DuocUC | ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

INFORME DE PROYECTO
FORMA C
"ETIQUETADO DE IMÁGENES PARA GOBIERNO DE CHILE"

PROFESOR: MARCO ANTONIO JAPKE ADRIASOLA
DEEP LEARNING
DLY0100_003V

Integrantes:

Cristóbal Cabezas
Jorge López

SANTIAGO, 10 DE MAYO DE 2024

Índice

Resumen ejecutivo

Este proyecto se enfoca en el desarrollo de un sistema de etiquetado de imágenes utilizando técnicas de Deep Learning, específicamente el conjunto de datos CIFAR-10. El objetivo principal es construir un modelo de aprendizaje automático capaz de clasificar imágenes en 10 clases distintas con alta precisión.

CIFAR-10 es un conjunto de datos que consta de 60,000 imágenes en color de tamaño 32x32, distribuidas en 10 categorías diferentes, con 6,000 imágenes por clase. Estas imágenes se dividen en 50,000 para entrenamiento y 10,000 para pruebas.

El gobierno de Chile busca implementar inteligencia artificial en su página web para etiquetar imágenes de diversos dominios. La necesidad de este proyecto radica en la creciente importancia de automatizar tareas que involucran grandes volúmenes de datos, en particular imágenes, y en la capacidad de las máquinas para tomar decisiones con mayor exactitud y escalabilidad que los humanos.

Para abordar este desafío, se utilizará la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining), que consta de seis fases: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue.

El éxito de este proyecto no solo radica en la creación de un modelo de clasificación de imágenes preciso, sino también en la implementación de un sistema escalable y eficiente que pueda manejar el etiquetado de imágenes en tiempo real para satisfacer las necesidades del gobierno de Chile en su página web.

Descripción del problema a resolver

El Gobierno de Chile busca implementar inteligencia artificial en su página web para automatizar el etiquetado de imágenes de diversos dominios. Actualmente, la clasificación manual de imágenes es un proceso lento y costoso, que impide una gestión eficiente de grandes volúmenes de datos visuales.

El problema es la necesidad de clasificar imágenes de manera precisa y eficiente en 10 categorías diferentes, crucial para mejorar la experiencia del usuario y optimizar la navegación en la página web gubernamental. Esto implica:

Clasificación precisa de imágenes: El sistema debe reconocer con precisión los objetos o elementos de las imágenes, asignándoles una de las 10 clases predefinidas.

Eficiencia y escalabilidad: El sistema debe ser capaz de manejar grandes volúmenes de imágenes de manera eficiente y escalable, sin comprometer la velocidad de procesamiento.

Adaptabilidad a diversos dominios: Las imágenes pueden provenir de diferentes dominios, como naturaleza, personas, objetos, etc. El sistema debe ser capaz de adaptarse y clasificar imágenes de cualquier dominio con precisión.

Automatización del proceso: La clasificación manual de imágenes es laboriosa y propensa a errores. El sistema debe automatizar este proceso para reducir el tiempo y los costos asociados con la clasificación manual.

Desarrollo de un modelo robusto: Se requiere la construcción de un modelo de aprendizaje profundo robusto y generalizable que pueda manejar la complejidad de las imágenes y generalizar patrones a partir de un conjunto de datos de entrenamiento limitado.

El desafío es desarrollar un sistema de etiquetado de imágenes basado en inteligencia artificial preciso, eficiente, escalable y adaptable a diferentes dominios, para mejorar la gestión de datos visuales en la página web del Gobierno de Chile.

Descripción técnica de la solución:

Para abordar el problema planteado, se propone una solución basada en técnicas de Deep Learning, utilizando el conjunto de datos CIFAR-10 y siguiendo la metodología CRISP-DM.

1. Comprensión del Negocio:

    1.1. Identificar las necesidades y objetivos del Gobierno de Chile en cuanto al etiquetado de imágenes en su página web.
    1.2. Definir las clases de imágenes relevantes para la clasificación.

1.3. Establecer métricas de evaluación, como precisión, recall y F1-score.

2. Comprensión de los Datos:

   2.1. Explorar el conjunto de datos CIFAR-10 para comprender su estructura y características.
   2.2. Visualizar ejemplos de imágenes de cada clase para entender la complejidad y variabilidad de los datos.
   2.3. Realizar un análisis estadístico de los datos para identificar posibles desafíos, como desequilibrios de clase.

3. Preparación de los Datos:

   3.1. Normalizar las imágenes para asegurar que tengan la misma escala y rango de valores.
   3.2. Dividir el conjunto de datos en conjuntos de entrenamiento, validación y prueba.
   3.3. Aplicar técnicas de aumento de datos, como rotaciones, desplazamientos y reflejos, para aumentar la variabilidad del conjunto de datos de entrenamiento.

4. Modelado:

   4.1. Seleccionar una arquitectura de red neuronal convolucional (CNN) adecuada para el problema, como ResNet, VGG o DenseNet.
   4.2. Diseñar y entrenar el modelo utilizando el conjunto de datos de entrenamiento y validación.
   4.3. Utilizar técnicas como transferencia de aprendizaje para inicializar los pesos del modelo con una red preentrenada en un conjunto de datos similar.
   4.4. Ajustar los hiperparámetros del modelo, como la tasa de aprendizaje, el tamaño del lote y la función de pérdida, mediante validación cruzada.

5. Evaluación:

   5.1. Evaluar el rendimiento del modelo utilizando el conjunto de datos de prueba.
   5.2. Analizar las métricas de evaluación para comprender la precisión y el rendimiento del modelo en la clasificación de imágenes.
   5.3. Realizar pruebas adicionales con imágenes nuevas para evaluar la capacidad de generalización del modelo.

6. Despliegue:

   6.1. Implementar el modelo entrenado en un entorno de producción, preferiblemente en la infraestructura en la nube.
   6.2. Configurar una interfaz de usuario amigable para que los usuarios puedan cargar imágenes y recibir las etiquetas predichas.

6.3. Realizar pruebas exhaustivas del sistema desplegado para garantizar su funcionamiento correcto y eficiente en tiempo real.

La solución propuesta busca desarrollar un sistema de etiquetado de imágenes preciso y eficiente, capaz de automatizar el proceso de clasificación de imágenes en la página web del Gobierno de Chile, mejorando así la gestión de datos visuales y la experiencia del usuario.

Fundamentación técnica de los modelos escogidos

Para el desarrollo del presente trabajo, se utilizará el siguiente flujo:

1) Se utilizarán 3 optimizadores:
    a. SGD: El optimizador SGD (Stochastic Gradient Descent) es un algoritmo de optimización utilizado en el entrenamiento de modelos de aprendizaje profundo. Funciona actualizando iterativamente los pesos de la red neuronal en la dirección opuesta al gradiente de la función de pérdida con respecto a los pesos. El gradiente se calcula utilizando un subconjunto aleatorio de ejemplos de entrenamiento en cada iteración, lo que lo hace "estocástico". SGD utiliza una tasa de aprendizaje para controlar el tamaño de los pasos de actualización de los pesos. Es uno de los optimizadores más utilizados debido a su simplicidad y eficacia.
    b. Adam: (Adaptive Moment Estimation) es un algoritmo de optimización que combina los conceptos de Momentum y RMSprop. Es un optimizador popular y eficiente para entrenar redes neuronales.
    c. RMSProp: Adam también utiliza un término similar al RMSprop, que adapta la tasa de aprendizaje de manera individual para cada parámetro en función de la magnitud de sus gradientes. Esto ayuda a controlar la velocidad de aprendizaje para cada parámetro de manera más adaptativa, lo que puede ser beneficioso en problemas con gradientes de magnitudes variables.
2) A cada de uno de estos optimizadores se les aplicarán 3 funciones de activación:
    a. ReLu Rectified Linear Unit): ReLU es una función de activación no lineal que asigna cualquier valor negativo a cero y deja los valores positivos intactos. Es una de las funciones de activación más utilizadas debido a su eficiencia computacional y a su capacidad para mitigar el problema de desvanecimiento del gradiente. ReLU permite que la red neuronal aprenda de manera más rápida y efectiva, especialmente en problemas de clasificación y regresión.
    b. Tanh (Tangente Hiperbólica): Tanh es una función de activación no lineal que asigna valores reales a un rango entre -1 y 1. A diferencia de ReLU, Tanh produce salidas centradas en cero, lo que puede ayudar a mitigar el problema del desvanecimiento del gradiente en redes neuronales profundas. Tanh es útil en capas ocultas para capturar relaciones no lineales y es comúnmente utilizada en arquitecturas de redes neuronales.

c. Sigmoid (Función Sigmoide): La función de activación Sigmoid es una función no lineal que transforma los valores de entrada a un rango entre 0 y 1. Se define como f(x) = 1 / (1 + exp(-x)). La salida de la función Sigmoid se interpreta como una probabilidad, donde valores cercanos a 1 indican alta probabilidad y valores cercanos a 0 indican baja probabilidad. La función Sigmoid es comúnmente utilizada en capas ocultas de redes neuronales para introducir no linealidades y en capas de salida para problemas de clasificación binaria.

Detalle de ajustes en las redes

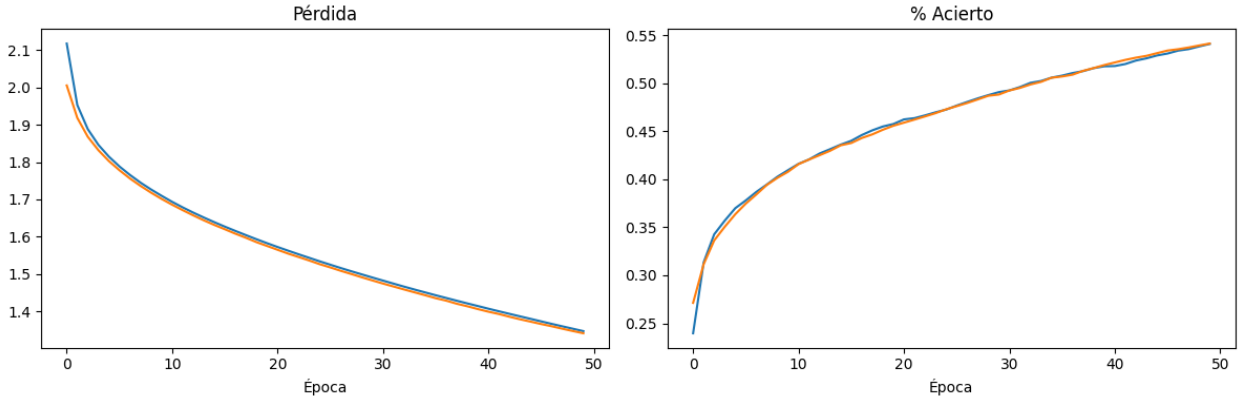
- Cada de unas iteraciones se realizarán utilizando 1, 2 y 3 capas ocultas.
  - La primera capa utilizará 512 neuronas.
  - La segunda, 256.
  - La tercera, 256
- Todos los modelos a entrenar se iterarán en 50 épocas (epochs)
- Salvo los casos claramente señalados, los optimzadores utilizarán sus valores por defecto.
- Asimismo, a todos los modelos se les medirán los resultados de los datos entrenamiento (accuracy, loss), los cuales se contrastarán con los resultados de los datos de prueba (val_accuracy, val_loss).

