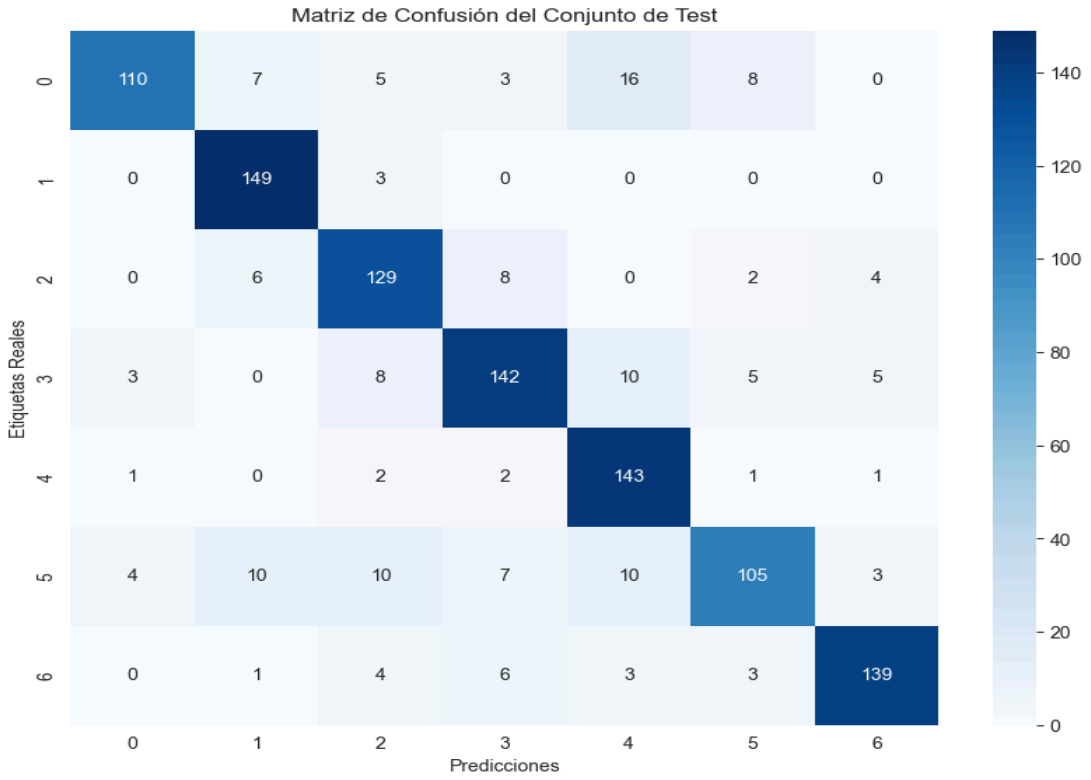


La gráfica de la izquierda muestra la evolución de la función de pérdida durante el entrenamiento del modelo. Se observa una disminución consistente de la pérdida a medida que avanzan las épocas, comenzando en 1.4 y alcanzando valores cercanos a 0.1. Esto indica que el modelo está aprendiendo de los datos y ajustándose progresivamente.

En la gráfica de la derecha se presenta la exactitud tanto en el conjunto de entrenamiento como en el de validación. La exactitud de entrenamiento (línea azul) aumenta rápidamente,

estabilizándose cerca del 95-100 %. Por otro lado, la exactitud de validación (línea naranja) es más baja y fluctúa ligeramente, estabilizándose alrededor del 80-85 %.

Y testeamos el mejor LR=0.005, Batch=32, Epochs=30 que nos dio una exactitud en test de 85.06%



La matriz de confusión del conjunto de test muestra que el modelo clasifica correctamente la mayoría de las muestras, con valores altos en la diagonal principal (e.g., 149 para la clase Barco (1) y 143 para la clase Globo (4)). Sin embargo, se observan errores relevantes, como en la clase Aviones (0), donde 16 muestras se clasificaron incorrectamente como clase Globo (4). Esto indica que, aunque el modelo tiene un buen desempeño general, existe confusión entre ciertas clases que podría optimizarse para mejorar su precisión.

Gracias a la implementación del ajuste automático del learning rate, el modelo logró una mejora significativa en su rendimiento. Este enfoque permitió optimizar el proceso de aprendizaje, lo que resultó en una mayor exactitud tanto en la validación como en el conjunto de test, reduciendo errores y mejorando la capacidad del modelo para generalizar sobre datos no vistos.

Prueba 3:

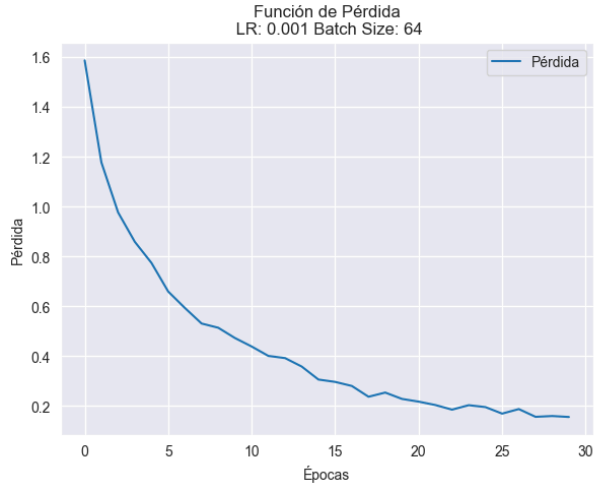
En esta configuración, añadimos una **nueva capa convolucional** a las tres capas convolucionales existentes, modificando sus parámetros de **tamaño del kernel (kernel size)**, **stride** y **padding** para explorar su impacto en la extracción de características.

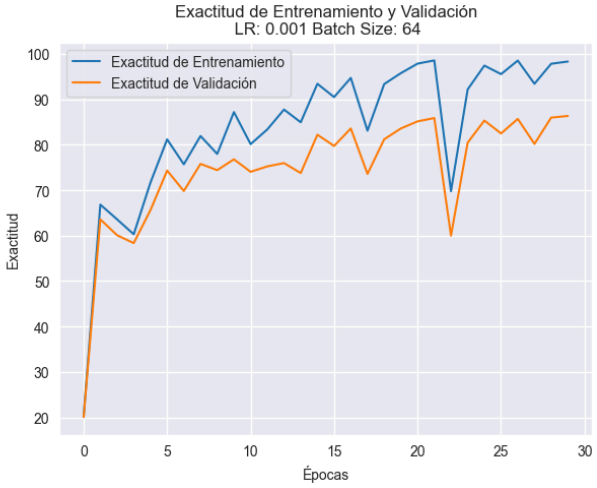
Mantuvimos las **tres capas completamente conectadas (fully connected)** en la arquitectura y utilizamos el optimizador **Adam** para ajustar los parámetros del modelo.

```
self.conv1 = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=5, stride=1, padding=2)
self.conv2 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size=5, stride=1, padding=2)
self.conv3 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=5, stride=1, padding=2)
self.conv4 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, stride=1, padding=1)  # Nueva capa convolucional
```

Las cinco mejores configuraciones de parámetros que obtuvimos fueron:

LR=0.001, Batch=64, Epochs=30, Exactitud validación=86.33%
LR=0.005, Batch=32, Epochs=30, Exactitud validación=86.24%
LR=0.001, Batch=32, Epochs=30, Exactitud validación=85.69%
LR=0.0005, Batch=64, Epochs=30, Exactitud validación=85.69%
LR=0.001, Batch=64, Epochs=15, Exactitud validación=84.50%



