Redes Neuronales Convolucionales (CNN)

Clasificación de Pokémon

Jessica Jazmin Castillo Rios

Tópicos Selectos de Grandes Bases de Datos II

Índice general

1 Introducción

Las redes neuronales convolucionales (Convolutional Neural Networks, CNN) representan hoy en día una de las arquitecturas fundamentales en el campo de la visión por computadora, gracias a su capacidad para aprender representaciones jerárquicas a partir de imágenes. Mediante operaciones de convolución, las CNN extraen características locales como bordes, texturas y formas, para posteriormente combinarlas en niveles más complejos que permiten tomar decisiones de clasificación con gran precisión.

En este proyecto se desarrolla un sistema basado en CNN para clasificar imágenes correspondientes a los primeros 151 Pokémon. El dataset utilizado proviene de una colección pública de Kaggle y presenta una alta variabilidad en pose, color, iluminación y estilo visual, lo cual constituye un escenario ideal para evaluar el desempeño de diferentes configuraciones de un modelo convolucional.

El objetivo principal de este trabajo es analizar cómo cambios en la arquitectura, en la configuración de entrenamiento y en las técnicas de preprocesamiento afectan el rendimiento final del modelo. Para ello se diseñaron tres experimentos incrementales:

- **Exp0**: Modelo base con arquitectura TinyVGG.
- **Exp1**: Data augmentation aplicado únicamente al conjunto de entrenamiento.
- **Exp2**: Optimización mejorada mediante AdamW y regularización L2.

Cada experimento fue ejecutado durante 20 épocas y se registraron métricas de pérdida y precisión tanto para entrenamiento como para validación. Los resultados se guardaron en formato CSV y posteriormente se graficaron para su análisis comparativo.

2 Arquitectura del Modelo

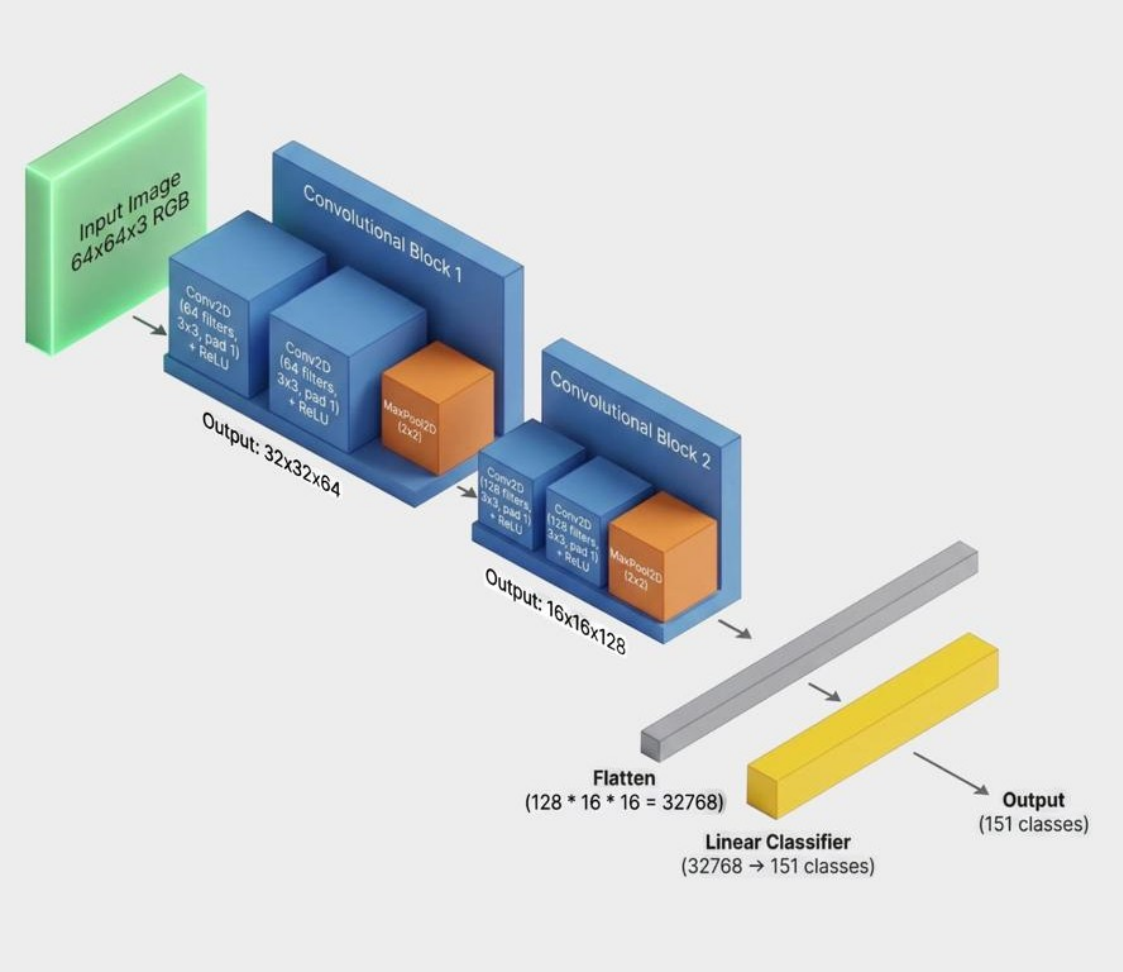
La arquitectura seleccionada como punto de partida es **TinyVGG**, una versión simplificada del modelo VGG tradicional que conserva su estructura de bloques convolucionales pero reduce significativamente su tamaño y costo computacional. TinyVGG es ampliamente utilizado en cursos y prácticas de aprendizaje profundo debido a su simplicidad, estabilidad y facilidad de entrenamiento incluso en CPU.