Visualizaciones:

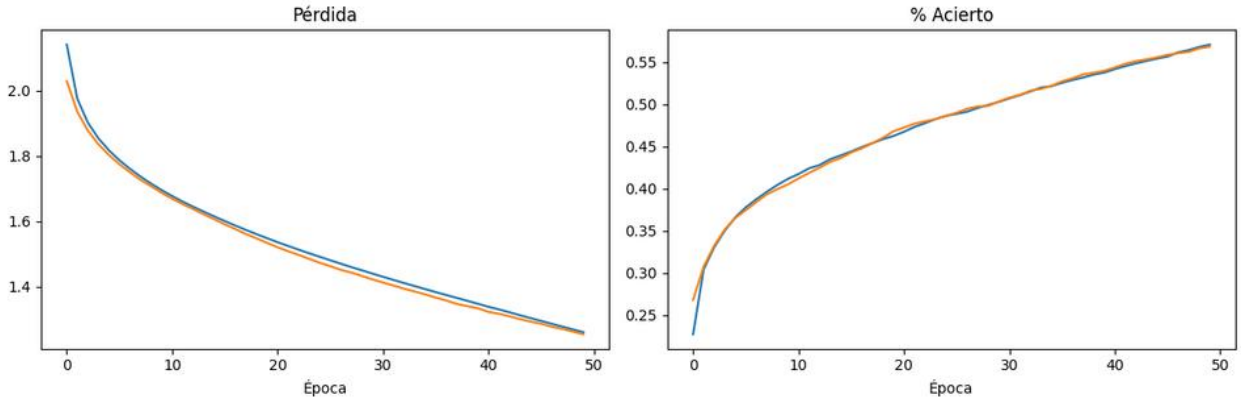
1. SGD

1.1. SGD + ReLu

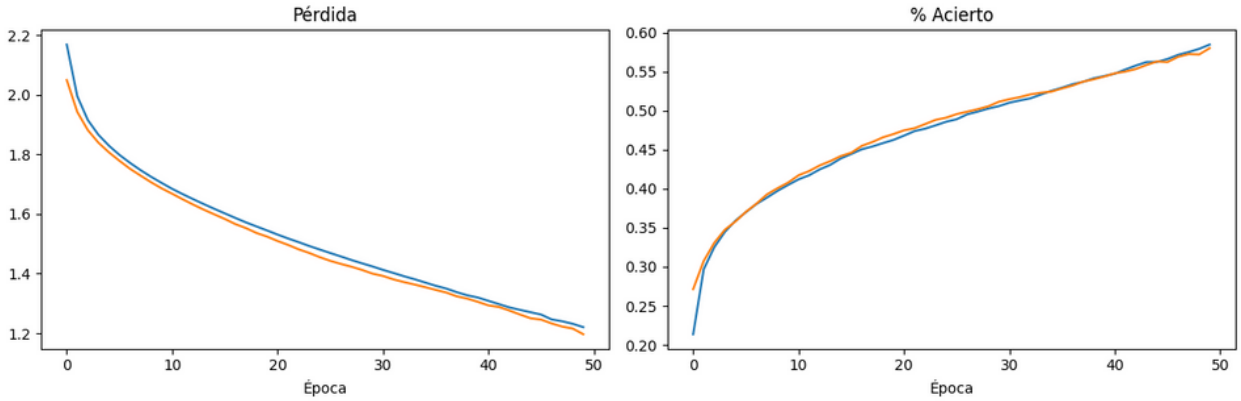
1.1.1. 1 Capa



1.1.2. 2 Capas



1.1.3. 3 Capas

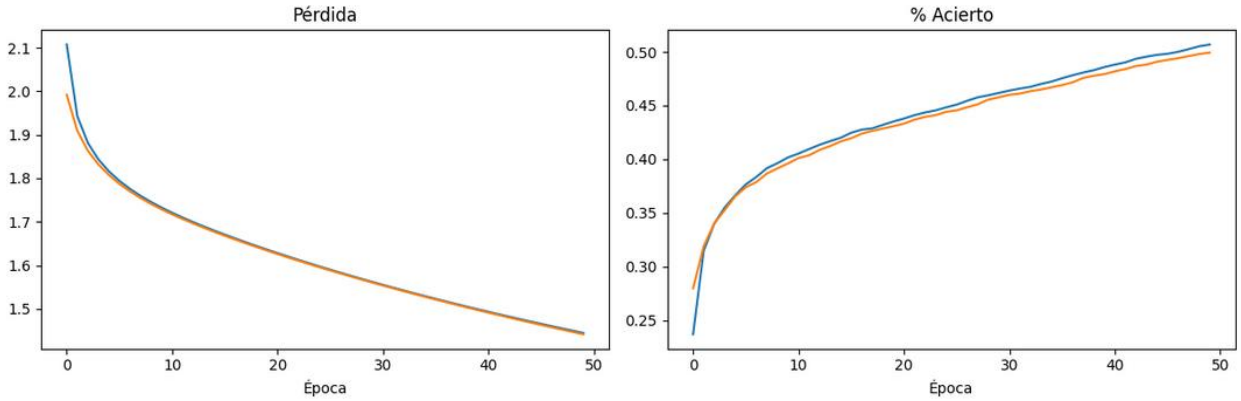


1.1.4. Resultados

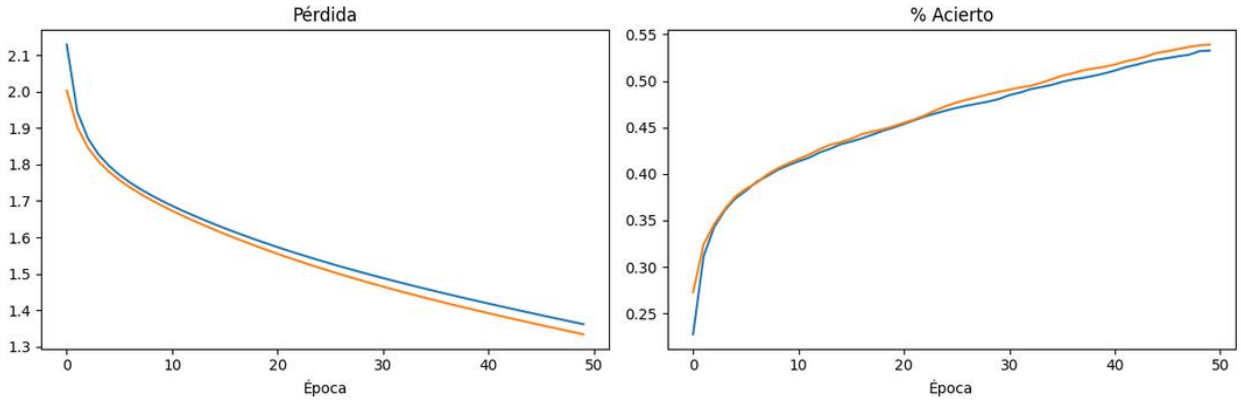
	Relu				
Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*
1	0,5448	1,3521	0,5414	1,3416	82
2	0,5731	1,2609	0,5685	1,2531	60
3	**0,59**	**1,2237**	**0,5802**	**1,1967**	**60**

1.2. SGD + Tanh

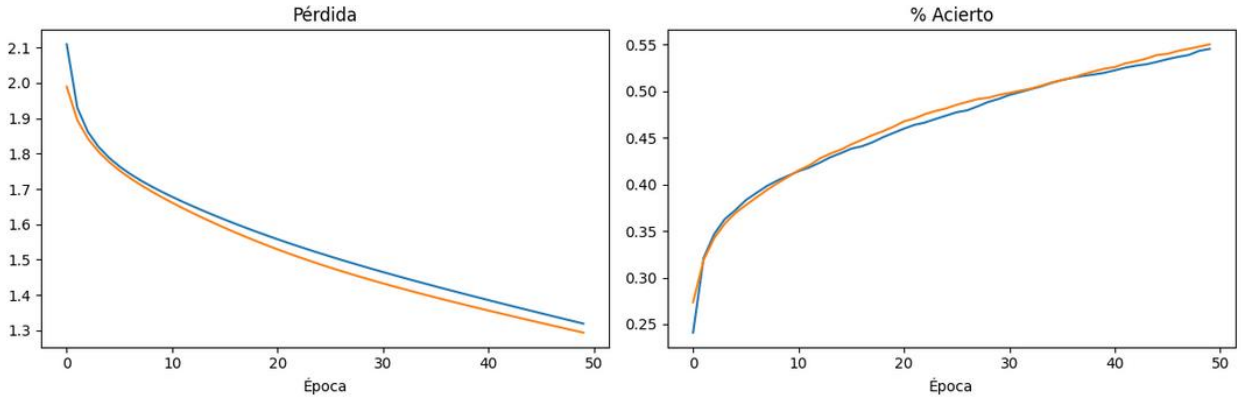
1.2.1. 1 capa



1.2.2. 2 capas



1.2.3. 3 capas



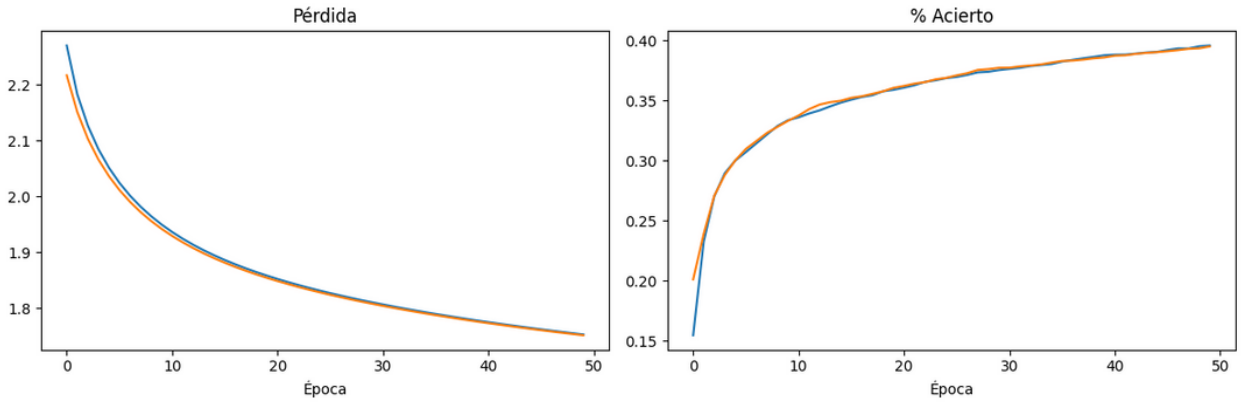
1.2.4. Resultados

	Tanh				
Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*
1	0,5083	1,4454	0,4993	1,4415	62

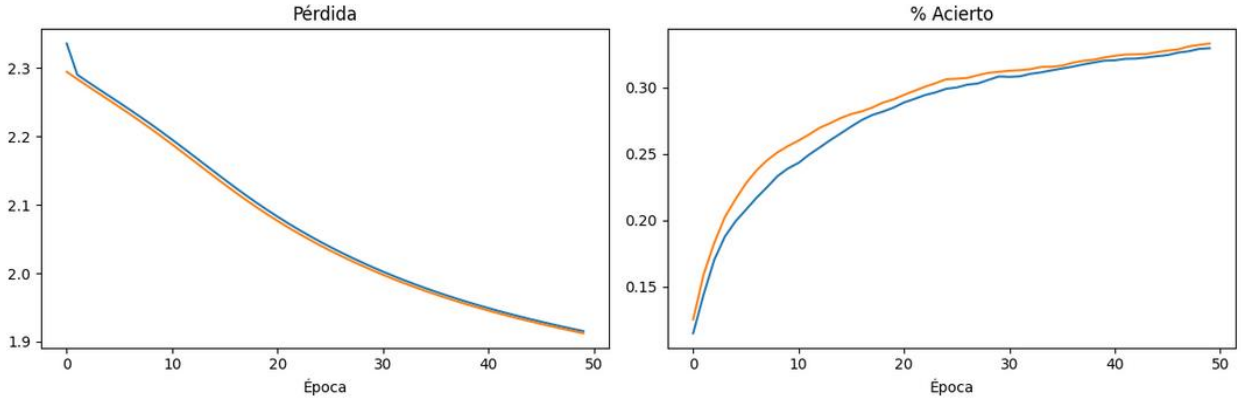
2	0,5334	1,3618	0,5391	1,3339	65
3	**0,5454**	**1,3208**	**0,5503**	**1,2933**	**68**