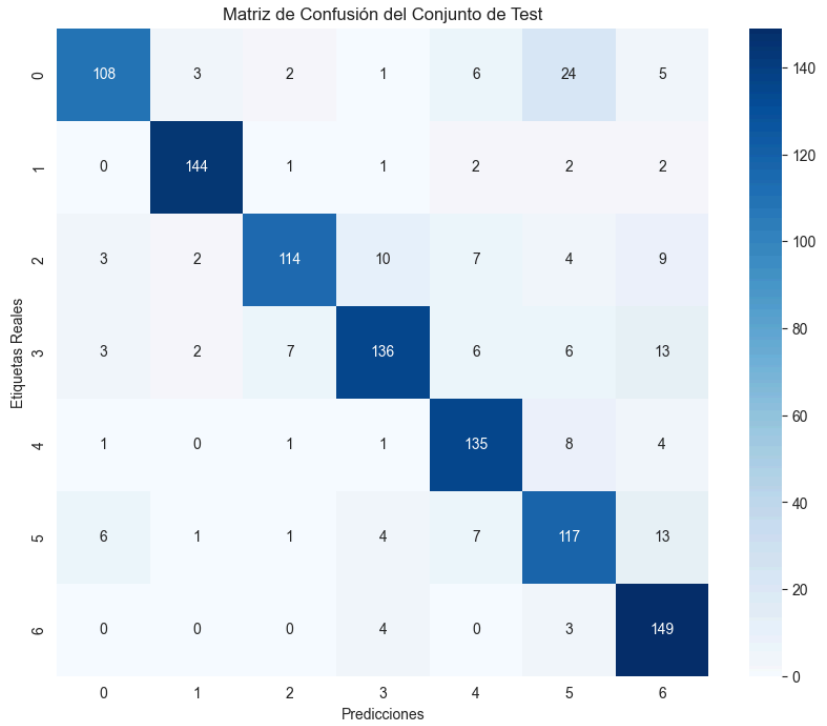


La función de pérdida muestra un descenso consistente a lo largo de las épocas, lo que indica que el modelo está aprendiendo de manera efectiva.

En cuanto a la exactitud de entrenamiento y validación, se observa un buen incremento durante las primeras 10 épocas, estabilizándose posteriormente. La diferencia entre la exactitud de entrenamiento y validación es razonable, lo cual indica que no hay un sobreajuste significativo. Esto es alentador, ya que significa que el modelo generaliza bien para datos que no ha visto durante el entrenamiento.

La mejor configuración de hiper parámetros (LR=0.001, Batch=64, Epochs=30) obtuvo una exactitud de validación del 86.33%, lo que coincide con los resultados observados en la gráfica de exactitud. Las configuraciones con un aprendizaje más rápido (LR=0.005) o un batch más pequeño también tuvieron un buen desempeño, pero la combinación de LR=0.001 y un batch size de 64 parece ofrecer el equilibrio ideal entre estabilidad y rendimiento.

Y testeamos el mejor LR=0.001, Batch=64, Epochs=30 que nos dio una exactitud en test de 83.77%



Con la configuración LR=0.001, Batch=64, Epochs=30, se obtuvo una exactitud en el conjunto de test del 83.77%. Según la matriz de confusión, las clases mejor predichas fueron moto (6), con 149 aciertos, y barco (1), con 144 aciertos. Por otro lado, las clases con mayor confusión fueron aviones (0), confundidos principalmente con globos (4) y helicópteros (5), y coche (3), confundidos con motos (6) y globos (4). En general, el modelo muestra un buen rendimiento, aunque las confusiones suelen darse entre clases con similitudes visuales.

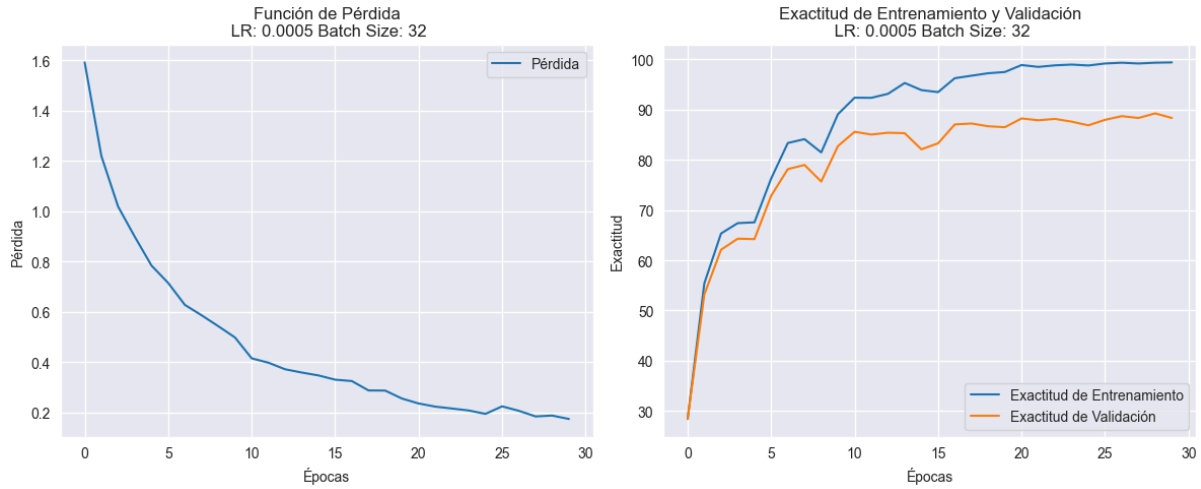
Prueba 4:

En esta configuración añadimos a la anterior configuración la variación del learning rate, reduciendo su valor a la mitad cada 10 épocas. Esta combinación permitió al modelo

aprovechar una mayor capacidad de extracción de características junto con un ajuste dinámico del ritmo de aprendizaje, logrando una convergencia más eficiente y mejorando los resultados en validación y prueba.

Las cinco mejores configuraciones de parámetros que obtuvimos fueron:

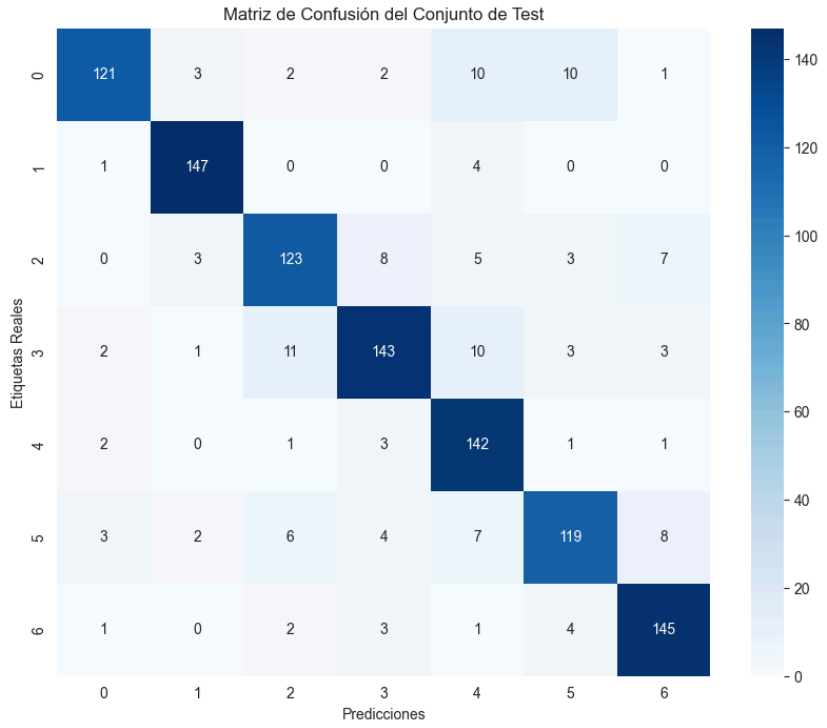
LR=0.0005, Batch=32, Epochs=30, Exactitud validación=88.35%
LR=0.005, Batch=32, Epochs=30, Exactitud validación=87.71%
LR=0.001, Batch=64, Epochs=30, Exactitud validación=87.71%
LR=0.005, Batch=32, Epochs=15, Exactitud validación=87.34%
LR=0.005, Batch=64, Epochs=30, Exactitud validación=87.25%



En la gráfica de la izquierda la pérdida disminuye de forma constante a lo largo de las 30 épocas, lo que indica que el modelo está aprendiendo correctamente. Al final del entrenamiento, la pérdida alcanza un valor bajo, mostrando una buena convergencia.

