DuocUC® | ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

Deep Learning

Curso: Deep Learning (DLY0100)

Docente: Marco Japke

Integrantes:

- Cristian Contreras
- Alejandro Ferrera
- Diego Gieminiani

Santiago, a 09 de junio de 2024

Índice

● Descripción del problema

En el competitivo mundo de la moda, las empresas buscan constantemente mejorar la experiencia de compra en línea para mantener la satisfacción del cliente y aumentar la eficiencia operativa. Una parte crucial de esta experiencia es la capacidad de clasificar con precisión las prendas de vestir en las categorías adecuadas, lo que facilita una búsqueda más rápida y efectiva por parte de los usuarios. Sin embargo, la clasificación manual es laboriosa y propensa a errores debido a la gran variedad y similitud entre diferentes prendas.

En este proyecto, nos proponemos desarrollar un modelo de Deep Learning que pueda clasificar con precisión imágenes de prendas de vestir utilizando el conjunto de datos Fashion-MNIST. Este conjunto consta de 70,000 imágenes en escala de grises, cada una de 28×28 píxeles, distribuidas en 10 categorías de productos de moda. El objetivo es automatizar y mejorar la clasificación de productos en las plataformas de comercio electrónico de moda, lo cual no sólo optimizará la gestión de inventarios sino que también enriquecerá la experiencia de compra de los usuarios.

Este proyecto beneficiará directamente a las empresas de moda que operan plataformas de comercio electrónico, al mejorar la precisión de la clasificación de productos y, por ende, la eficiencia en la gestión de inventarios y la satisfacción del cliente. Indirectamente, los consumidores también se beneficiarán de una experiencia de compra más fluida y personalizada.

Implementar una solución de Deep Learning para la clasificación automática de prendas puede significar una reducción considerable en el tiempo y costo asociados con la gestión de inventarios, así como un aumento en las tasas de conversión y fidelidad de los clientes debido a una mejor experiencia de usuario. Además, establecerá un precedente tecnológico para el uso de inteligencia artificial en otras áreas del comercio electrónico de moda.

- **Descripción técnicas de la solución**

La clasificación automática de prendas de vestir representa un desafío técnico significativo, debido a las variaciones de patrones en escala de grises que deben ser correctamente interpretadas por un sistema automatizado. Además, las tendencias de moda cambiantes pueden requerir que el sistema se adapte rápidamente a nuevos estilos y categorías de productos.

Para ello, es necesario tener conocimiento y entender el modelo de negocios para proponer de la mejor manera soluciones que aporten a la solución de la problemática.

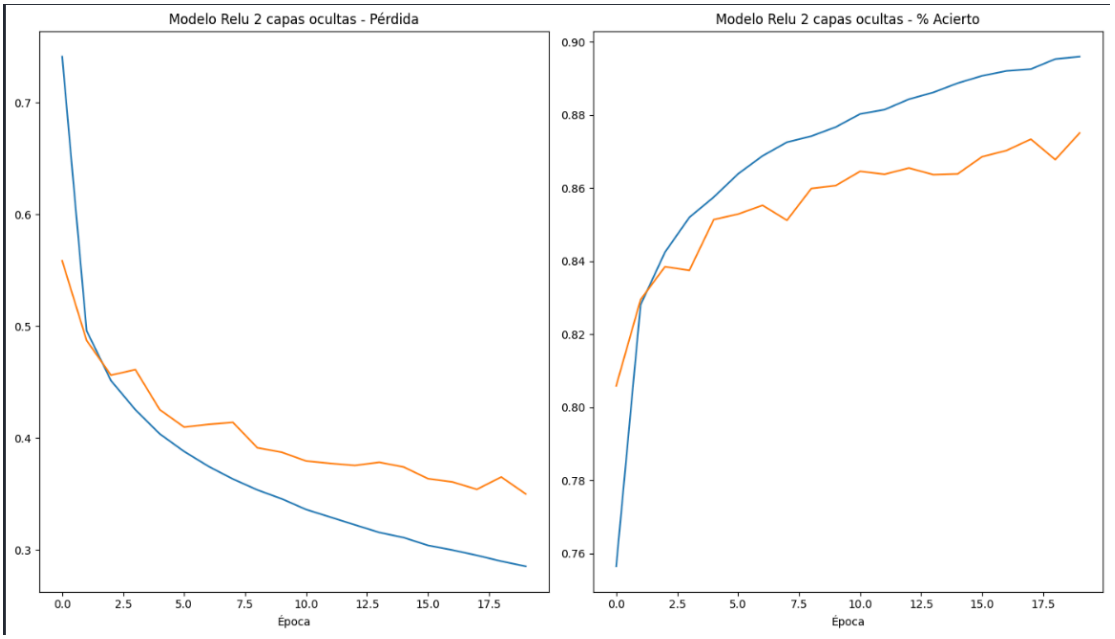
Dentro de las opciones técnicas se realizará un análisis usando deep learning para ello en primer lugar se implementarán modelos con diferentes funciones de activación, luego se estrenarán los modelos con diferentes hiperparametros para la mejor modelo hecho en el punto anterior y luego se usará el mejor modelo con hiperparametros con diferentes optimizadores.