1.3. SGD + Sigmoid

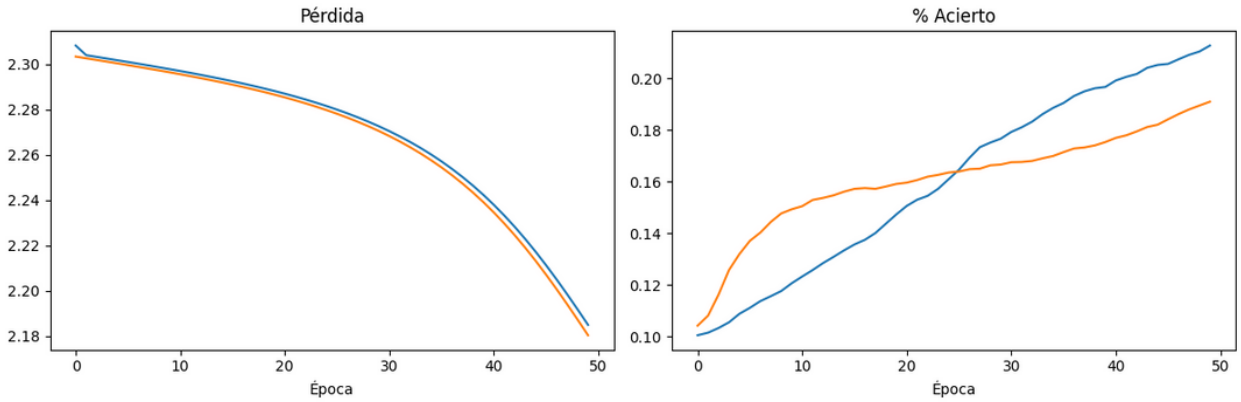
1.3.1. 1 Capa



1.3.2. 2 Capas



1.3.3. 3 Capas



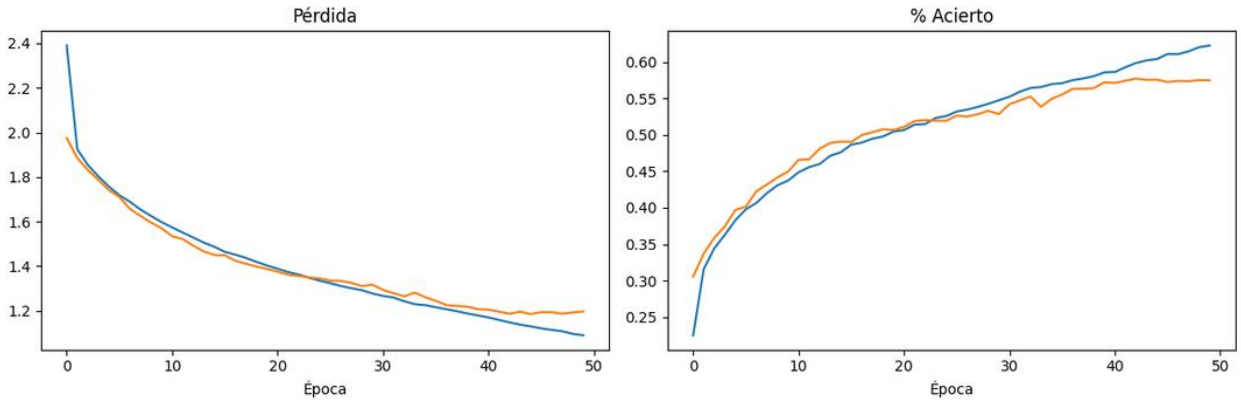
1.3.4. Resultados

	SGD + Sigmoid				
Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*
1	**0,4004**	**1,7536**	**0,3949**	**1,7515**	**59**
2	0,3269	1,9224	0,3330	1,9123	65
3	0,2126	2,1884	0,1909	2,1804	67

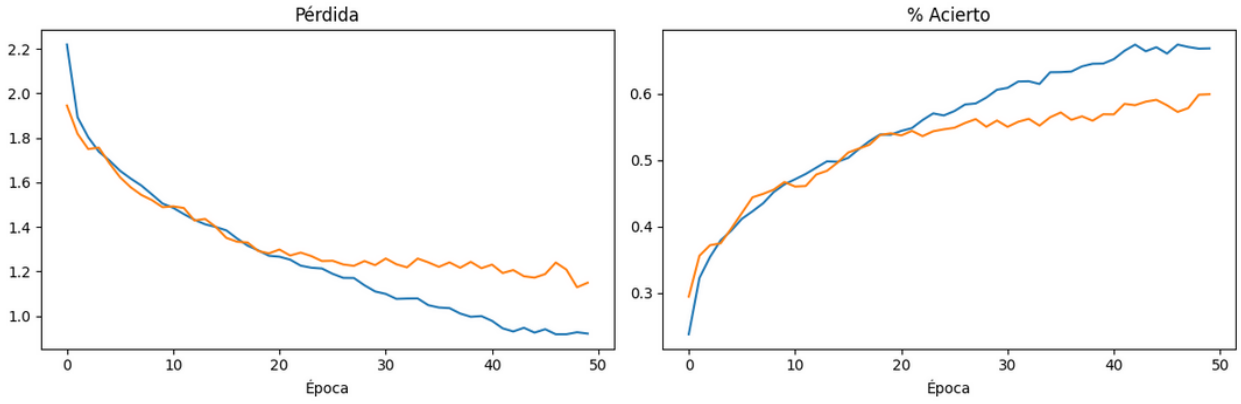
2. ADAM

2.1. ADAM + ReLu

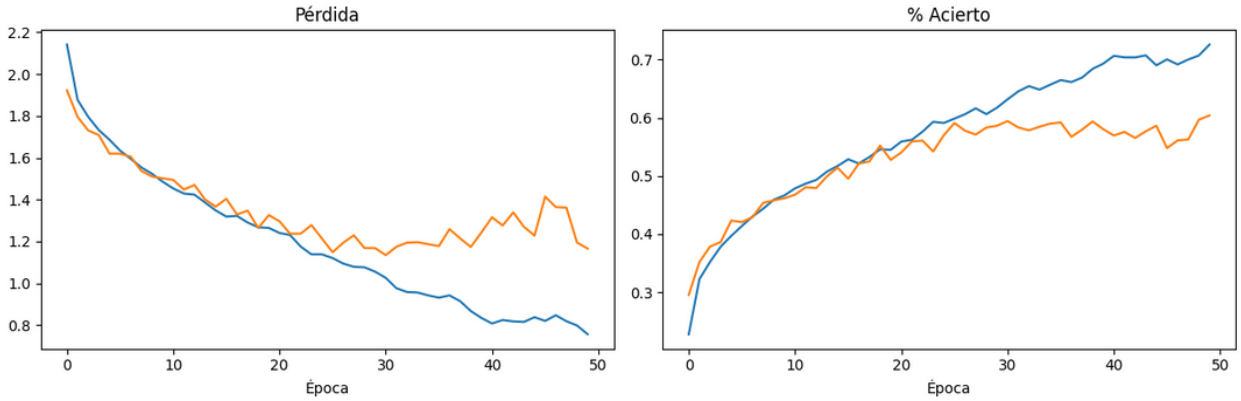
2.1.1. 1 Capa



2.1.2. 2 Capas



2.1.3. 3 Capas

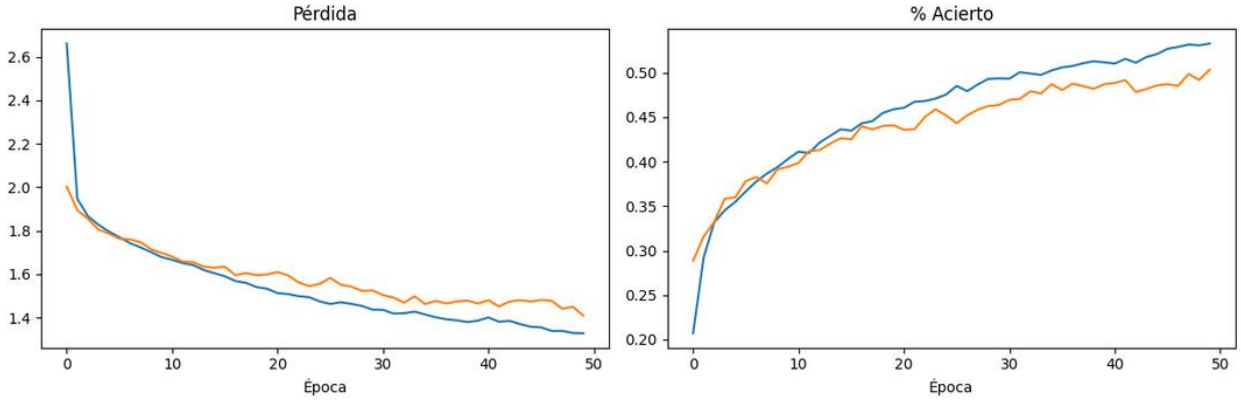


2.1.4. Resultados

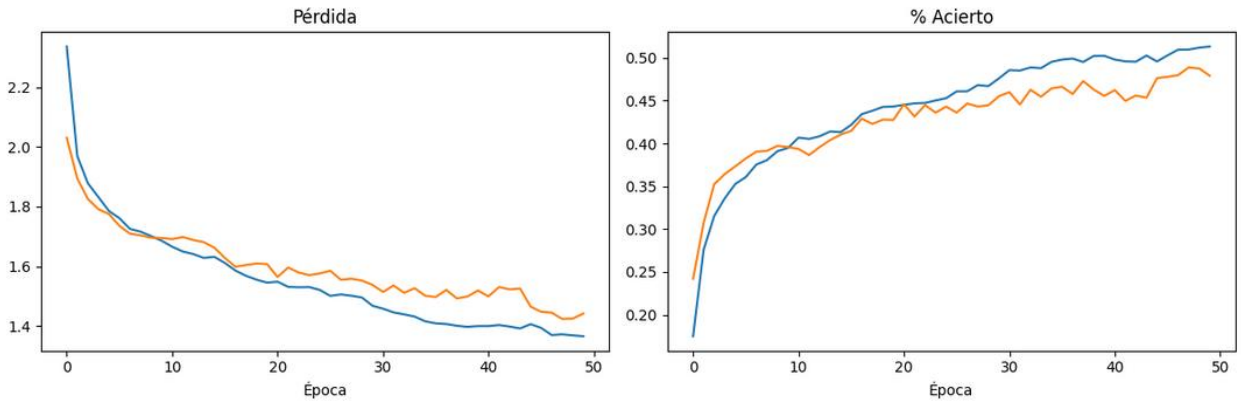
	ADAM + Relu				
Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*
1	0,6154	1,1054	0,5748	1,1962	174
2	0,6713	0,9236	0,5989	1,1495	180
3	**0,7160**	**0,7959**	**0,6039**	**1,1658**	**299**