En la gráfica de la derecha la exactitud de entrenamiento mejora rápidamente durante las primeras 10 épocas y luego se estabiliza, alcanzando un valor cercano al 100%. La exactitud de validación sigue una tendencia similar, aunque se estabiliza en un nivel ligeramente más bajo que la de entrenamiento, alrededor del 90%.

Y testeamos el mejor LR=0.0005, Batch=32, Epochs=30 que nos dió una exactitud de: 87.20%



El desempeño general del modelo es bueno, con la mayoría de las predicciones correctamente clasificadas, como se refleja en los valores altos en la diagonal principal de la matriz de confusión. Esto demuestra que el modelo ha logrado capturar las características principales de las clases en el conjunto de datos. Sin embargo, se pueden identificar algunas áreas problemáticas que necesitan atención.

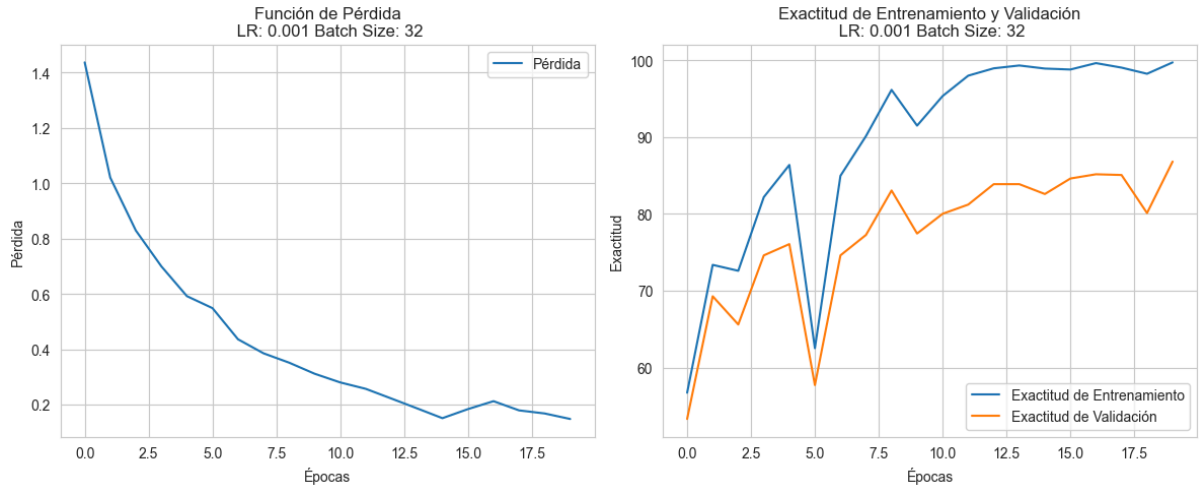
Las clases Aviones (0), Guagua (2) y Helicóptero (5)presentan mayores niveles de confusión. Por ejemplo, la clase Aviones (0) es confundida frecuentemente con la clase Globo (4)(10 casos), mientras que la clase Guagua (2) se confunde con las clases Coche(3)(8 casos) y Moto (6) (7 casos). Por su parte, la clase Helicóptero (5) es confundida con varias otras clases, incluyendo la Moto (6) (8 casos). Estos errores indican que el modelo encuentra dificultades para diferenciar entre estas clases, posiblemente debido a similitudes en sus características o a una representación insuficiente en los datos de entrenamiento.

En contraste, las clases Barco (1), Globo (4) y Moto (6) destacan como las mejor clasificadas, con pocos errores fuera de la diagonal principal. Esto sugiere que el modelo tiene un buen entendimiento de las características que definen estas clases.

EJEMPLO CON PARADA ANTICIPADA:

A continuación, se presenta un ejemplo práctico de la técnica de parada anticipada, utilizada comúnmente en el entrenamiento de modelos de aprendizaje profundo para prevenir el sobreajuste. Esta técnica monitorea el desempeño del modelo en el conjunto de validación durante el proceso de entrenamiento y detiene automáticamente el entrenamiento cuando no se observan mejoras significativas en la exactitud o la pérdida de validación. El objetivo principal es optimizar el rendimiento del modelo, garantizando una buena generalización y evitando ajustes excesivos a los datos de entrenamiento.

Entrenando con lr=0.001, batch_size=32, epochs=30
Época 1, pérdida: 1.4364, exactitud entrenamiento: 56.72%, exactitud validación: **53.30%**
Época 2, pérdida: 1.0201, exactitud entrenamiento: 73.35%, exactitud validación**: 69.27%**
Época 3, pérdida: 0.8290, exactitud entrenamiento: 72.58%, exactitud validación: **65.60%**
Época 4, pérdida: 0.6995, exactitud entrenamiento: 82.16%, exactitud validación: **74.59%**
Época 5, pérdida: 0.5918, exactitud entrenamiento: 86.35%, exactitud validación: **76.06%**
Época 6, pérdida: 0.5484, exactitud entrenamiento: 62.51%, exactitud validación: **57.71%**
Época 7, pérdida: 0.4359, exactitud entrenamiento: 84.94%, exactitud validación: **74.59%**
Época 8, pérdida: 0.3851, exactitud entrenamiento: 90.11%, exactitud validación: **77.25%**
Época 9, pérdida: 0.3514, exactitud entrenamiento: 96.12%, exactitud validación: **83.03%**
Época 10, pérdida: 0.3113, exactitud entrenamiento: 91.47%, exactitud validación: **77.43%**
Época 11, pérdida: 0.2800, exactitud entrenamiento: 95.32%, exactitud validación: **80.00%**
Época 12, pérdida: 0.2568, exactitud entrenamiento: 97.97%, exactitud validación: **81.19%**
Época 13, pérdida: 0.2218, exactitud entrenamiento: 98.92%, exactitud validación: **83.85%**
Época 14, pérdida: 0.1865, exactitud entrenamiento: 99.29%, exactitud validación: **83.85%**
Época 15, pérdida: 0.1507, exactitud entrenamiento: 98.89%, exactitud validación: **82.57%**
Época 16, pérdida: 0.1836, exactitud entrenamiento: 98.77%, exactitud validación: **84.59%**
Época 17, pérdida: 0.2122, exactitud entrenamiento: 99.60%, exactitud validación: **85.14%**
Época 18, pérdida: 0.1789, exactitud entrenamiento: 99.01%, exactitud validación: **85.05%**
Época 19, pérdida: 0.1678, exactitud entrenamiento: 98.21%, exactitud validación: **80.09%**
Parada anticipada activada



En la gráfica de la función de pérdida, se observa un descenso constante hasta estabilizarse alrededor de la época 19. Esto indica que el modelo deja de obtener mejoras significativas en el conjunto de validación, señalando el momento adecuado para detener el entrenamiento.

En la gráfica de la función de pérdida, se puede observar que esta desciende de manera constante hasta estabilizarse, lo cual es consistente con el aprendizaje del modelo.