- **Fundamentación técnica de los modelos escogidos**

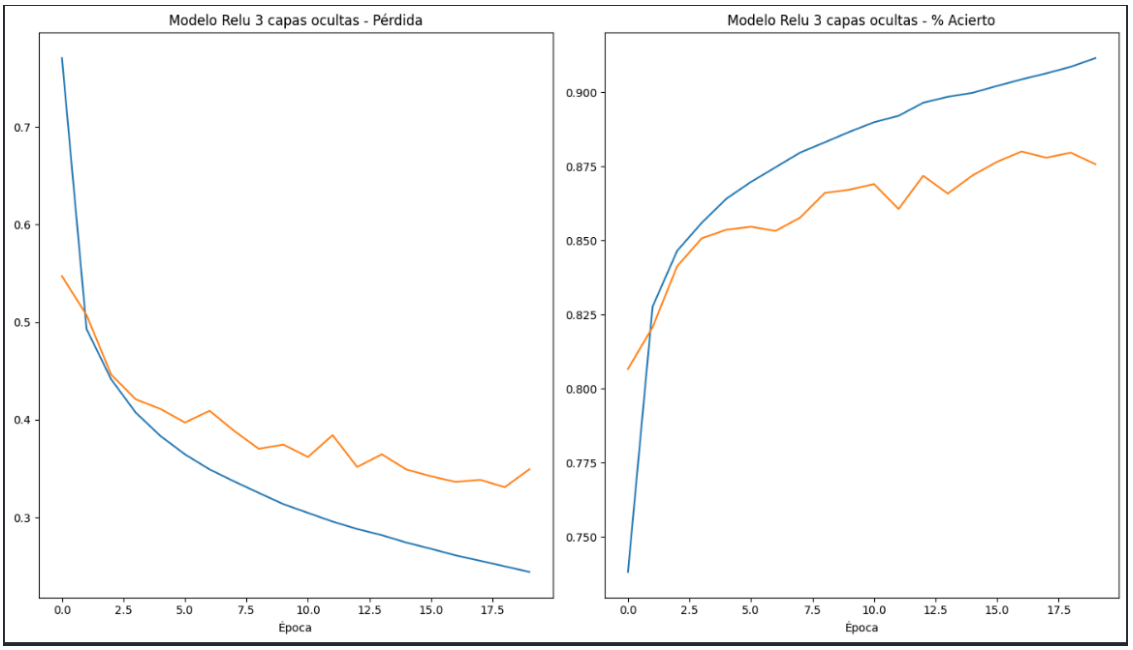
Implementaremos una **Red Feed Forward Fully Connected**, para ello necesitamos realizar labores previas

1. Modificamos el tamaño de la matriz cambiando la estructura del dataset de 60.000 imágenes de 28x28 pixeles a una muestra de 60.000 pero con un vector unidimensional de 784 (28x28).
2. Normalizaremos los valores del vector, para dejarlos entre 0 y 1, ya que esto ayudara a que el modelo no sesgue la información por el valor numerico del vector, dejando todo en el mismo rango numerico, para eso dividiremos los valores en 255 ya que es el maximo valor que puede tomar los datos dentro del vector puesto que la escala de grises va de 0 para el negro a 255 para el blanco.
3. Realizaremos one-hot encoding al dataset con las etiquetas dejándolo como una lista de 10 elementos con nueve 0 y un 1 que indica la etiqueta de la imagen.
4. Asignaremos 60000 ejemplos, 784 Neuronas de entrada y 10 Neuronas de salida.
5. También, asignaremos 4 capas ocultas : h1 con 256 registros, h2 con 128, he con 64 y h4 con 32 registros.
6. Se entrenan 9 modelos con 3 funciones de activación Relu, Tanh y Sigmoide y para cada función se realizarán con diferentes capas ocultas, usando SGD, 20 épocas y un batch size de 32.