2.2. ADAM + Tanh

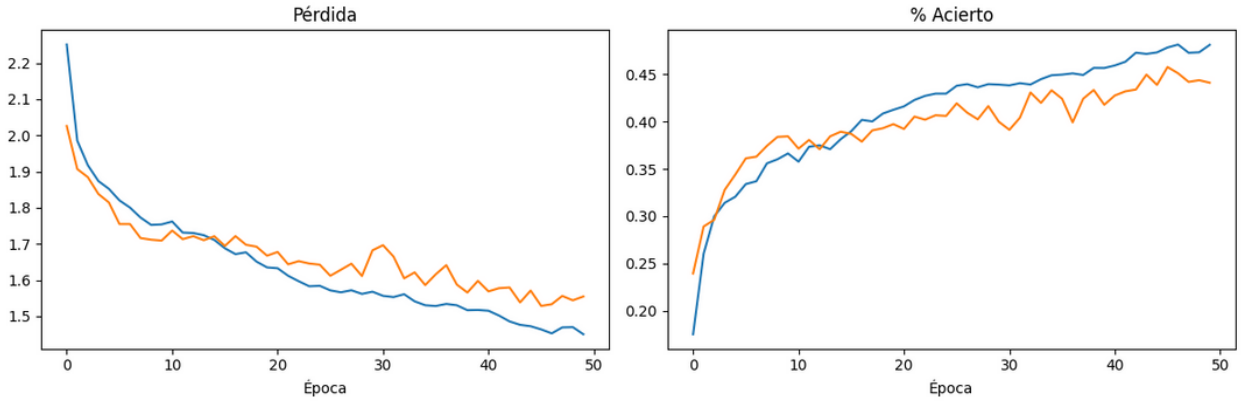
2.2.1. 1 Capa



2.2.2. 2 Capas



2.2.3. 3 Capas

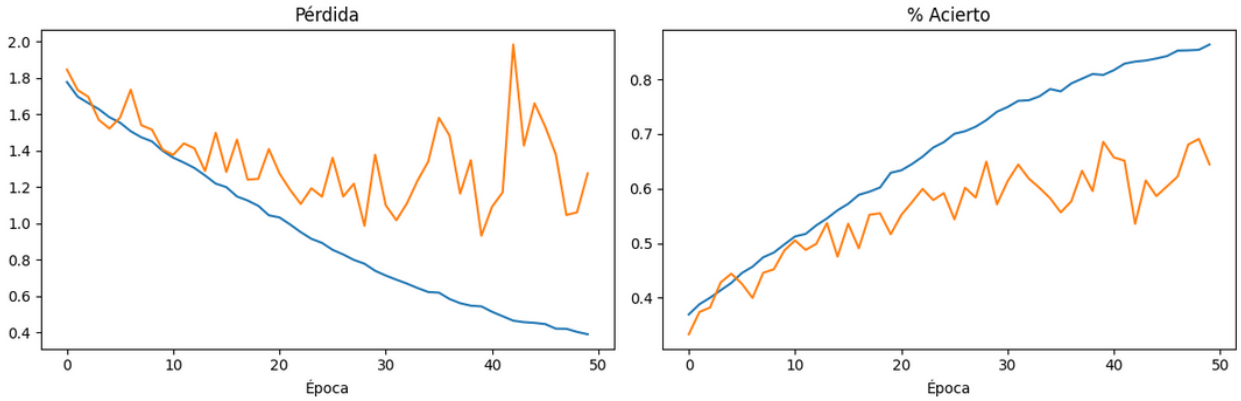


2.2.4. Resultados

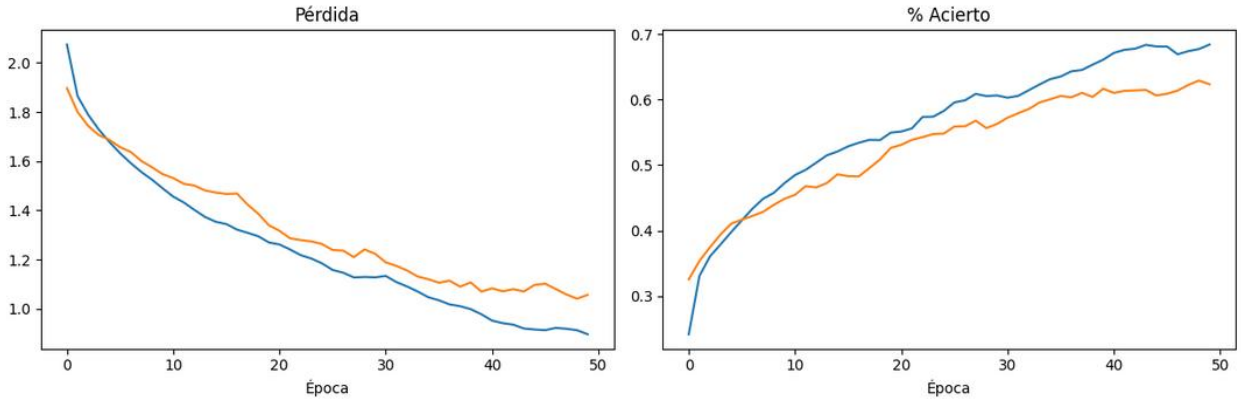
	Adam + Tanh				
Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*
1	**0,5350**	**1,3251**	**0,5034**	**1,4098**	**158**
2	0,5191	1,3531	0,4791	1,4417	183
3	0,4814	1,4582	0,4411	1,5545	193

2.3. ADAM + Sigmoid

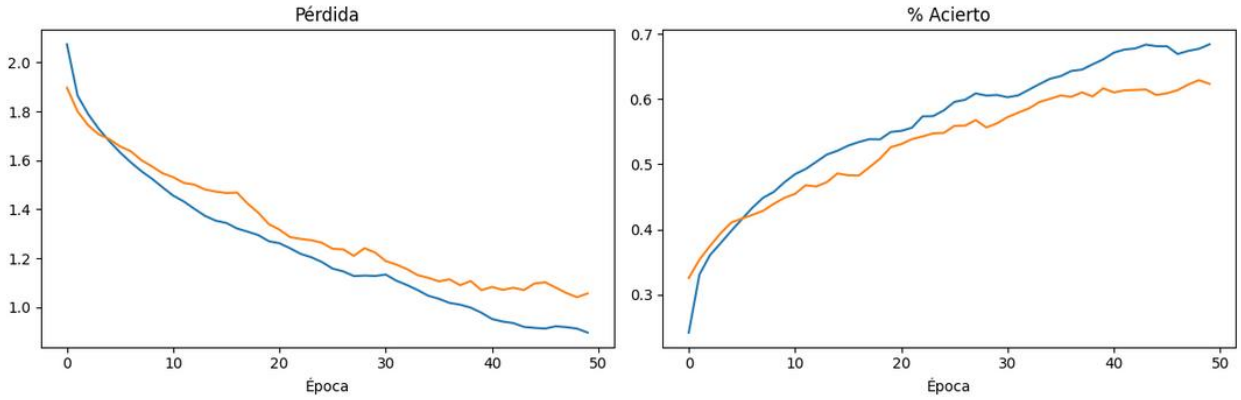
2.3.1. 1 Capas



2.3.2. 2 Capas



2.3.3. 3 Capas



2.3.4. Resultados

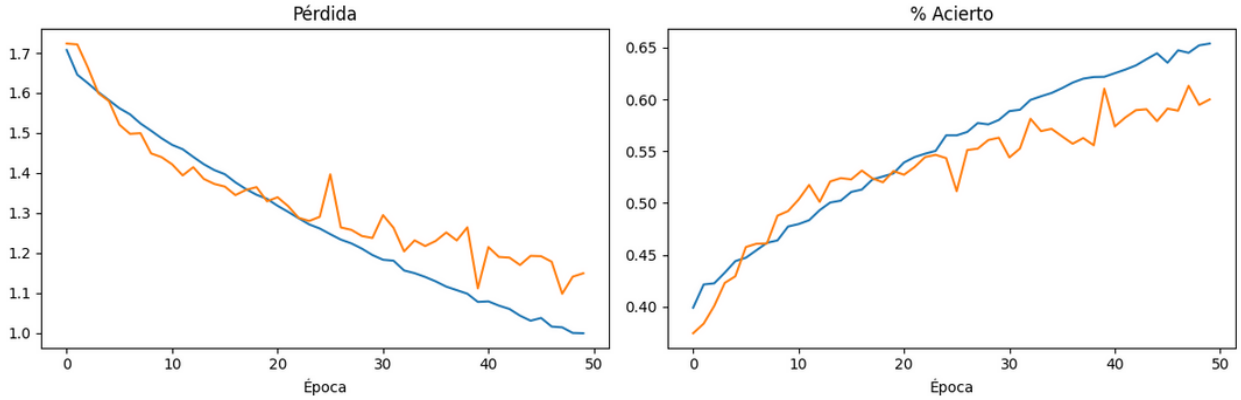
	ADAM + Sigmoid				
Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*

1	**0,7184**	**0,8806**	**0,6015**	**1,1168**	**175**
2	0,6796	0,9099	0,6235	1,0565	149
3	0,6276	1,0324	0,5667	1,2117	139