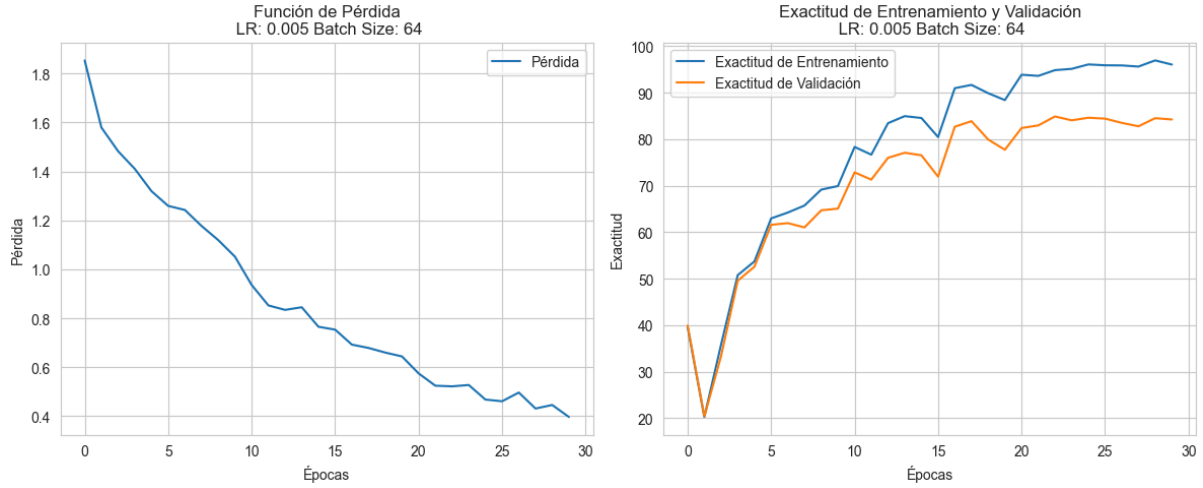
En la gráfica de exactitud, las curvas muestran una divergencia a partir de la época 14, lo que refleja el inicio del sobreajuste. Este fenómeno ocurre cuando el modelo comienza a ajustar demasiado a los datos de entrenamiento, perdiendo capacidad de generalización en datos nuevos. La parada anticipada ayuda a mitigar este efecto, optimizando el balance entre el ajuste al conjunto de entrenamiento y la generalización al conjunto de validación.

Prueba 5:

Para la siguiente prueba, ampliamos la configuración anterior añadiendo **cinco capas completamente conectadas (fully connected)**. Esta modificación buscó mejorar la capacidad del modelo para combinar las características extraídas por las capas convolucionales, permitiéndole aprender representaciones más complejas y precisas para la tarea objetivo.

Las cinco mejores configuraciones de parámetros que obtuvimos fueron:

LR=0.005, Batch=64, Epochs=30, Exactitud validación=84.22%
LR=0.001, Batch=32, Epochs=30, Exactitud validación=83.49%
LR=0.0005, Batch=64, Epochs=30, Exactitud validación=83.12%
LR=0.005, Batch=32, Epochs=30, Exactitud validación=82.39%
LR=0.001, Batch=64, Epochs=30, Exactitud validación=81.19%

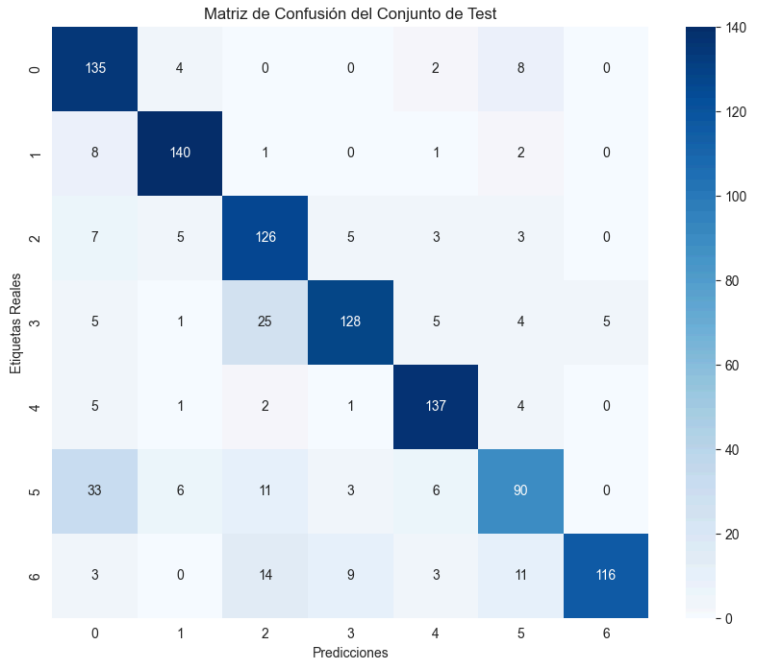


Las gráficas muestran el comportamiento de la pérdida y la exactitud durante el entrenamiento y la validación con la configuración LR=0.005, Batch=64, Epochs=30.

En la primera gráfica, la función de pérdida disminuye de manera constante a lo largo de las épocas, lo que indica que el modelo está aprendiendo progresivamente. Este descenso es significativo durante las primeras 10 épocas y luego se estabiliza, reflejando un aprendizaje más refinado.

En la segunda gráfica, se observa un incremento consistente en la exactitud tanto de entrenamiento como de validación. La exactitud de validación sigue de cerca a la de entrenamiento, lo que sugiere que el modelo generaliza bien al conjunto de validación. Aunque hay fluctuaciones menores, ambas métricas muestran una tendencia estable después de la época 20, alcanzando aproximadamente el 84% de exactitud de validación.

Y testeamos el mejor LR=0.005, Batch=64, Epochs=30 que nos dió una exactitud de: 80.89 %



Según la matriz de confusión, las clases mejor predichas fueron aviones (0), con 135 predicciones correctas y pocas confusiones con helicópteros (5) y motos (6), y barco (1), con 140 aciertos y errores mínimos. Moto (6) también destacó, con 116 clasificaciones correctas, aunque hubo confusiones con coches (9) y helicópteros (11).

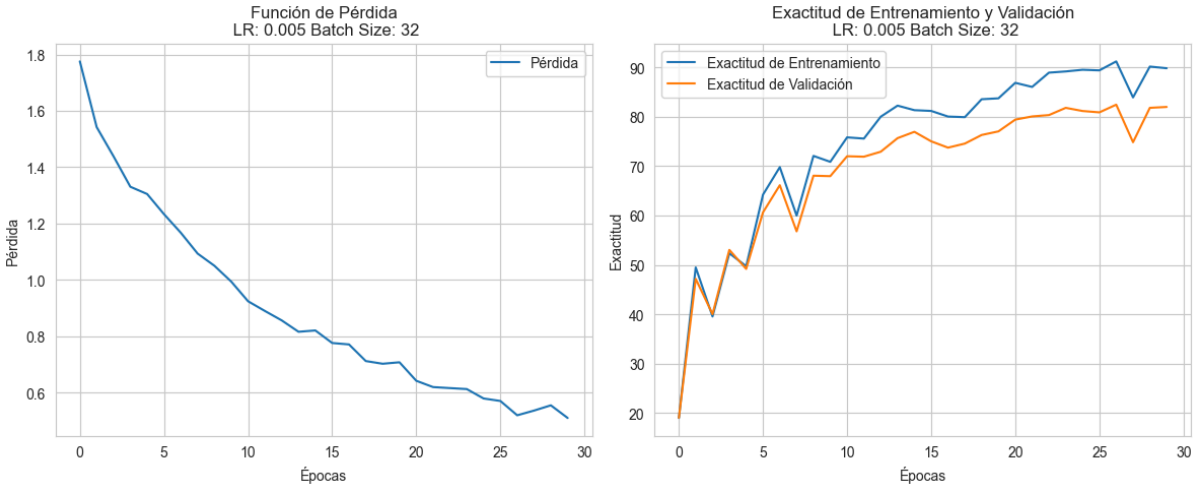
Por otro lado, las clases con mayores confusiones fueron helicóptero (5), que tuvo 90 aciertos pero fue confundida con frecuencia con aviones (33) y guaguas (11), y coche (3), que mostró confusiones notables con guaguas (25) y motos (5). Estos resultados reflejan que, aunque el modelo tiene un rendimiento general sólido, las confusiones tienden a concentrarse en clases con características visuales similares, como aviones y helicópteros, o coches y guaguas.