2.1. Estructura de TinyVGG

El modelo se compone de dos bloques convolucionales principales, seguidos por una capa completamente conectada que realiza la clasificación final. A continuación se describe su arquitectura:

- **Entrada:** Imagen RGB de 64×64 píxeles.
- **Bloque Convolucional 1:**
  - Conv2D (64 filtros, kernel 3×3, padding 1)
  - ReLU
  - Conv2D (64 filtros, kernel 3×3, padding 1)
  - ReLU
  - MaxPool2D (kernel 2×2)
- **Bloque Convolucional 2:**
  - Conv2D (128 filtros, kernel 3×3, padding 1)
  - ReLU
  - Conv2D (128 filtros, kernel 3×3, padding 1)
  - ReLU
  - MaxPool2D (kernel 2×2)
- **Clasificador:**
  - Flatten
  - Linear ($128 \times 16 \times 16 \rightarrow 151$ clases)

2.2. Diagrama de la arquitectura



3 Metodología

Para evaluar el impacto de distintas configuraciones en el rendimiento del modelo, se definieron tres experimentos consecutivos:

3.1. Experimento 0: Modelo base (TinyVGG)

- Arquitectura TinyVGG sin modificaciones.
- Optimizador Adam.
- Learning rate = 0.001.
- Transformación básica: resize + normalización.

3.2. Experimento 1: Data Augmentation

Se incorporaron técnicas de aumento de datos en el conjunto de entrenamiento para incrementar la variabilidad de las muestras y mejorar la capacidad de generalización:

- RandomHorizontalFlip
- RandomRotation
- ColorJitter (brillo, contraste, saturación, tono)
- Normalización estándar

El conjunto de prueba permaneció sin transformaciones aleatorias.

3.3. Experimento 2: Optimización mejorada

Con el fin de reducir overfitting y mejorar la estabilidad:

- Optimizador **AdamW**.
- Regularización L2 (weight decay = 1e-4).
- Data augmentation moderado.

4 Resultados

En esta sección se presentan las curvas de pérdida y precisión por época para cada uno de los experimentos. Todas las métricas provienen de los archivos CSV generados automáticamente durante el entrenamiento.