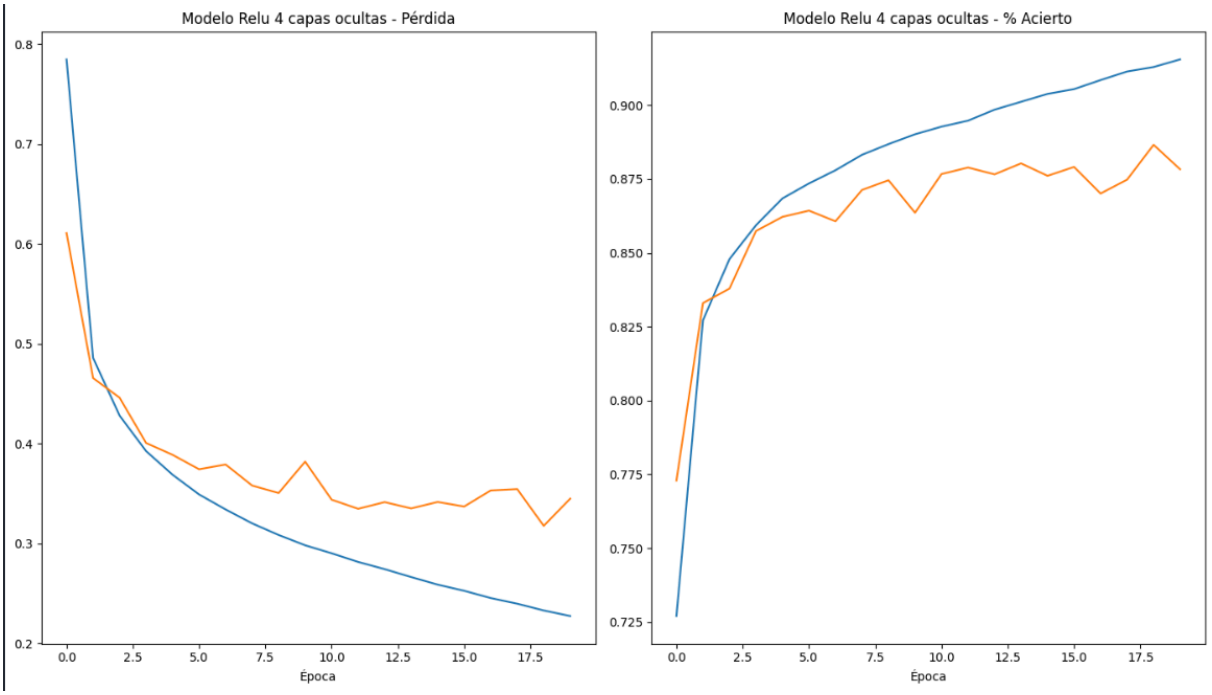
6.a) En primer lugar usaremos la función ReLu con 2 capas ocultas con h2 - h4.



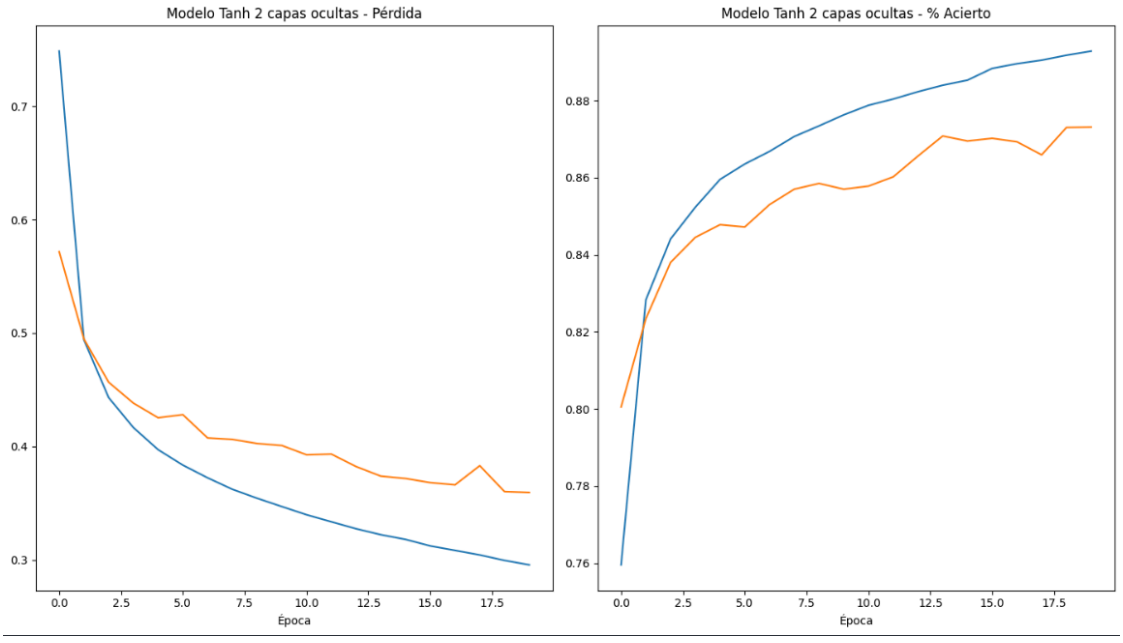
6.b) En segundo lugar usaremos la función ReLu con 3 capas ocultas con h1 - h3 - h4.