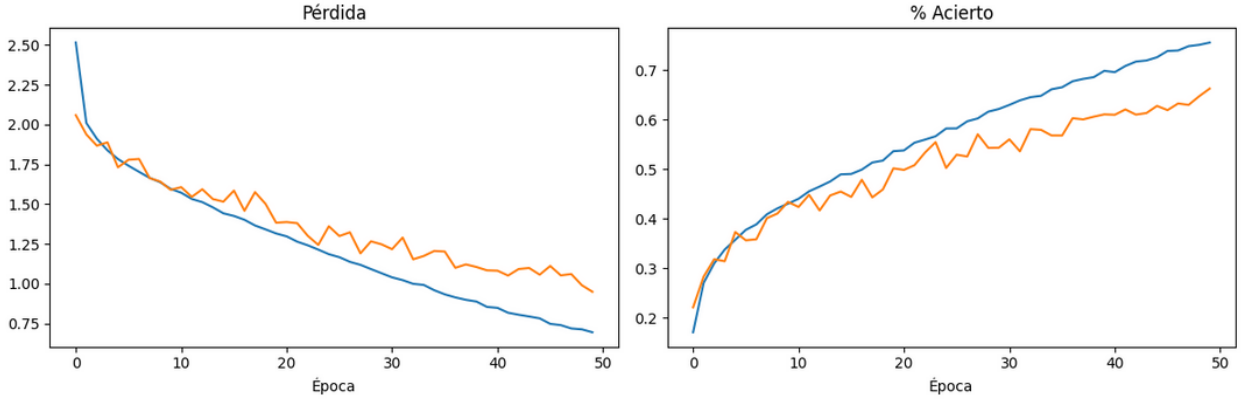
3. RMSProp

3.1. RMSProp + ReLu

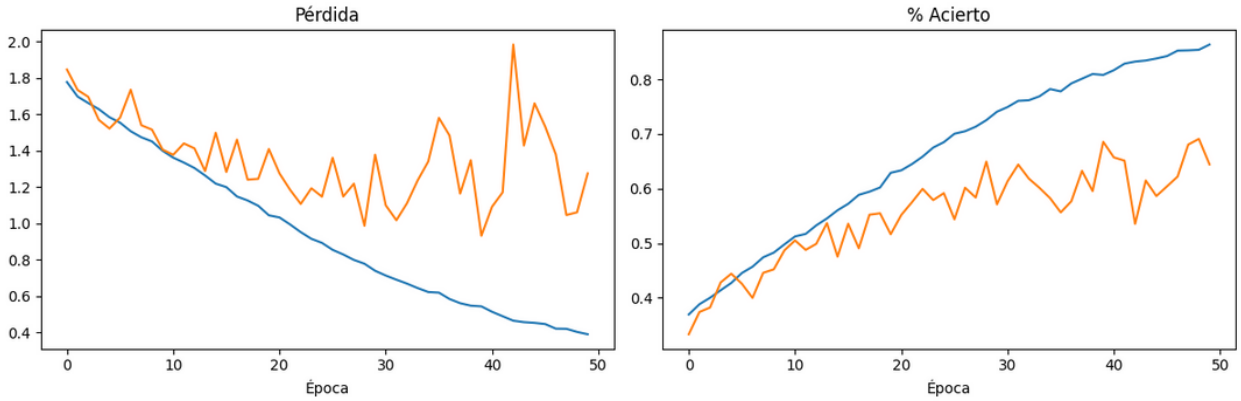
3.1.1. 1 Capa



3.1.2. 2 Capas



3.1.3. 3 Capas

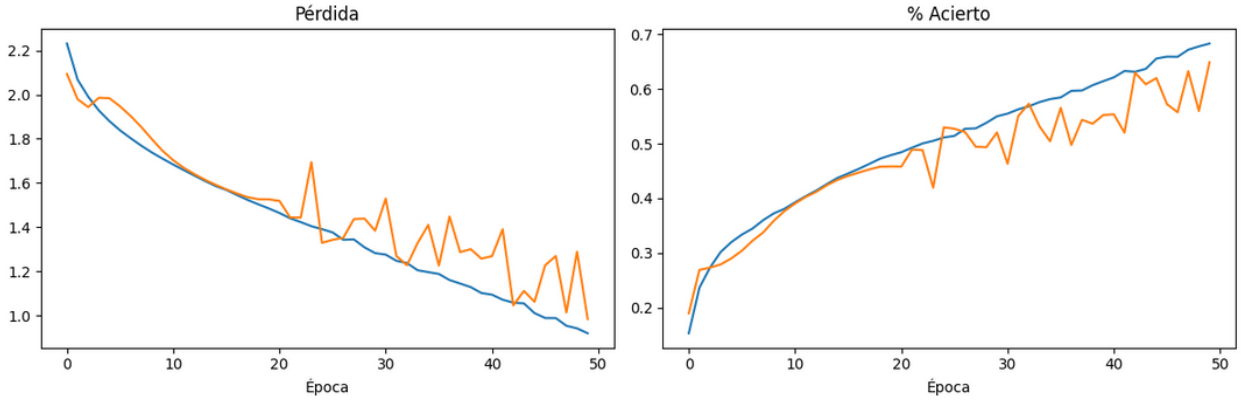


3.1.4. Resultados

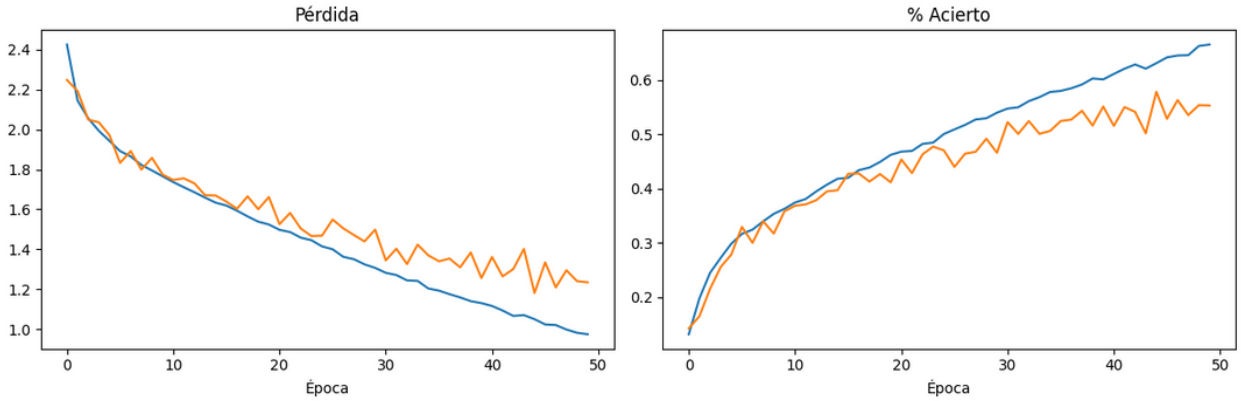
RMSProp + ReLu					
Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*
1	0,6474	1,0088	0,6000	1,1492	119
2	0,7576	0,6916	0,6630	0,9490	123
3	**0,8641**	**0,3901**	**0,6443**	**1,2746**	**132**

3.2. RMSProp + Tanh

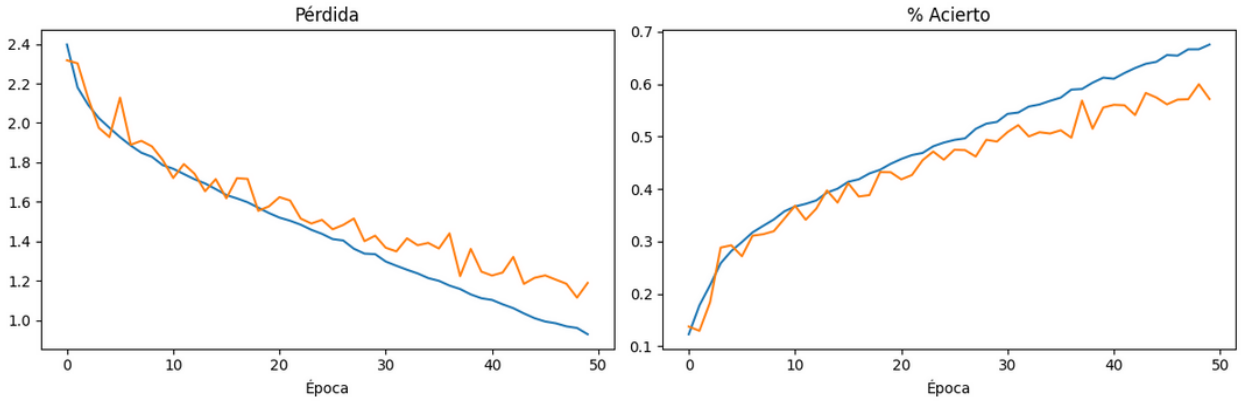
3.2.1. 1 Capa



3.2.2. 2 Capas



3.2.3. 3 Capas

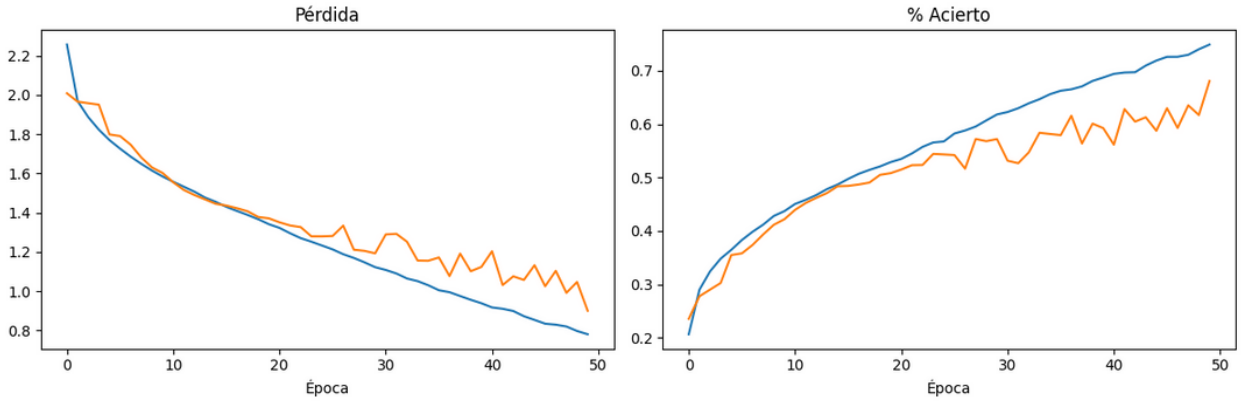


3.2.4. Resultados

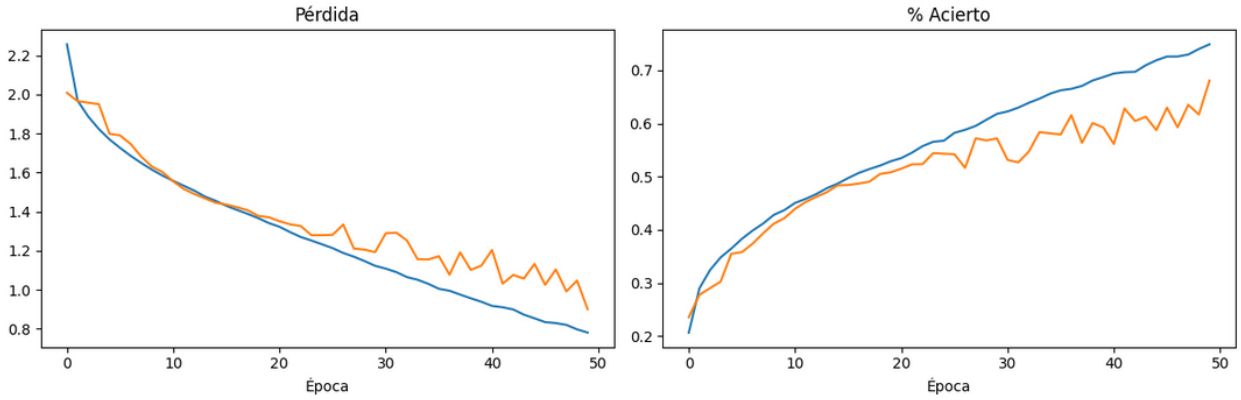
	RMSProp + Tanh				
Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*
1	0,6218	1,108	0,4997	1,4631	117
2	0,66	0,9886	0,5533	1,2350	126
3	**0,6771**	**0,9359**	**0,5717**	**1,1889**	**129**

3.3. RMSProp + Sigmoid

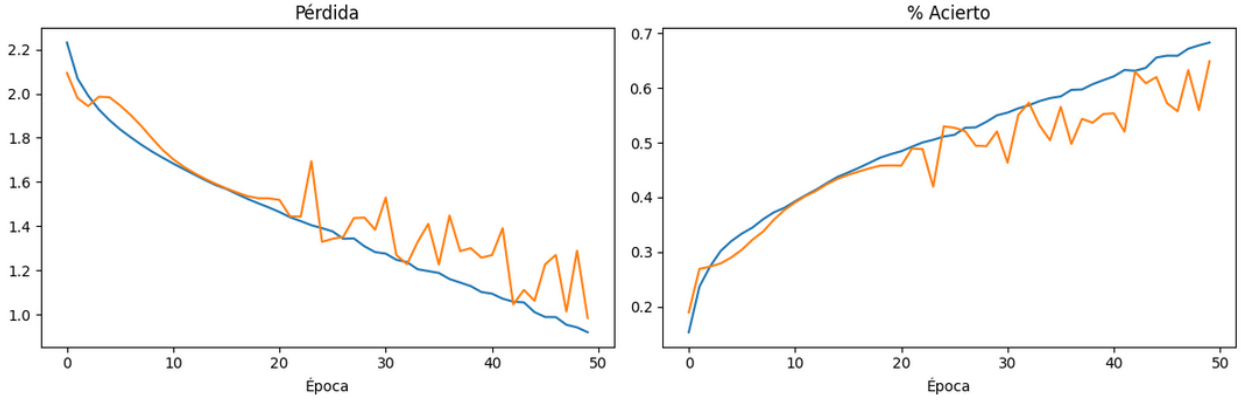
3.3.1. 1 Capa



3.3.2. 2 Capas



3.3.3. 3 Capas



3.3.4. Resultados

	RMSPRop + Sigmoid				
Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*
1	**0,7184**	**0,8806**	**0,6015**	**1,1168**	**175**

2	0,6796	0,9099	0,6235	1,0565	149
3	0,6276	1,0324	0,5667	1,2117	139

Justificación de la solución

Conclusiones respecto del trabajo realizado

Observaciones:

- Más no siempre es mejor: Dependiendo la combinación, rendimiento mejora con menos capas.
- Regularizadores no mejora necesariamente accuracy pero sí el rendimiento de datos de prueba, especialmente con Sigmoid.
- La combinación SGD + ReLu mostró el mejor rendimiento, es decir, la mejor relación de accuracy entre los datos de entrenamiento y datos de prueba.