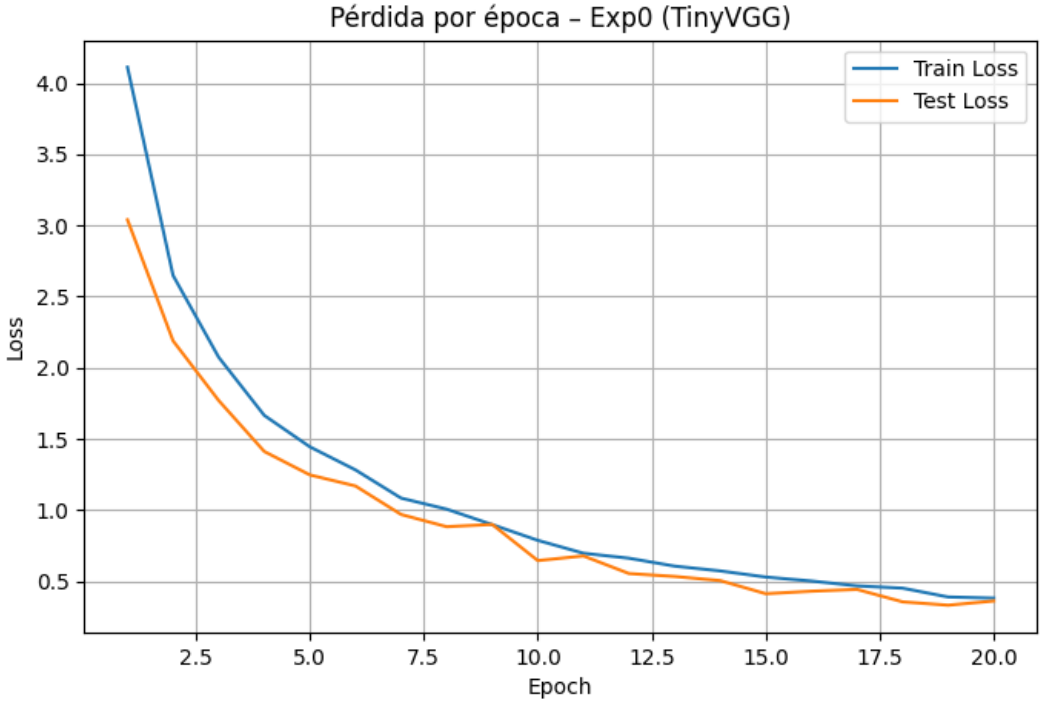
4.1. Experimento 0

Tabla de resultados

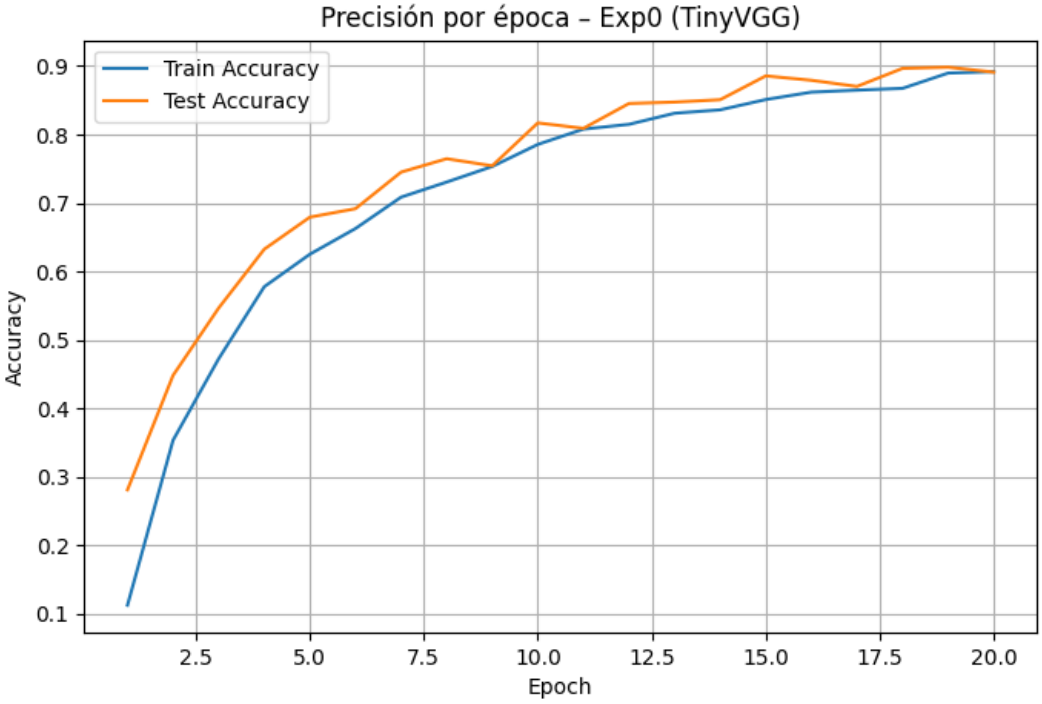
A continuación se presentan los valores obtenidos por época para el experimento:

Epoch	Train Loss	Train Acc	Test Loss	Test Acc
0	4.11137358	0.112149533	3.039359125	0.280742145
1	2.64808159	0.353241822	2.188696850	0.448029523
2	2.07376506	0.472254673	1.769709856	0.546384074
3	1.66507691	0.577540888	1.411547893	0.632346966
4	1.44412486	0.625000000	1.246795302	0.679171181
5	1.28209421	0.662529206	1.169119302	0.691528982
6	1.08390344	0.708382009	0.968973114	0.744955309
7	1.00620193	0.730432243	0.882916458	0.764558505
8	0.89764431	0.753212617	0.898828749	0.754232124
9	0.78695435	0.785484813	0.644751354	0.816935265
10	0.69651825	0.807827103	0.677475136	0.808978873
11	0.66139633	0.814836449	0.553797855	0.845307421
12	0.60596341	0.831045561	0.533283006	0.847304984
13	0.57218855	0.836010514	0.504230925	0.850826111
14	0.52910589	0.851051402	0.412067141	0.885597237
15	0.50126999	0.861857477	0.429610118	0.879198267
16	0.46718794	0.864778037	0.442321374	0.870192308
17	0.45116315	0.867406542	0.354990175	0.896600758
18	0.38908813	0.889748832	0.331363576	0.898327465
19	0.38107870	0.891793224	0.361018805	0.891082069