Prueba 6:

En esta prueba decidimos sustituir la función de activación ReLU por la función sigmoide para evaluar si el cambio en las funciones no lineales podría ofrecer un mejor rendimiento.

Las cinco mejores configuraciones de parámetros que obtuvimos fueron:

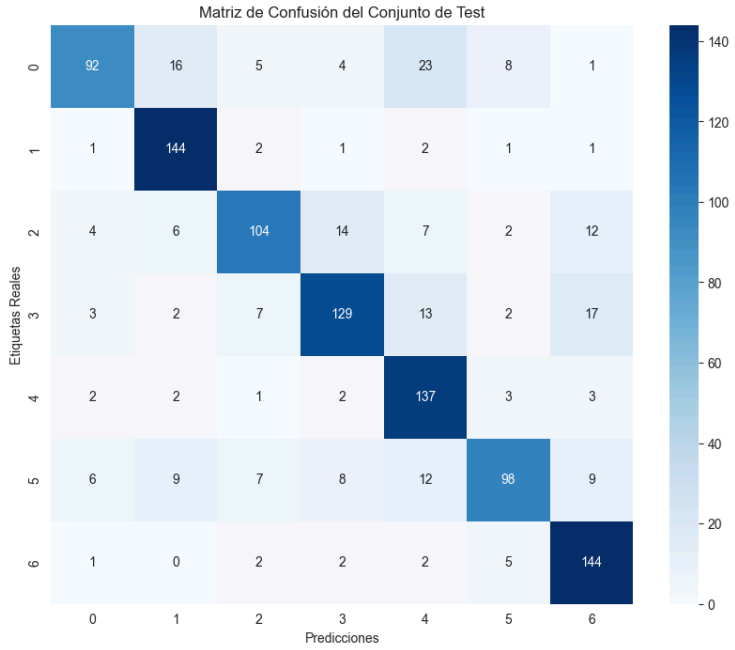
LR=0.005, Batch=32, Epochs=30, Exactitud validación=82.02%
LR=0.005, Batch=64, Epochs=30, Exactitud validación=79.27%
LR=0.005, Batch=32, Epochs=15, Exactitud validación=74.50%
LR=0.001, Batch=64, Epochs=30, Exactitud validación=72.75%
LR=0.0005, Batch=32, Epochs=30, Exactitud validación=69.82%



En el gráfico de la izquierda la pérdida disminuye consistentemente a lo largo de las 30 épocas, lo que indica que el modelo está aprendiendo de manera efectiva. No se observa una estabilización completa de la pérdida al final del entrenamiento, lo que sugiere que el modelo podría beneficiarse de más épocas para alcanzar una convergencia completa.

En el gráfico de la derecha la exactitud de entrenamiento mejora rápidamente en las primeras épocas, alcanzando valores cercanos al 90%. La exactitud de validación también mejora, pero se estabiliza alrededor del 80%, lo que indica una brecha moderada entre entrenamiento y validación.

Exactitud del modelo en el conjunto de test para el conjunto de hiperparametros LR=0.005, Batch=32, Epochs=30 : 78.66%



En general, el modelo tiene un buen rendimiento, con la mayoría de las predicciones correctas alineadas en la diagonal principal. Sin embargo, se observan áreas específicas con confusiones notables que pueden mejorarse.

El modelo clasifica con alta precisión las clases Barco (1), Globo (4) y Moto (6), con 144 predicciones correctas para cada una. Estas categorías presentan pocas confusiones con otras clases, lo que demuestra que el modelo identifica de manera efectiva sus características distintivas. Este rendimiento indica que las representaciones aprendidas por el modelo para estas clases son sólidas y discriminativas.

Por otro lado, algunas clases presentan mayores niveles de confusión. La clase Aviones (0) se confunde frecuentemente con la clase Globo (4), con 23 casos mal clasificados, lo que podría deberse a similitudes visuales entre estas categorías. La clase Guagua (2) muestra problemas al ser confundida con Coche (3) en 14 casos y con Moto (6) en 12 casos, lo que sugiere que estas clases comparten patrones o características comunes que dificultan su distinción. Además, la clase Helicóptero (5) tiene 12 clasificaciones incorrectas hacia la clase Globo (4), lo que indica que las características de los helicópteros no son suficientemente discriminativas frente a los globos.

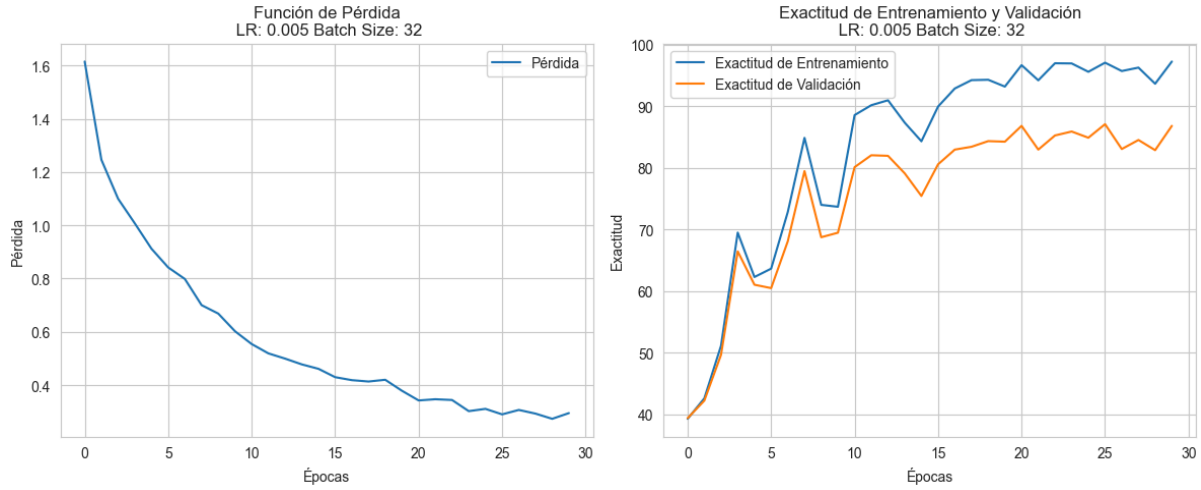
Prueba 7:

En esta prueba, partimos de la configuración utilizada en la **Prueba 4** y realizamos un cambio en el optimizador, reemplazando **Adam** por el optimizador **estocástico (SGD - Stochastic Gradient Descent)**. Este ajuste nos permitió analizar cómo un enfoque más

sencillo y tradicional para la optimización afecta el rendimiento del modelo en términos de convergencia y precisión.