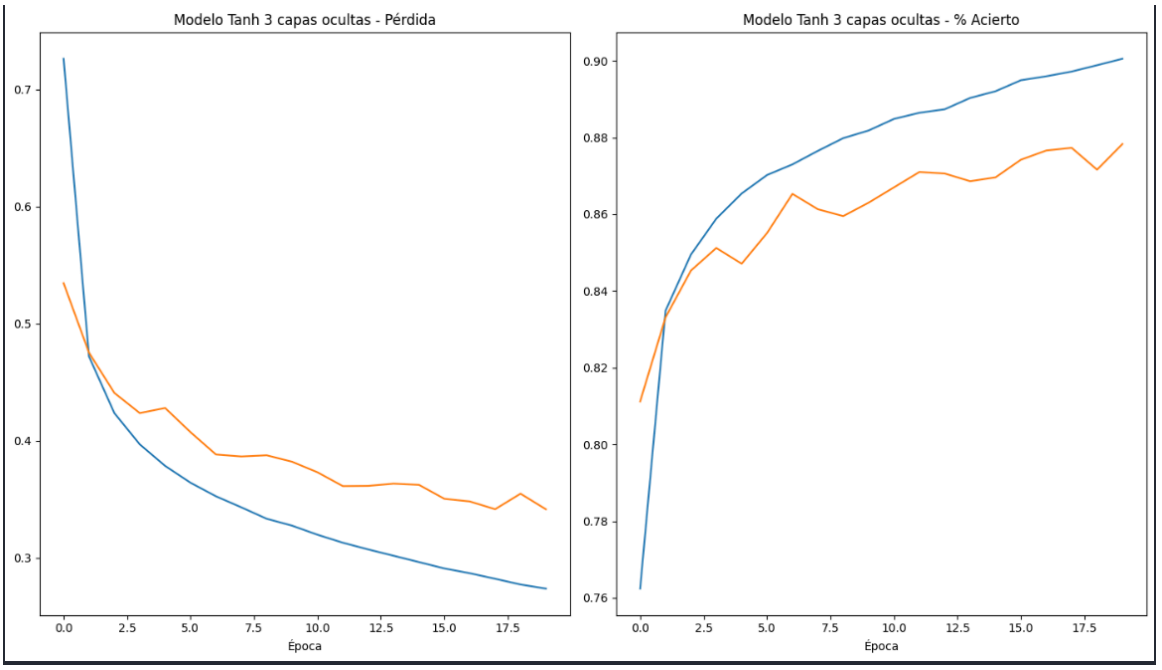
6.c) En tercer lugar usaremos la función ReLu con 4 capas ocultas con h1 - h2 - h3 - h4.



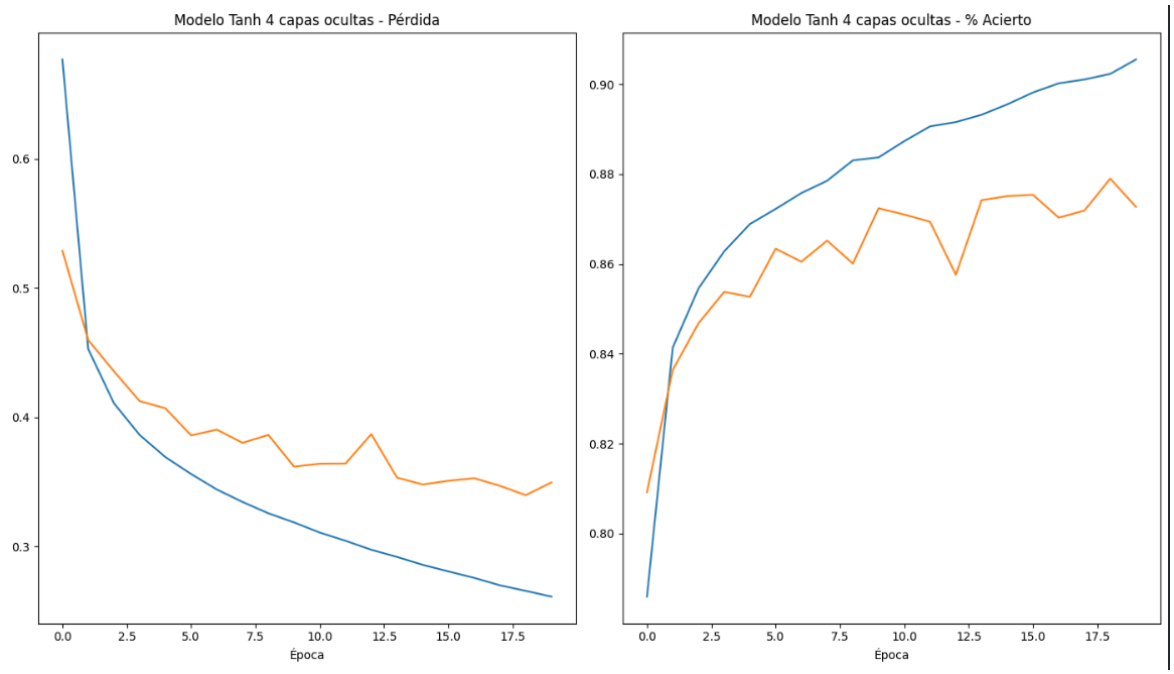
6.d) En cuarto lugar usaremos la función Tangente hiperbólica Tanh con 2 capas con h2 - h4.