Premios:

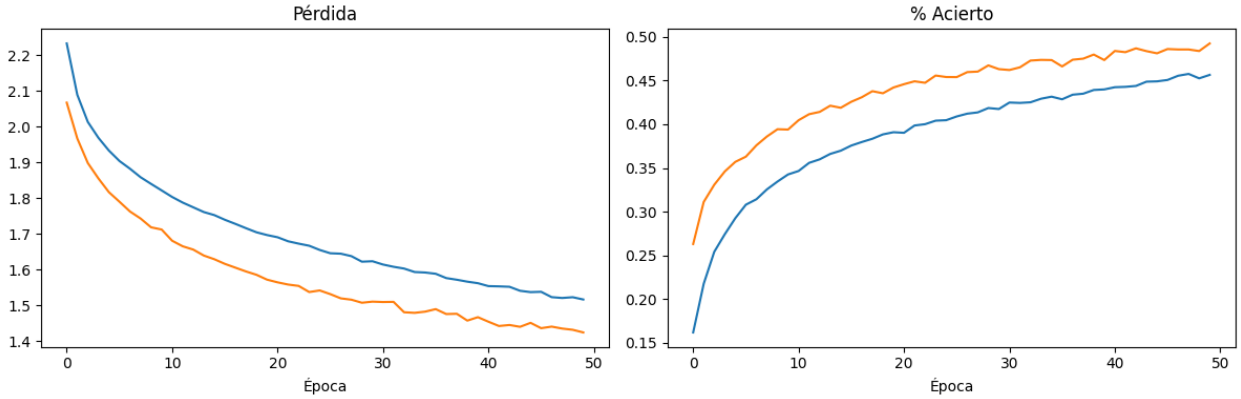
- Mejor optimizador: **SGD**
- Mejor función de activación: **ReLu**
- Mejor regularizador: **Dropout**
- Mejores combinaciones:
  - SGD: ReLu + 3 capas (60% vs 58%)
  - ADAM: ReLu + 3 capas (71% vs 60%)
  - RMSProp: Tanh + 3 capas (35% vs 41%)
  - Dropout: SGD + ReLu + 3 capas (45% vs 49%)
  - EarlyStopping: Adam + Tanh + 1 capa (40% vs 40%)

Propuesta de mejora al proyecto utilizando arquitecturas especializadas para una versión mejorada

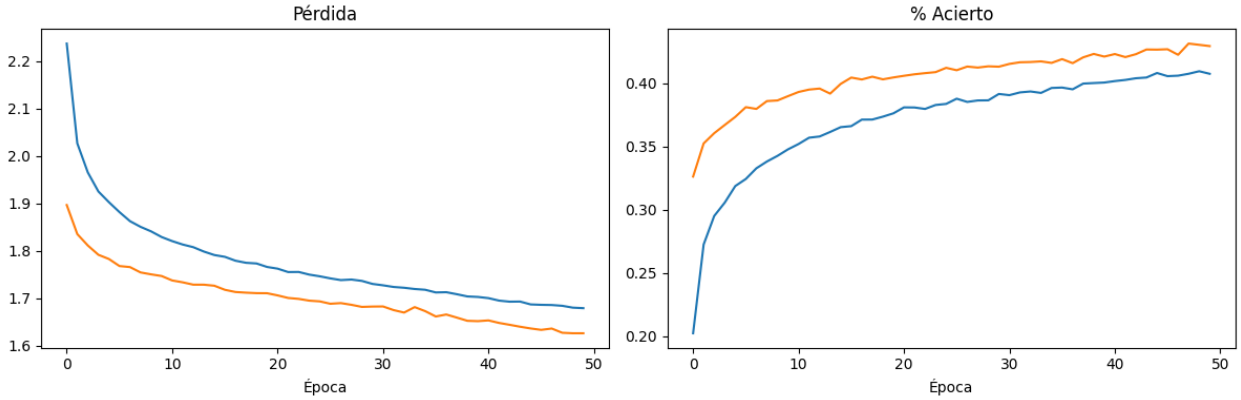
1. Regularización con Dropout

1.1. SGD + Dropout

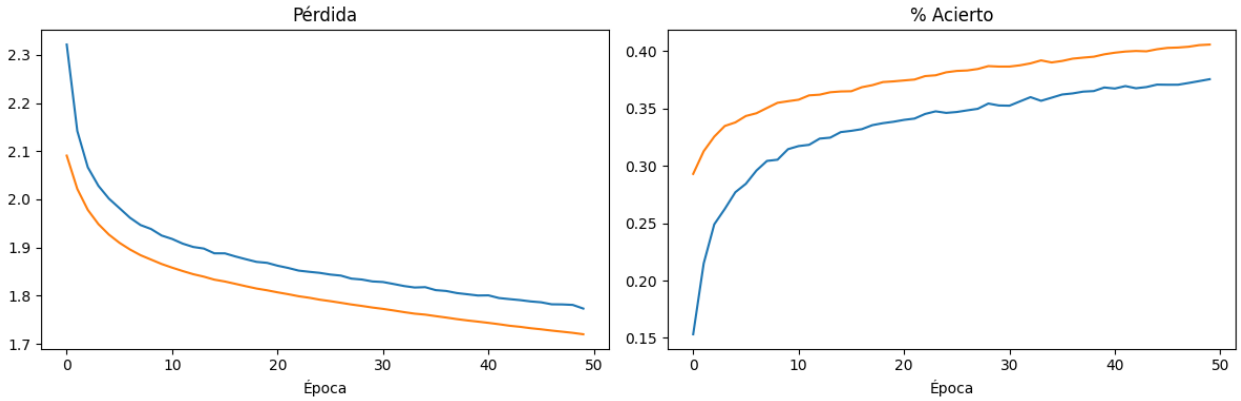
1.1.1. SGD + ReLu + Dropout (3 capas)



1.1.2. SGD + Tanh + Dropout (3 capas)



1.1.3. SGD + Sigmoid (1 capa)

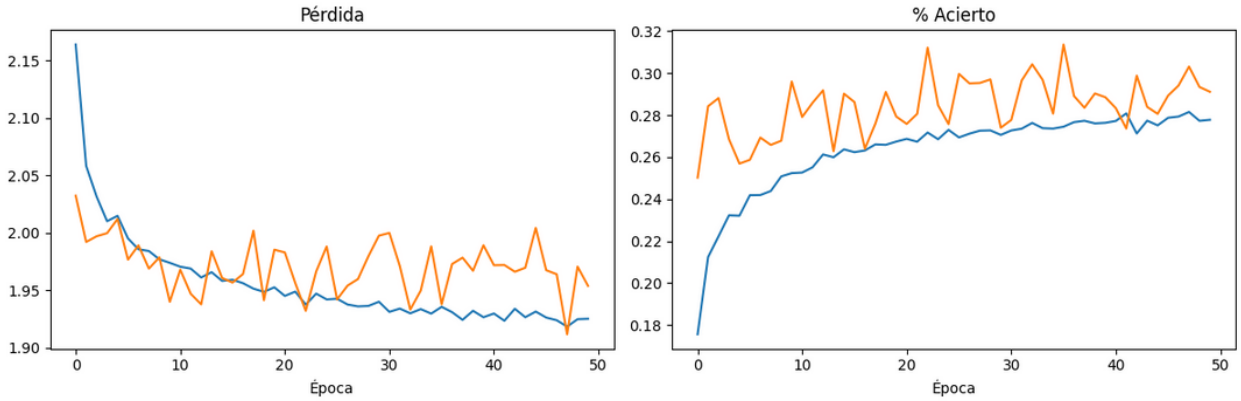


1.1.4. Resultados

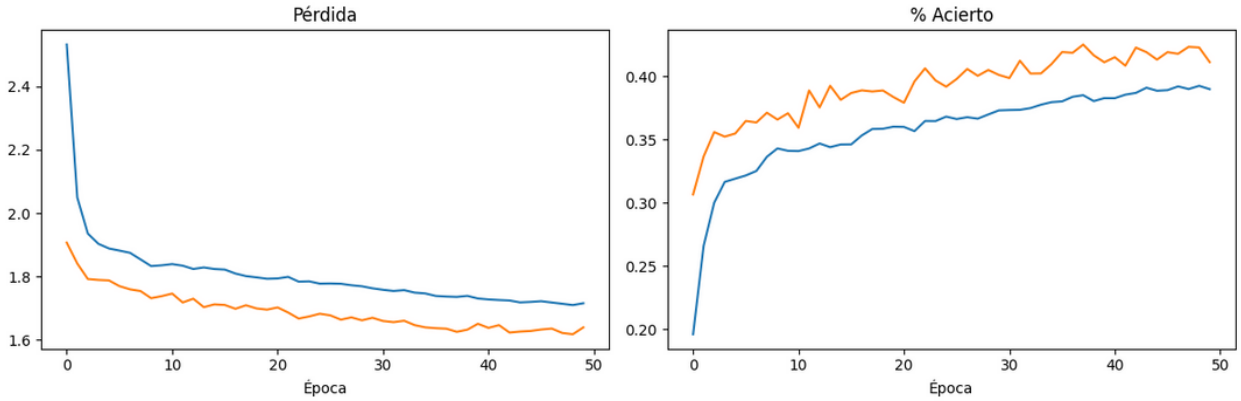
	SGD + Dropout					
	Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*
Relu	3	0,4537	1,5168	0,4924	1,4245	247
Tanh	3	0,4083	1,6801	0,4296	1,6265	244
Sigmoid	1	0,3705	1,7757	0,4058	1,7198	189