Las cinco mejores configuraciones de parámetros que obtuvimos fueron:

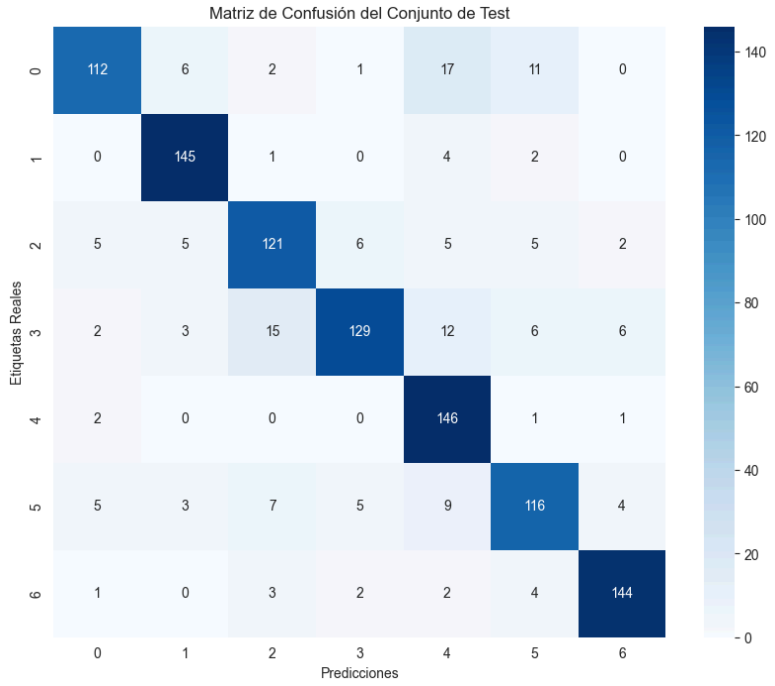
LR=0.005, Batch=32, Epochs=30, Exactitud validación=86.79%
LR=0.005, Batch=32, Epochs=15, Exactitud validación=85.96%
LR=0.005, Batch=64, Epochs=15, Exactitud validación=78.44%
LR=0.001, Batch=32, Epochs=15, Exactitud validación=73.12%
LR=0.001, Batch=32, Epochs=30, Exactitud validación=72.75%



La mejor configuración fue LR=0.005, Batch=32, Epochs=30, alcanzando una exactitud de validación del 86.79%, lo cual sería considerado un buen resultado si el conjunto de testing logra al menos un 85% de exactitud. Otras configuraciones destacadas incluyeron LR=0.005, Batch=32, Epochs=15, con una exactitud de validación del 85.96%, lo que también cumple con el umbral definido.

En las gráficas, la función de pérdida decrece de manera constante, mostrando una buena convergencia con el optimizador SGD. La exactitud de entrenamiento y validación aumenta rápidamente en las primeras épocas y se estabiliza después de aproximadamente 15 épocas. Esto refleja que el modelo es capaz de aprender de manera efectiva con esta configuración, manteniendo una brecha razonable entre entrenamiento y validación.

Exactitud del modelo en el conjunto de test para el conjunto de hiperparametros LR=0.005, Batch=32, Epochs=30 : 84.69%



Aunque no superó el umbral del 85% para considerarse un resultado óptimo, el desempeño es razonablemente bueno en general.

Las clases mejor predichas fueron barco (1), con 145 aciertos y muy pocas confusiones, y globo (4), con 146 clasificaciones correctas, siendo la clase con mayor precisión. Por otro lado, aviones (0) logró 112 aciertos, pero presentó confusiones notables con globos (17) y helicópteros (11).

Otra clase destacada fue coche (3), con 129 clasificaciones correctas, aunque confundida con guaguas (15) y motos (6). Finalmente, helicóptero (5) tuvo 116 aciertos, pero presentó confusiones dispersas con motos (9) y aviones (5).

Aunque esta configuración no logró alcanzar el umbral definido, los resultados son consistentes y reflejan un buen comportamiento en la mayoría de las clases, con algunas áreas que presentan confusiones entre categorías similares visualmente.

Prueba 8:

En esta nueva configuración se ha incrementado la capacidad de la red neuronal convolucional mediante la inclusión de más capas completamente conectadas y el aumento del número de parámetros. Este ajuste persigue optimizar la extracción de características y, con ello, mejorar la capacidad de generalización del modelo.

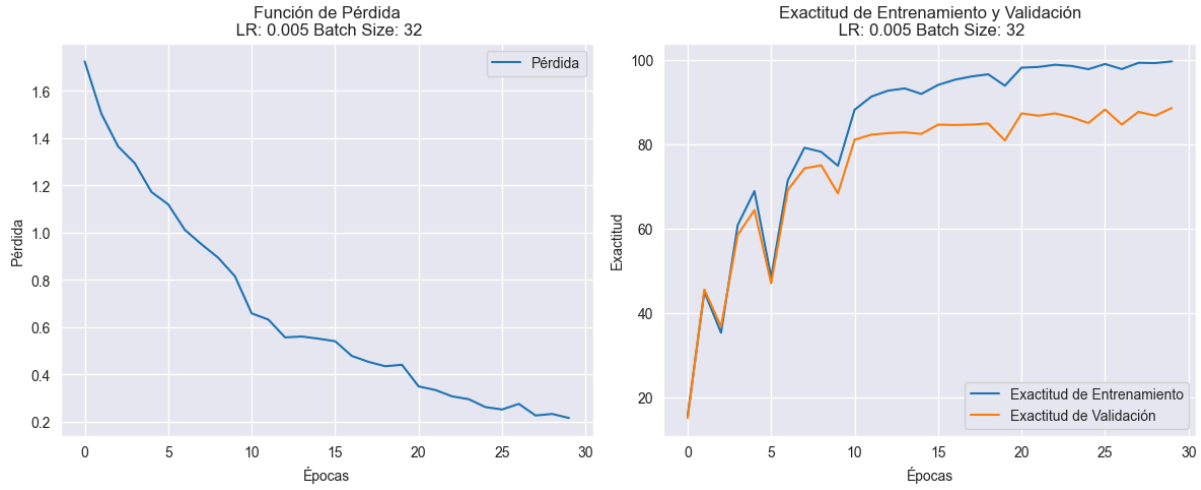
La arquitectura previa se caracterizaba por una primera capa completamente conectada que recibía una entrada de 128×6×6 y generaba 256 salidas, seguida de otra capa que transformaba esas 256 salidas en 128, y concluía con una última que iba de 128 a 7. En la propuesta actual, se incrementa la complejidad de la red al introducir una primera capa de

128×6×6 a 512, continuando con una capa de 512 a 256, otra de 256 a 128, a la que se suma una etapa adicional de 128 a 64, y culminando finalmente en la salida de 64 a 7.

Con esta estrategia, se persigue un potencial incremento en la precisión del modelo, al permitirle capturar relaciones más sutiles y matices presentes en las muestras de entrenamiento.

Las cinco mejores configuraciones de parámetros que obtuvimos fueron:

LR=0.005, Batch=32, Epochs=30, Exactitud validación=88.62%
LR=0.001, Batch=64, Epochs=30, Exactitud validación=87.98%
LR=0.001, Batch=32, Epochs=30, Exactitud validación=87.25%
LR=0.0005, Batch=32, Epochs=30, Exactitud validación=86.61%
LR=0.005, Batch=32, Epochs=15, Exactitud validación=83.85%