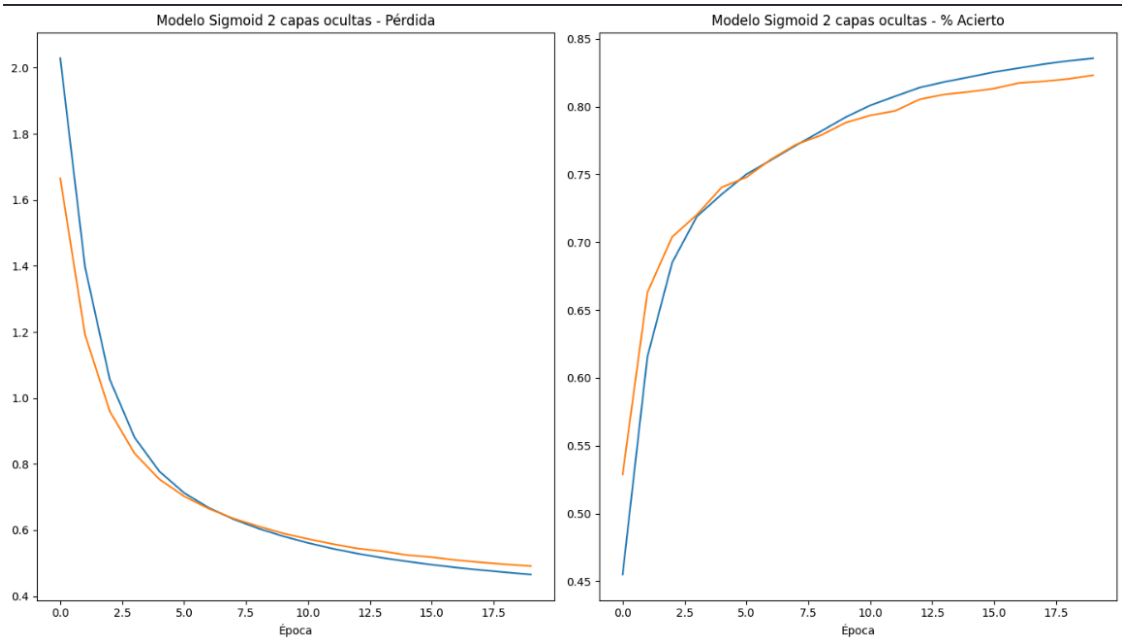
6.e) En quinto lugar usaremos la función Tangente hiperbólica Tanh con h1 - h3 - h4.



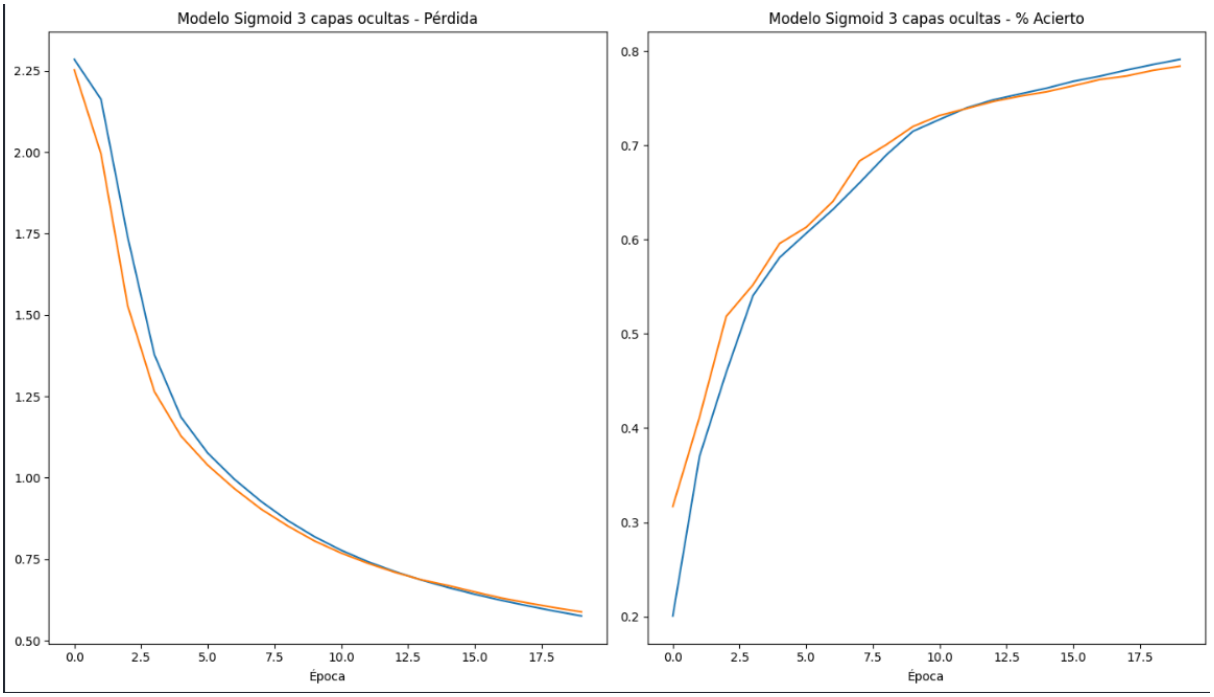
6.f) En sexto lugar usaremos la función Tangente hiperbólica Tanh con con h1 - h2 - h3 - h4.