1.2. ADAM + Dropout

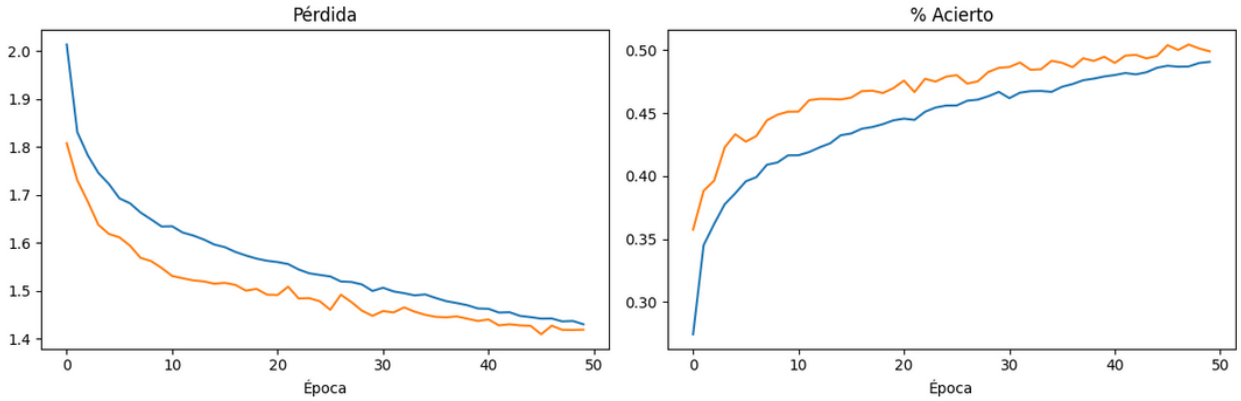
1.2.1. ADAM + ReLu + Dropout (3 capas)



1.2.2. ADAM + Tanh + Dropout (1 capa)



1.2.3. ADAM + Sigmoid + Dropout (1 capa)

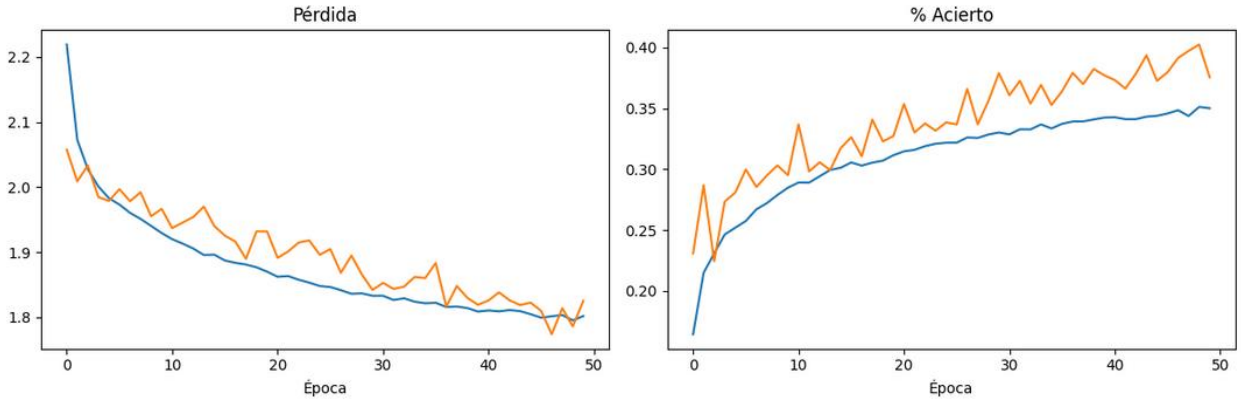


1.2.4. Resultados

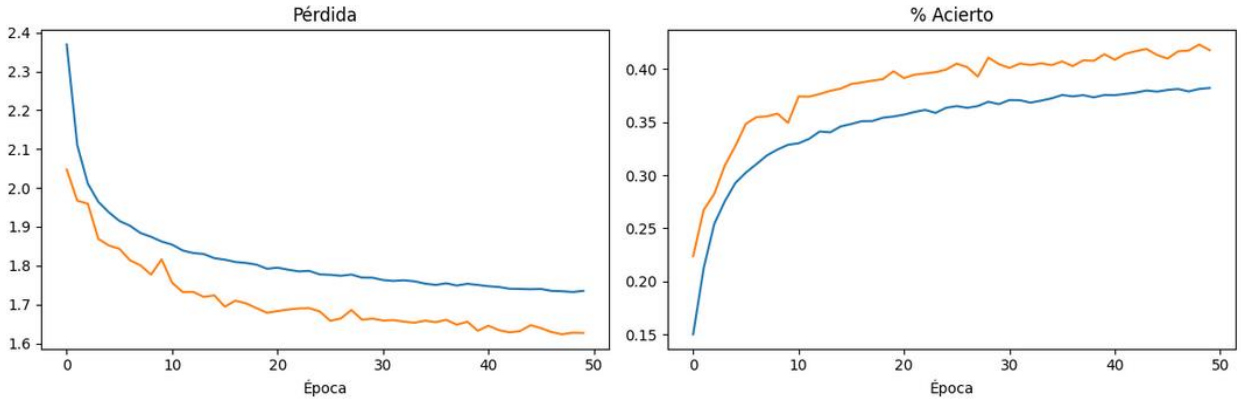
	ADAM + Dropout					
	Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*
Relu	3	0,2766	1,9304	0,2911	1,9537	501
Tanh	1	0,3905	1,7166	0,4114	1,6395	435
Sigmoid	1	0,4919	1,4266	0,4991	1,4188	491

1.3. RMSProp + Dropout

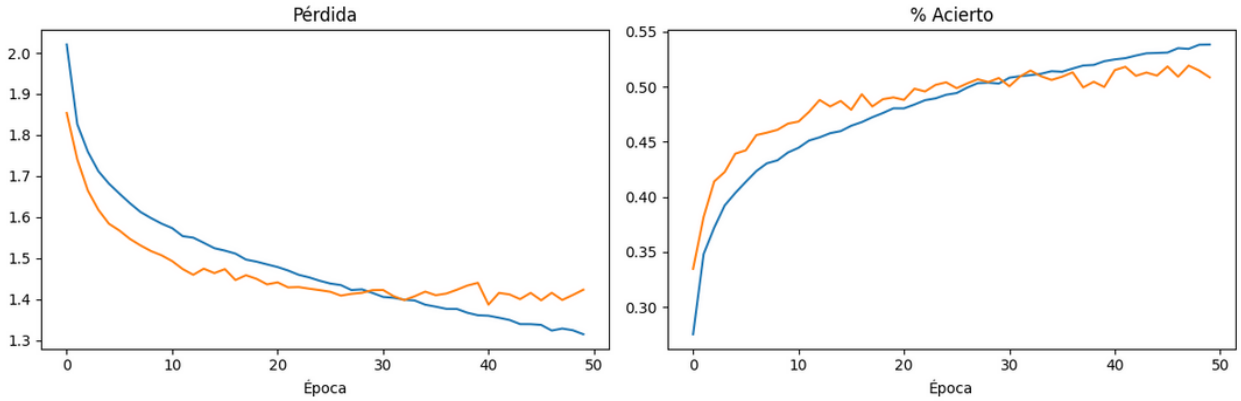
1.3.1. RMSProp + ReLu + Dropout



1.3.2. RMSProp + Tanh + Dropout



1.3.3. RMSProp + Sigmoid + Dropout



1.3.4. Resultados

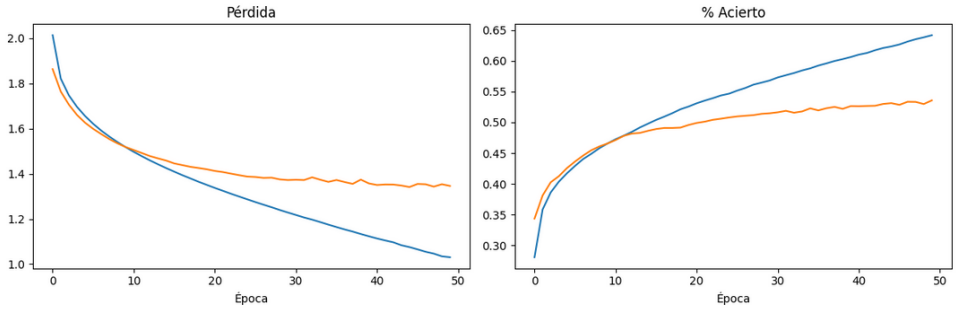
	RMSProp + Dropout					
	Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*
Relu	3	0,3462	1,8060	0,3756	1,8251	450