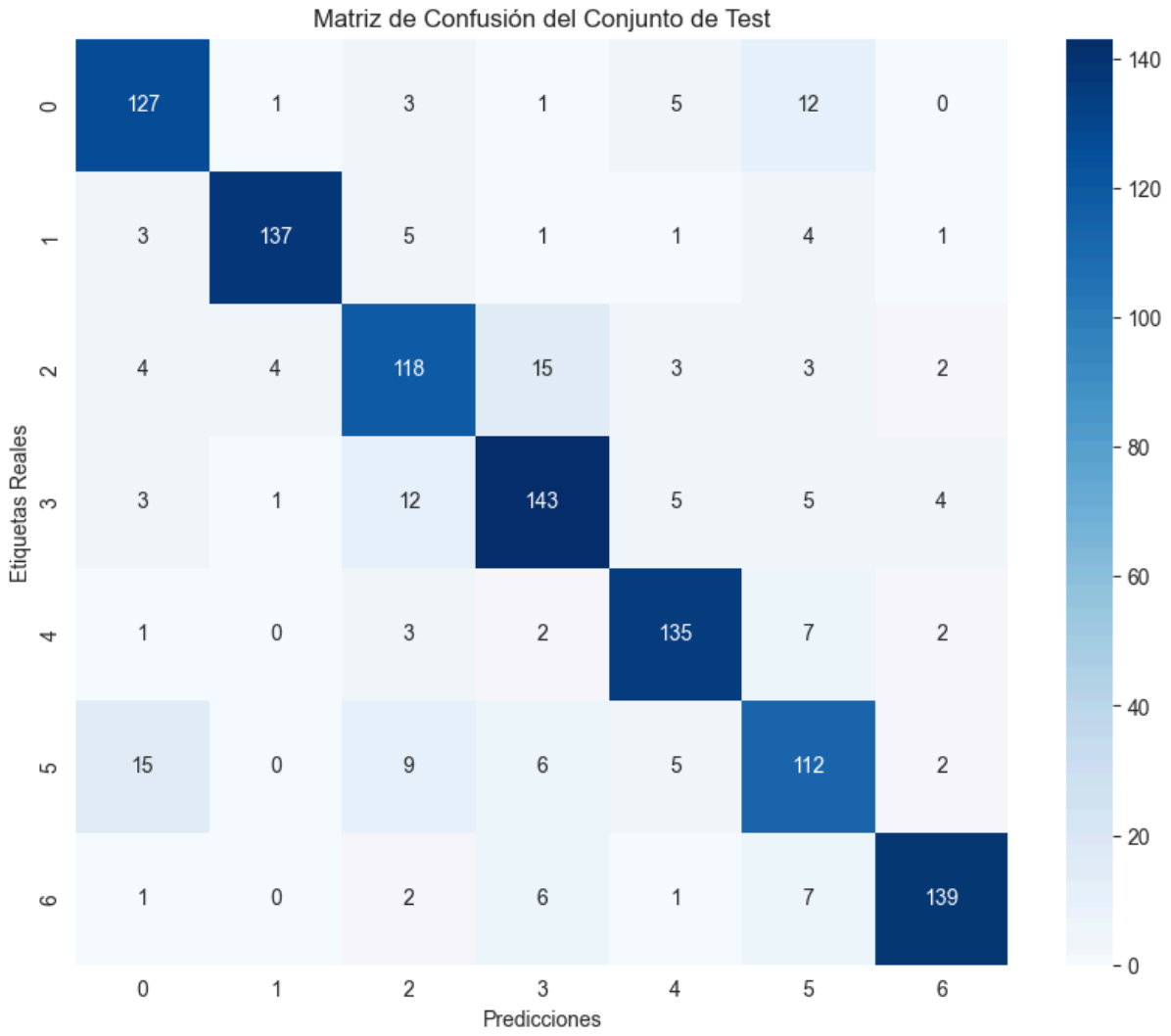


En la gráfica de pérdida, se observa una disminución continua y significativa a lo largo de las 30 épocas, lo que indica un aprendizaje eficiente por parte del modelo. Aunque la pérdida no se estabiliza completamente, su ritmo de descenso disminuye después de aproximadamente 20 épocas, sugiriendo que el modelo se acerca a la convergencia.

En la gráfica de exactitud, el desempeño del modelo en el entrenamiento mejora rápidamente en las primeras épocas y se estabiliza en valores cercanos al 100%. Sin embargo, la exactitud en el conjunto de validación se estabiliza alrededor del 80%, creando una brecha constante entre entrenamiento y validación. Esto indica un leve sobreajuste, ya que el modelo se adapta demasiado a los datos de entrenamiento, comprometiendo ligeramente su capacidad de generalización.

El modelo alcanza una exactitud en el conjunto de prueba del **84.51%**, lo cual está cerca del umbral considerado como un buen desempeño.

Exactitud del modelo en el conjunto de test para el conjunto de hiperparametros LR=0.005, Batch=32, Epochs=30 : 84.51%



Las clases **Aviones (0)**, **Barco (1)**, **Coche (3)**, **Globo (4)** y **Moto (6)** tienen un desempeño destacado, con un alto número de predicciones correctas (127, 137, 143, 135 y 139, respectivamente). Esto indica que el modelo logra identificar de manera efectiva las características distintivas de estas clases. Sin embargo, hay algunas confusiones menores, como **Aviones (0)** siendo confundido con **Helicóptero (5)** en 12 casos, y **Globo (4)** confundido con **Helicóptero (5)** en 7 casos.

Por otro lado, las clases **Guagua (2)** y **Helicóptero (5)** presentan mayores niveles de confusión. La clase **Guagua (2)** es confundida con **Coche (3)** en 15 ocasiones. La clase **Helicóptero (5)** tiene una notable confusión con **Aviones (0)** en 15 ocasiones y **Guagua (2)** en 9 ocasiones.

En términos generales, el modelo demuestra un buen nivel de precisión, pero las confusiones en las clases **Guagua (2)** y **Helicóptero (5)** sugieren la necesidad de mejorar la representación de estas categorías en los datos de entrenamiento.

Prueba 9:

En la siguiente prueba, se ha implementado una estrategia de aumento de datos (data augmentation) para enriquecer el conjunto de entrenamiento y mejorar la capacidad de generalización del modelo. Se aplicaron dos transformaciones diferentes a cada imagen, diseñadas para simular variaciones comunes en los datos reales y reducir el sobreajuste.

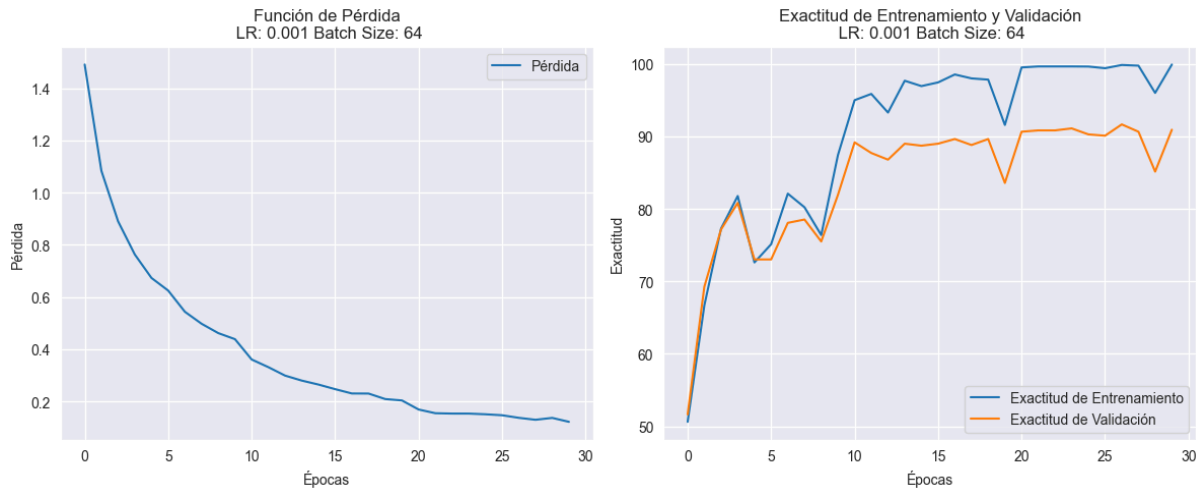
La primera transformación, aplica un volteo horizontal aleatorio con una probabilidad de 50%. Este tipo de transformación es útil para capturar simetrías en las imágenes y expandir la diversidad de orientaciones en los datos.

La segunda transformación, combina varias operaciones para introducir una mayor variabilidad. Incluye una rotación aleatoria de hasta 35 grados, que simula diferentes perspectivas o ángulos de captura. Además, se aplica un ajuste aleatorio de brillo, contraste y saturación, con un rango de 0.3 para cada parámetro, lo que permite simular cambios en las condiciones de iluminación y el balance de colores.

Con estas transformaciones, el modelo puede aprender características más robustas y generalizar mejor en datos no vistos, ya que estará expuesto a un conjunto de entrenamiento más diverso y representativo. Este enfoque es particularmente útil cuando se dispone de un conjunto de datos limitado y se desea maximizar su eficacia.