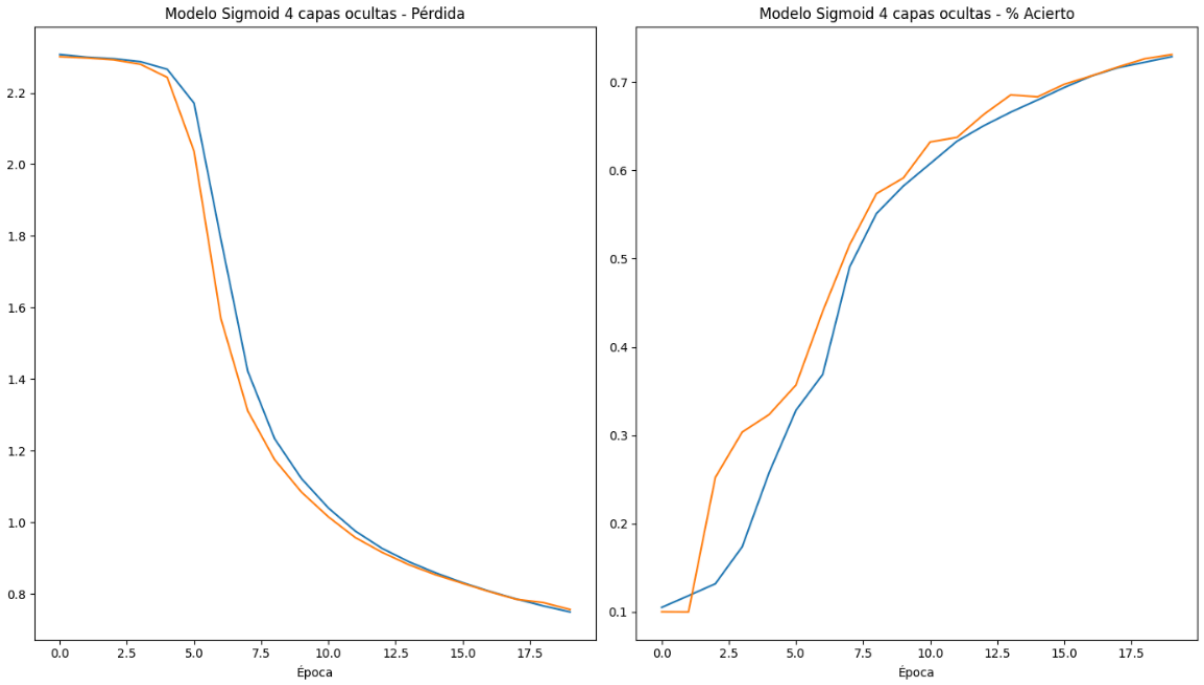
6.g) En séptimo lugar usaremos la función Tangente hiperbólica Tanh con h2 - h4.



6.h) En octavo lugar la función sigmoide con h1 - h3 - h4.



6.i) En noveno lugar la función sigmoide con h1 - h2 - h3 - h4.

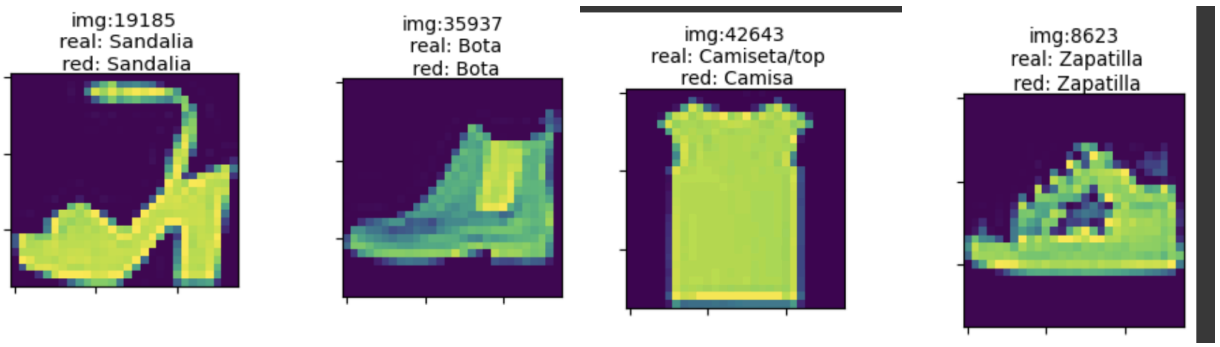


7. Resumiendo

	Nombre	Tiempo de ejecución	Perdida	Accuracy	Val_Accuracy	Val_Loss
2	Modelo Relu 4 capas ocultas	64.837876	0.227586	0.915517	0.8783	0.345055
4	Modelo Tanh 3 capas ocultas	59.431098	0.273793	0.900517	0.8783	0.341514
1	Modelo Relu 3 capas ocultas	63.941934	0.244174	0.911617	0.8758	0.349339
0	Modelo Relu 2 capas ocultas	52.780692	0.285140	0.895983	0.8751	0.349939
3	Modelo Tanh 2 capas ocultas	52.307445	0.295721	0.892817	0.8731	0.359482
5	Modelo Tanh 4 capas ocultas	65.857923	0.261160	0.905500	0.8727	0.349506
6	Modelo Sigmoid 2 capas ocultas	52.039589	0.465559	0.835700	0.8232	0.491043
7	Modelo Sigmoid 3 capas ocultas	59.673061	0.575680	0.791267	0.7841	0.587976
8	Modelo Sigmoid 4 capas ocultas	64.388945	0.750073	0.728850	0.7312	0.756893

Notamos que los mejores modelos son los Relu con 4 y 3 capas y el peor las 3 funciones sigmoides, por ende usaremos la Función ReLu con 4 capas para realizar los ajustes con Hiperparametros.

8. Acá se muestran unos ejemplos de como predijo el modelo, donde notamos que puede diferenciar de buena manera el calzado pero aún tiene problemas con la ropa.