Tanh	3	0,3835	1,7362	0,4178	1,6266	465
Sigmoid	1	0,5405	1,3153	0,5084	1,4231	422

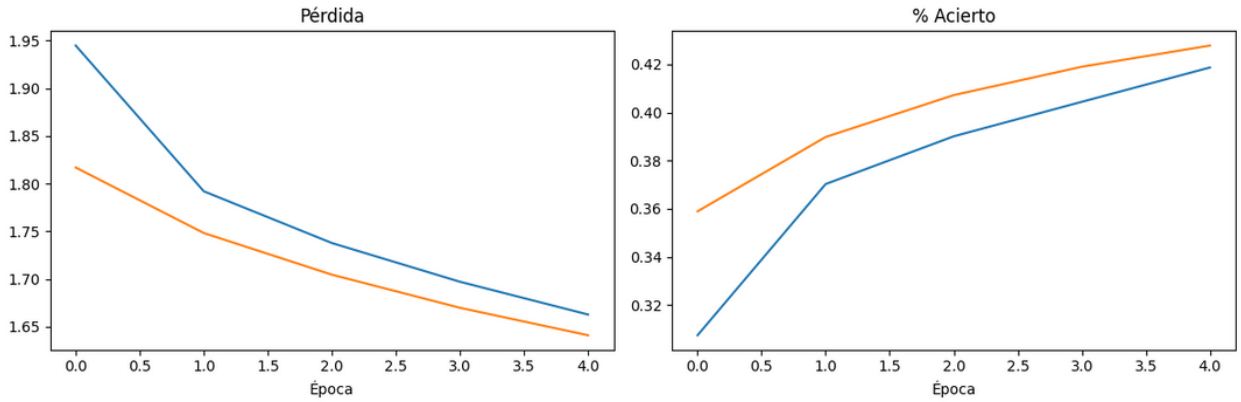
2. Regularización con Earlystopping

2.1. SGD + Earlystopping

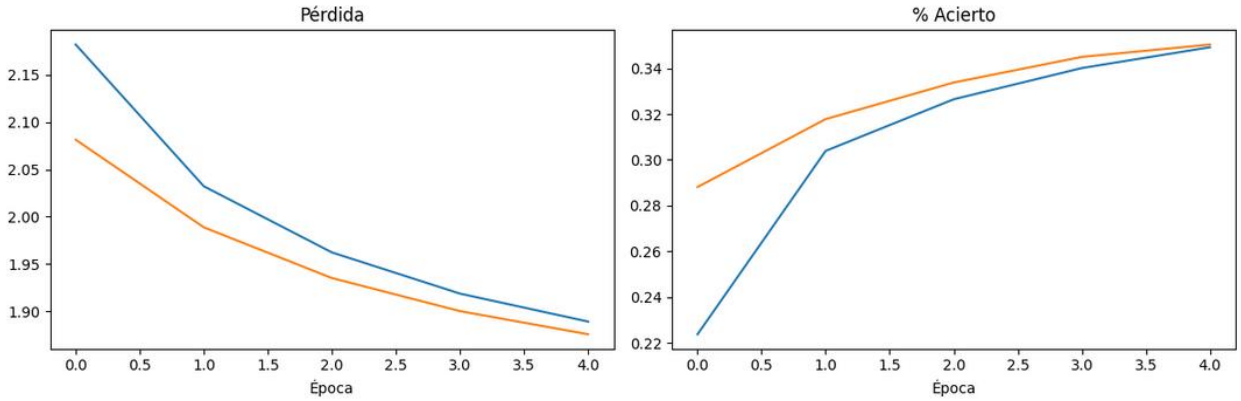
2.1.1. SGD + ReLu + Earlystopping



2.1.2. SGD + Tanh + Earlystopping



2.1.3. SGD + Sigmoid + Earlystopping

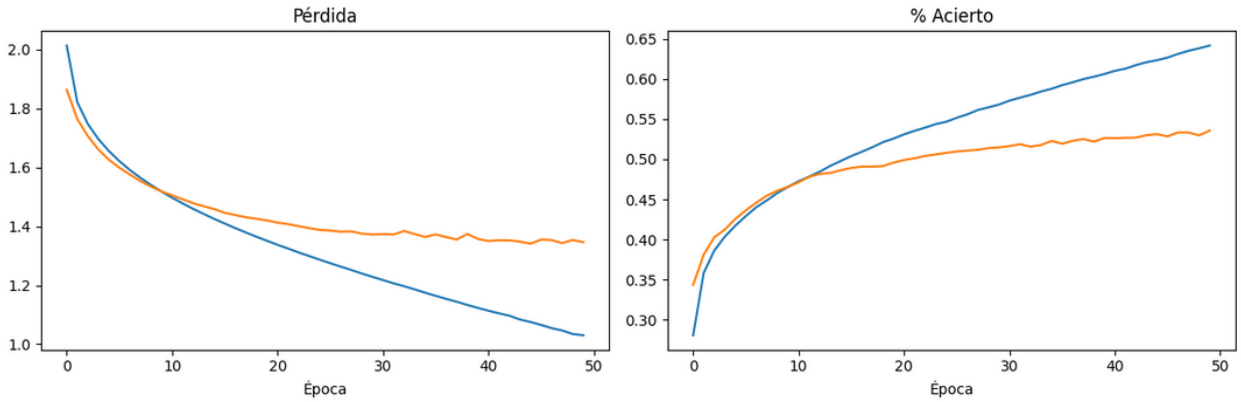


2.1.4. Resultados

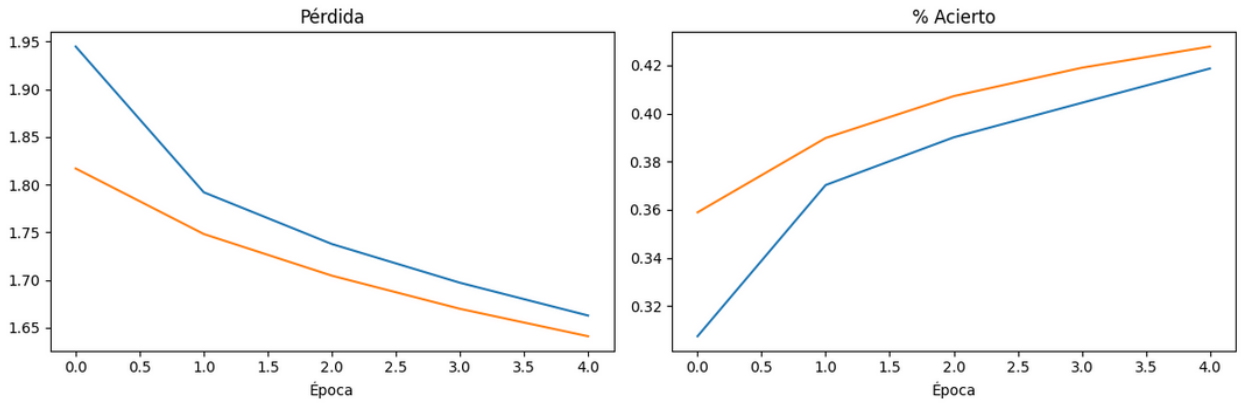
	SGD + Earlystopping						
	Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*	*epoc*
Relu	3	0,6413	1,0331	0,5357	1,3461	230	50
Tanh	3	0,4171	1,6724	0,4278	1,6412	27	5
Sigmoid	1	0,3473	1,8967	0,3504	1,8757	23	5

2.2. Adam + Earlystopping

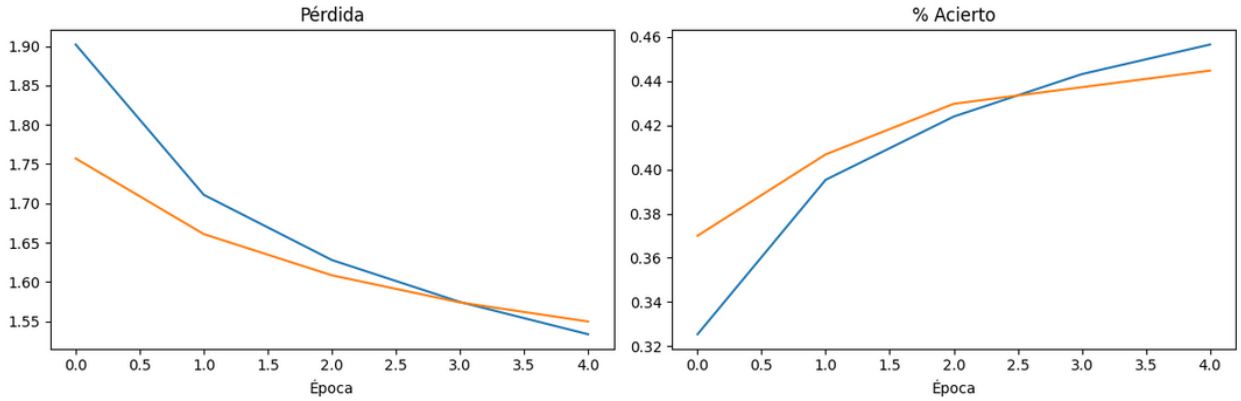
2.2.1. Adam + ReLu + Earlystopping



2.2.2. Adam + Tanh + Earlystopping



2.2.3. Adam + Sigmoid + Earlystopping

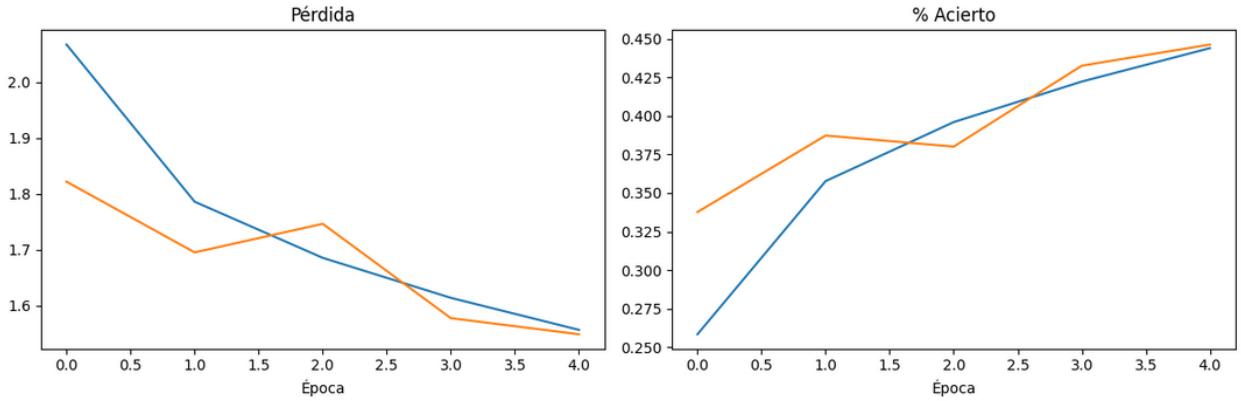


2.2.4. Resultados

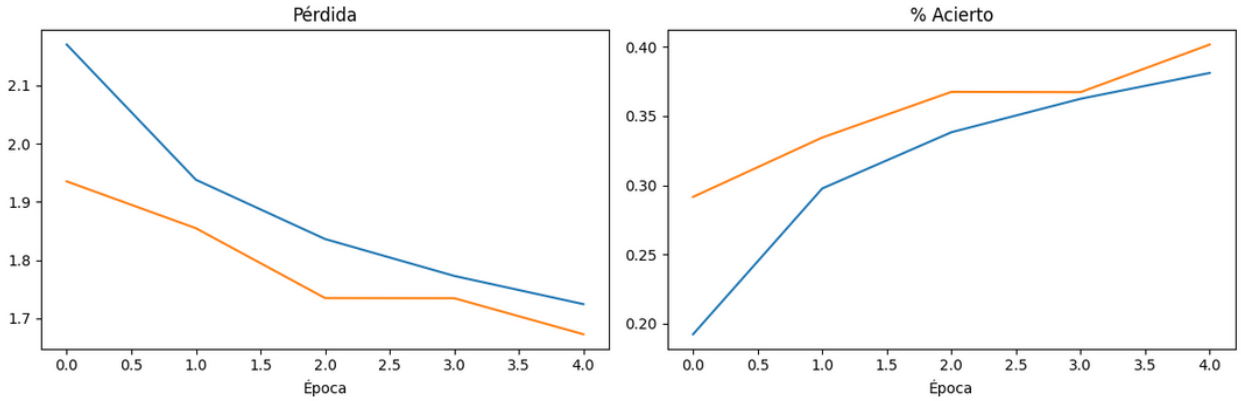
	ADAM + Earlystopping						
	Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*	*epoc*
Relu	3	0,4620	1,4975	0,4621	1,5093	52	5
Tanh	1	0,4096	1,6627	0,4064	1,6590	45	5
Sigmoid	1	0,4522	1,5431	0,4447	1,5496	48	5

2.3. RMSProp + Earlystopping

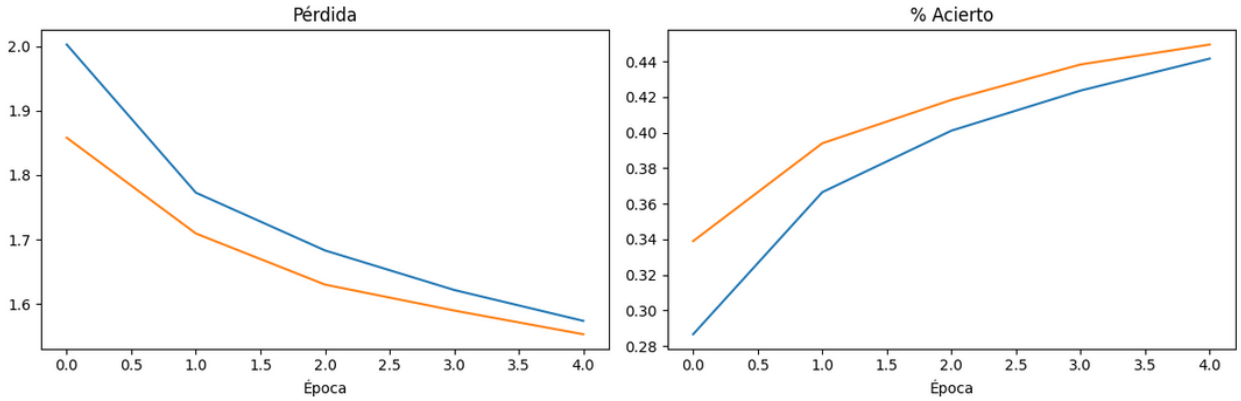
2.3.1. RMSProp + ReLu + Earlystopping



2.3.2. RMSProp + Tanh + Earlystopping



2.3.3. RMSProp + Sigmoid + Earlystopping



2.3.4. Resultados

	RMSProp + Earlystopping						
	Capas	*accuracy*	*loss*	*val_accuracy*	*val_loss*	*tiempo*	*epoc*
Relu	3	0,4364	1,5700	0,4463	1,5485	44	5

Tanh	3	0,3706	1,7445	0,4017	1,6727	48	5
Sigmoid	1	0,4354	1,5872	0,4495	1,5596	44	5