Pérdida por época



Precisión por época



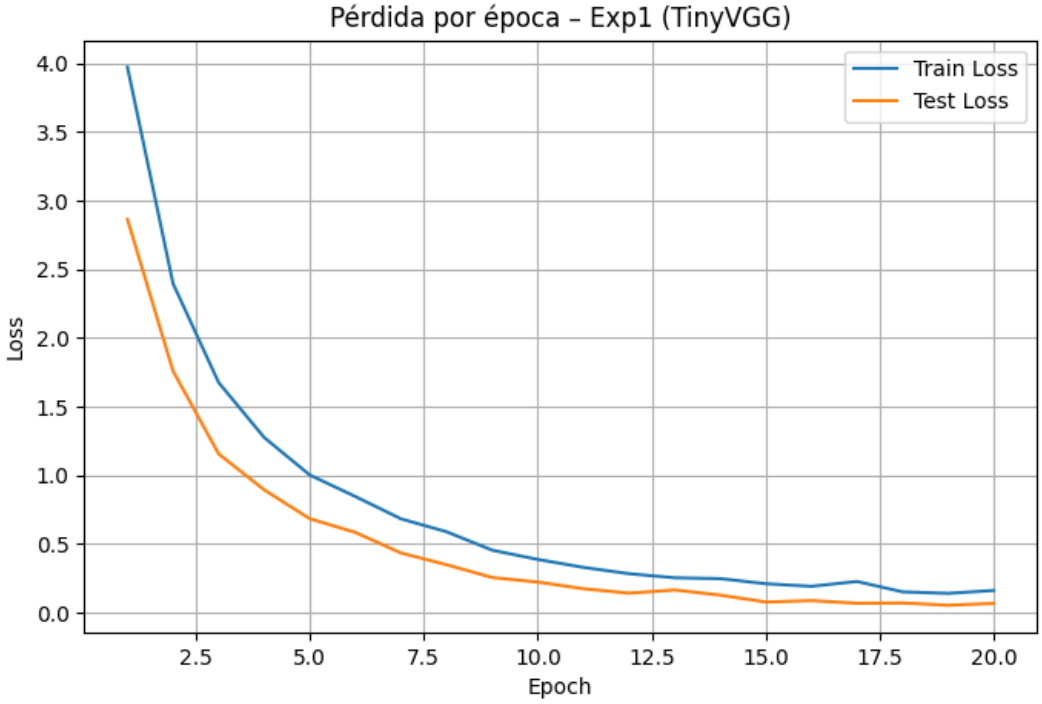
4.2. Experimento 1

Tabla de resultados

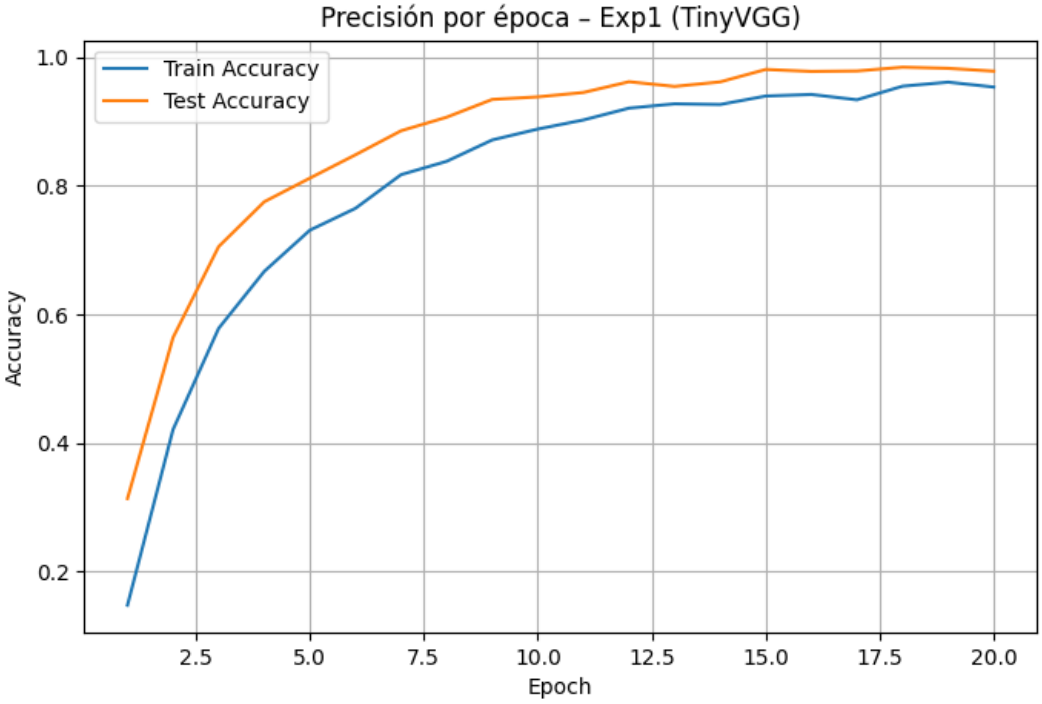
A continuación se presentan los valores obtenidos por época para el experimento:

Epoch	Train Loss	Train Acc	Test Loss	Test Acc
0	3.97068203	0.147196262	2.862922662	0.313210997
1	2.39552793	0.420706776	1.758836034	0.563888137
2	1.67545938	0.578125000	1.156429666	0.705376490
3	1.27530219	0.666764019	0.894181215	0.775121885
4	1.00229956	0.730870327	0.684080536	0.811890574
5	0.84585043	0.764894860	0.584485310	0.848219122
6	0.68346987	0.817318925	0.434868411	0.885597237
7	0.58942338	0.838054907	0.349514641	0.906723998
8	0.45526621	0.871349299	0.256445388	0.934419014
9	0.38803632	0.888288551	0.223467154	0.938380282
10	0.33034518	0.902453271	0.174941269	0.945219393
11	0.28448044	0.920852804	0.142781323	0.961944745
12	0.25471668	0.927570093	0.165882755	0.954665493
13	0.24861964	0.926547897	0.128541745	0.961707746
14	0.21143782	0.939690421	0.077978253	0.981073944
15	0.19257259	0.942172897	0.088489450	0.977992958
16	0.22785539	0.933849299	0.069219663	0.978670098
17	0.15210514	0.955023364	0.071112022	0.984595070
18	0.14128143	0.961302570	0.055099695	0.982834507
19	0.16229625	0.953709112	0.067963865	0.978433099