Backpropagation en Fashion MNIST

Se toma una imagen de una prenda de vestir del conjunto de datos y se pasa a través de la red neuronal, capa por capa. Cada capa aplica una transformación lineal (producto punto entre los pesos y las entradas) seguida de una función de activación, como ReLU (Rectified Linear Unit) en las capas ocultas y una función softmax en la capa de salida. Esto produce una salida de la red, que son las probabilidades de que la imagen pertenezca a cada una de las clases de prendas de vestir.

La retropropagación del error se calcula el error entre las probabilidades predichas por la red y las etiquetas reales de las imágenes del conjunto de datos (por ejemplo, si la imagen es un pantalón, la etiqueta correspondiente es "pantalón"). Luego, este error se propaga hacia atrás a través de la red, capa por capa, calculando el gradiente de la función de pérdida (como la entropía cruzada) con respecto a los pesos de las conexiones. Este gradiente indica cómo cambiar los pesos para reducir el error de predicción.

Descenso de gradiente en Fashion MNIST:

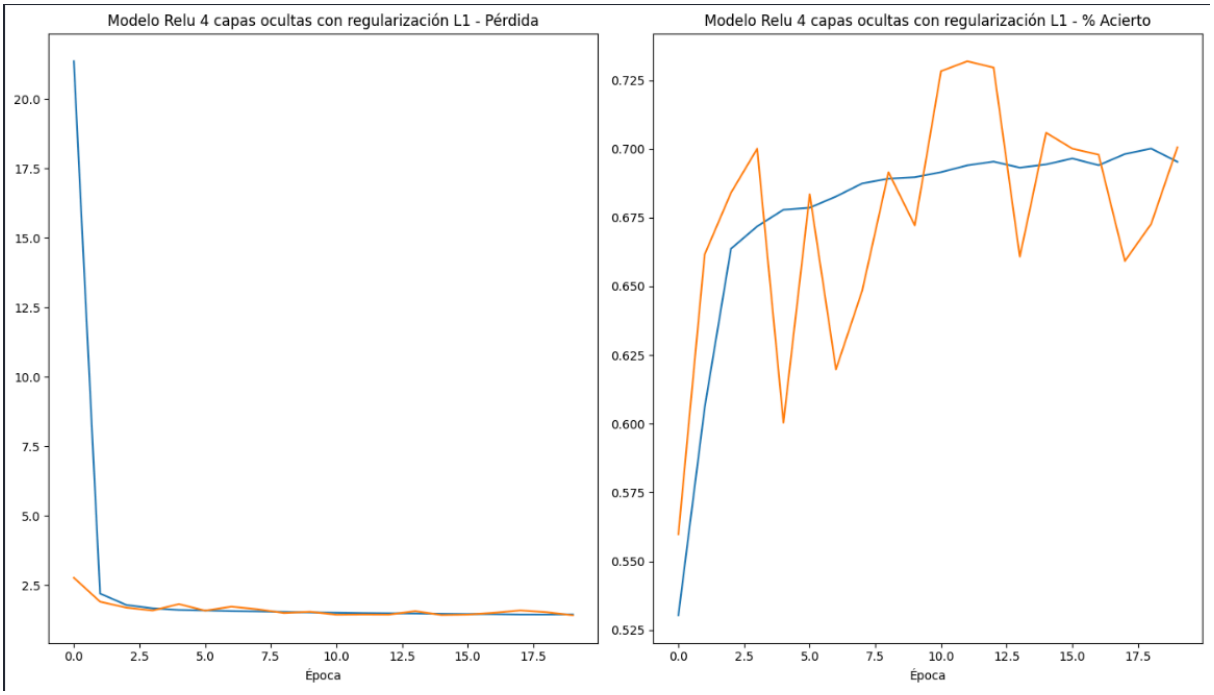
El descenso de gradiente es el proceso de actualizar los pesos de las conexiones de la red neuronal en función del gradiente calculado durante la retropropagación del error. En el contexto de Fashion MNIST, esto implica ajustar los pesos para minimizar la función de pérdida (que mide la discrepancia entre las predicciones de la red y las etiquetas reales).

Descenso de gradiente estocástico (SGD): En cada iteración del entrenamiento, se selecciona aleatoriamente un lote de imágenes del conjunto de datos. Se calcula el error y el gradiente correspondiente para ese lote y se actualizan los pesos de la red en función de este gradiente. Este proceso se repite para múltiples iteraciones hasta que se alcanza un criterio de convergencia, como un número máximo de épocas o una tolerancia de error.
El objetivo final de este proceso es entrenar una red neuronal que pueda clasificar con precisión las prendas de vestir en las imágenes del conjunto de datos Fashion MNIST.