Las cinco mejores configuraciones de parámetros que obtuvimos fueron:

LR=0.001, Batch=64, Epochs=30, Exactitud validación=90.92%
LR=0.001, Batch=32, Epochs=30, Exactitud validación=90.73%
LR=0.0005, Batch=32, Epochs=30, Exactitud validación=90.00%
LR=0.005, Batch=32, Epochs=30, Exactitud validación=88.90%
LR=0.005, Batch=64, Epochs=30, Exactitud validación=88.44%

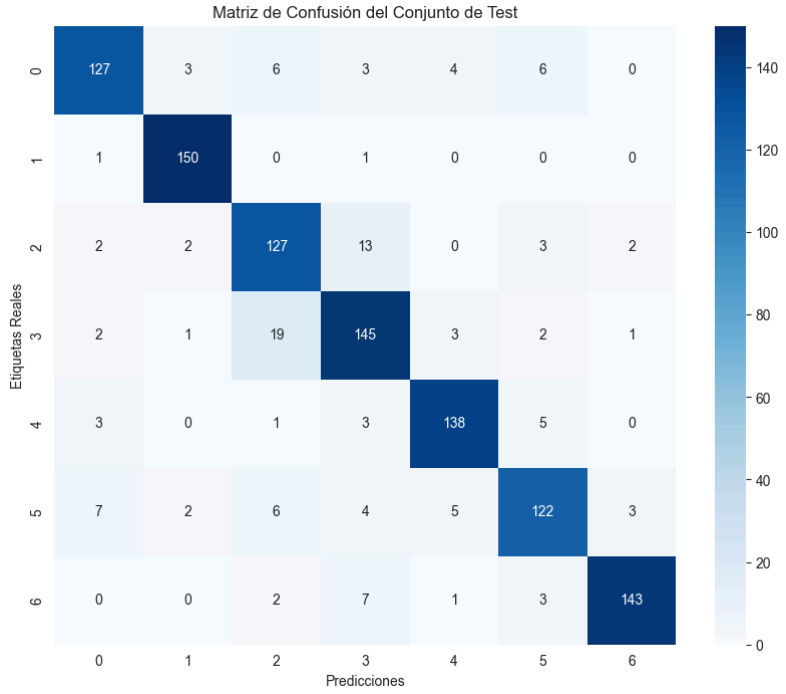


En la gráfica de pérdida, se observa un descenso constante y progresivo hasta estabilizarse cerca de las últimas épocas, lo que indica una buena convergencia del modelo. Esto sugiere que los parámetros elegidos permiten un aprendizaje eficiente sin problemas evidentes de divergencia.

En la gráfica de exactitud, se aprecia un incremento rápido en las primeras épocas para tanto entrenamiento como validación, alcanzando niveles altos alrededor de la época 10. La exactitud de validación se mantiene cercana a la de entrenamiento, lo que indica un bajo nivel de sobreajuste y buena generalización. La elección de un tamaño de lote de 64 parece haber contribuido a este equilibrio, facilitando un aprendizaje estable y eficiente.

Comparando con otras configuraciones, esta destaca por lograr la mayor exactitud de validación, lo que confirma la idoneidad de los parámetros seleccionados para este caso. Sin embargo, otras configuraciones como **LR=0.001, Batch=32, Epochs=30** (90.73%) ofrecen un rendimiento similar, lo que sugiere que la variación del tamaño del lote tiene un impacto moderado en este escenario. Las configuraciones con tasas de aprendizaje más altas (**LR=0.005**) alcanzaron exactitudes menores, lo que indica que estas podrían ser demasiado agresivas para este conjunto de datos y modelo.

Exactitud del modelo en el conjunto de test para el conjunto de hiperparametros LR=0.001, Batch=64, Epochs=30: 88.31%



La matriz de confusión muestra el desempeño del modelo en el conjunto de prueba con una exactitud global del 88.31%, utilizando los hiper parámetros LR=0.001, Batch=64 y Epochs=30. En general, el modelo logra clasificaciones precisas en la mayoría de las categorías, como lo indican los valores altos en la diagonal principal.

La clase Barco (1) es la mejor clasificada, con 150 predicciones correctas y muy pocos errores hacia otras clases, lo que refleja una clara distinción entre esta categoría y las demás. Asimismo, las clases Coche (3) y Moto (6) también presentan un buen desempeño, con 145 y 143 predicciones correctas, respectivamente, y un bajo número de confusiones.

Por otro lado, las clases Helicóptero (5) y Aviones (0) muestran mayores niveles de confusión. Por ejemplo, la clase Helicóptero (5) tiene 7 instancias mal clasificadas como Aviones (0) y 6 como Guagua (2), lo que podría deberse a similitudes en ciertas características visuales entre estas categorías. Por otro lado, la clase coche (3) tiene el mayor número de confusiones con la clase guagua (2). La clase Aviones (0) también presenta errores notables, con 6 instancias clasificadas incorrectamente como Helicóptero (5) y 4 como Globo (4). Estas confusiones indican la necesidad de un mayor refinamiento en las características utilizadas por el modelo para diferenciar estas clases.

Es importante destacar el impacto positivo del *data augmentation* en los resultados. Las transformaciones aplicadas, como el volteo horizontal, la rotación aleatoria y los ajustes de brillo, contraste y saturación, han contribuido a enriquecer el conjunto de datos, permitiendo al modelo aprender características más robustas. Esto se refleja en el hecho de que esta configuración de hiper parámetros ha alcanzado la mejor exactitud global en comparación con las configuraciones anteriores. El *data augmentation* ha sido clave para mejorar la capacidad de generalización del modelo.

Conclusiones finales

Las conclusiones finales del proyecto enfatizan varios aspectos importantes derivados del análisis y las pruebas realizadas:

1. **Eficiencia del modelo:** La implementación de técnicas avanzadas como data augmentation, dropout, batch normalization y max pooling resultaron clave para mejorar el rendimiento general del modelo y reducir el sobreajuste. Entre las diferentes pruebas realizadas, la configuración que incluyó data augmentation fue la que alcanzó el mejor desempeño, logrando una exactitud máxima en el conjunto de prueba del **88.31%** (LR=0.001, Batch=64, Epochs=30). Esto destaca el impacto positivo de estas técnicas en la capacidad del modelo para generalizar y manejar datos no vistos.
2. **Mejores configuraciones**: Las pruebas sistemáticas identificaron combinaciones específicas de hiper parámetros que maximizan el rendimiento. Las configuraciones con tasas de aprendizaje moderadas y tamaños de lotes medianos (e.g., LR=0.001, Batch=64) proporcionaron un equilibrio ideal entre estabilidad y precisión.
3. **Impacto del data augmentation**: La introducción de transformaciones como rotaciones aleatorias, volteo horizontal y ajustes de brillo, contraste y saturación enriqueció el conjunto de datos, mejorando significativamente la capacidad de generalización del modelo.
4. **Análisis de confusiones**: Aunque el modelo mostró un buen desempeño general, se observaron confusiones entre clases con características visuales similares, como

entre aviones y helicópteros, y entre coches y guaguas. Esto sugiere que una mayor diferenciación de estas características podría mejorar aún más la precisión.

5. **Optimización dinámica**: Estrategias como la reducción del learning rate durante el entrenamiento ayudaron a mejorar la convergencia y a ajustar el modelo de manera más eficiente a los datos.
6. **Pruebas con diferentes arquitecturas**: Las pruebas con arquitecturas más profundas y optimizadores alternativos, como SGD, demostraron ser útiles para entender las capacidades y limitaciones del modelo. Aunque algunas configuraciones no superaron el rendimiento base, proporcionaron valiosa información sobre la flexibilidad y robustez del modelo.