Pérdida por época



Precisión por época



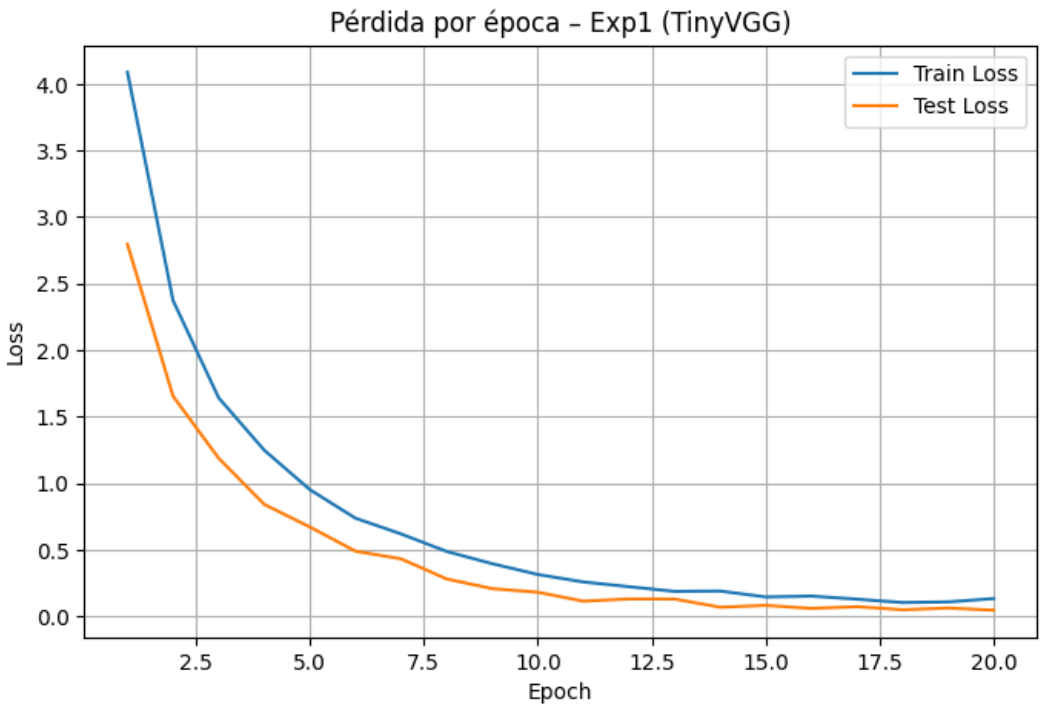
4.3. Experimento 2

Tabla de resultados

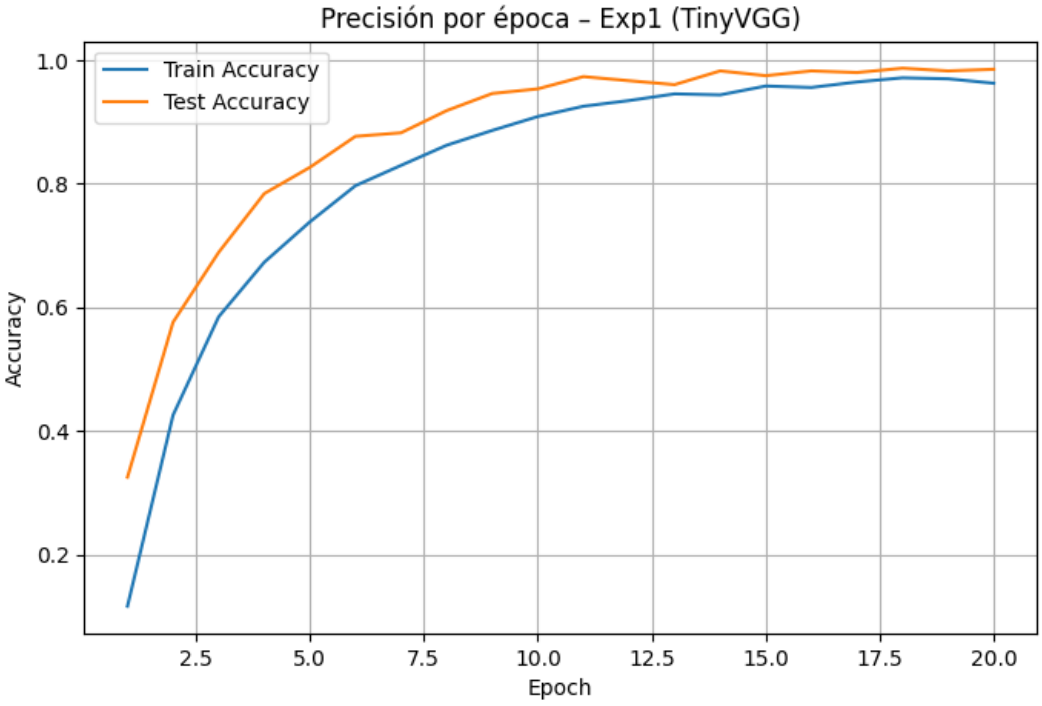
A continuación se presentan los valores obtenidos por época para el experimento:

Epoch	Train Loss	Train Acc	Test Loss	Test Acc
0	4.088593905	0.116530374	2.795660304	0.325230228
1	2.372895450	0.425671729	1.654418094	0.575975081
2	1.642323690	0.584696262	1.189090280	0.688888137
3	1.248933110	0.672897196	0.841891631	0.783924702
4	0.952687141	0.738171729	0.671941164	0.826381365
5	0.738125737	0.796728972	0.490434559	0.876557421
6	0.619783142	0.829585280	0.432797104	0.882279252
7	0.488321141	0.862149533	0.281721021	0.918133803
8	0.396144243	0.886098131	0.208935518	0.945862676
9	0.315260905	0.908586449	0.181985120	0.953345070
10	0.258941366	0.925233645	0.114604500	0.973151408
11	0.223462396	0.934433411	0.131154108	0.966549296
12	0.188152347	0.945239486	0.130475516	0.960184182
13	0.190737811	0.943633178	0.069460039	0.982394366
14	0.146411861	0.957943925	0.083032999	0.974708830
15	0.152894889	0.955607477	0.060067534	0.982394366
16	0.129736418	0.964515187	0.073389196	0.979753521
17	0.104401082	0.971232477	0.050159291	0.986795775
18	0.109241575	0.969626168	0.063123156	0.982394366
19	0.134048269	0.962470794	0.046899996	0.985035211

Pérdida por época



Precisión por época



5 Análisis de Resultados

Los tres experimentos permiten observar claramente el impacto de las técnicas de regularización y de la ingeniería en los datos sobre la precisión final del modelo.

5.1. Comparación general

Experimento	Mejor Test Acc	Epoch	Test Acc Final	Test Loss Final
Exp0 (Base)	0.89108	19	0.89108	0.36102
Exp1 (Augmentación)	0.98283	18	0.97843	0.06796
Exp2 (AdamW + L2)	0.98679	17	0.98503	0.04689

- **Exp0** logra una precisión máxima cercana al 89 %.
- **Exp1** mejora sustancialmente la generalización alcanzando valores entre 97 % y 98 %.
- **Exp2** obtiene el mejor desempeño, con una precisión estable que supera el 98 %.

5.2. Discusión

- El aumento de datos contribuyó significativamente a la capacidad de generalización del modelo.
- AdamW permitió controlar el sobreajuste y estabilizar las curvas de pérdida.
- TinyVGG, pese a ser una arquitectura ligera, fue capaz de aprender representaciones útiles gracias al preprocesamiento y ajustes aplicados.

6 Conclusiones

A lo largo de este proyecto se demostró que pequeñas modificaciones en la configuración de una CNN pueden tener un impacto considerable en su desempeño. Entre las estrategias más efectivas se encontraron:

- Uso de data augmentation controlado.
- Optimización mediante AdamW.
- Regularización L2 para reducir overfitting.

El mejor modelo alcanzó una precisión superior al 98 %, lo cual evidencia que TinyVGG, a pesar de su simplicidad, puede ser altamente competitivo cuando se acompaña de técnicas adecuadas de entrenamiento.

Este trabajo permite concluir que la combinación de una arquitectura ligera con técnicas de regularización y augmentación es suficiente para resolver con éxito problemas de clasificación moderadamente complejos.