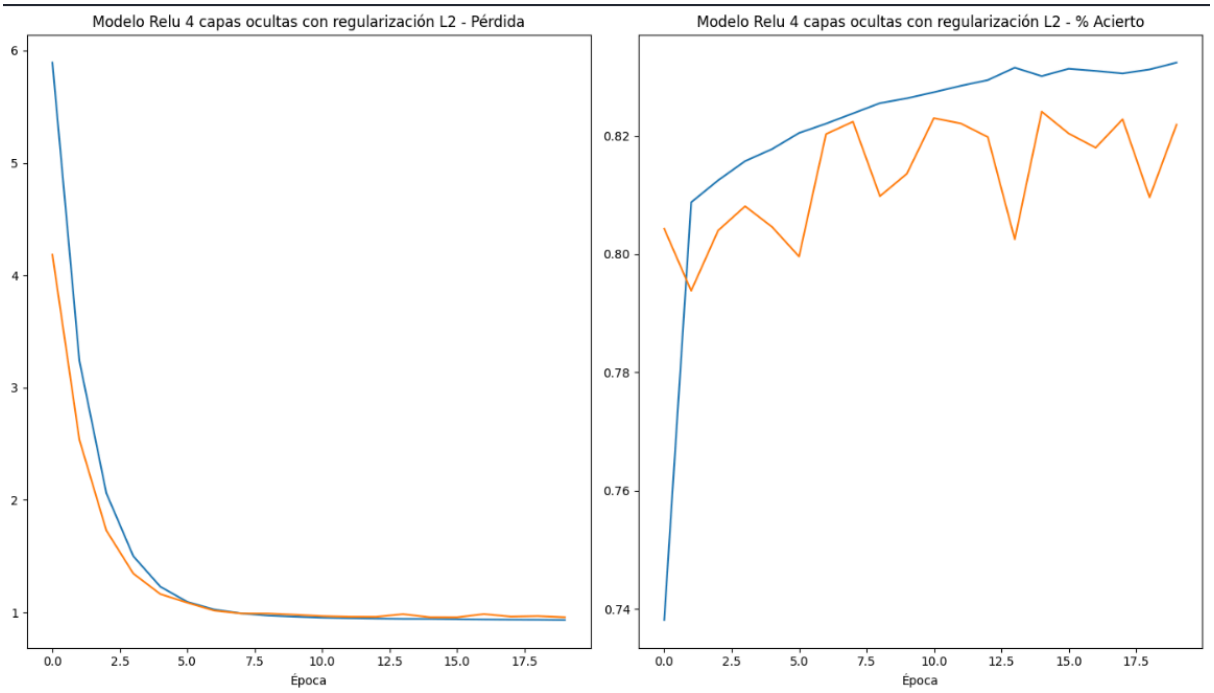
- **Regulación del modelo**

Como nombramos anteriormente usaremos el modelo ReLu 4 capas

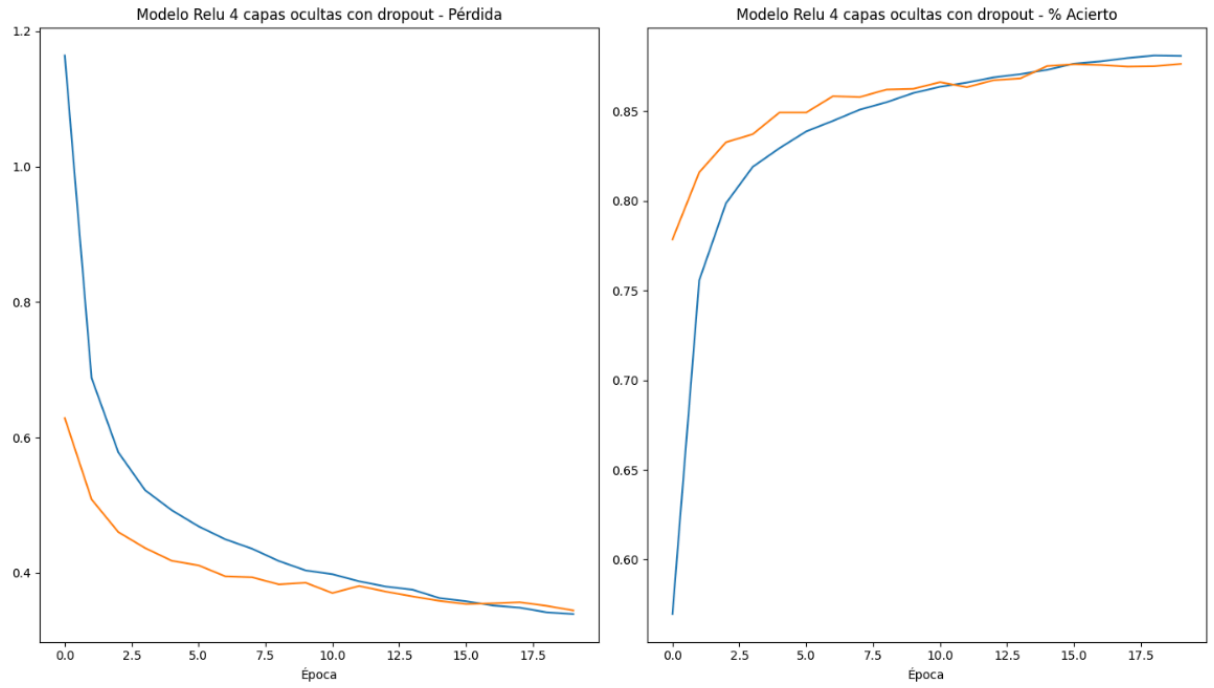
1. Primero usaremos coeficiente L1 de 0.01la cual se utiliza para evitar el sobreajuste (overfitting) y para promover la simplicidad del modelo al penalizar los coeficientes de los pesos de la red neuronal que son demasiado grandes.



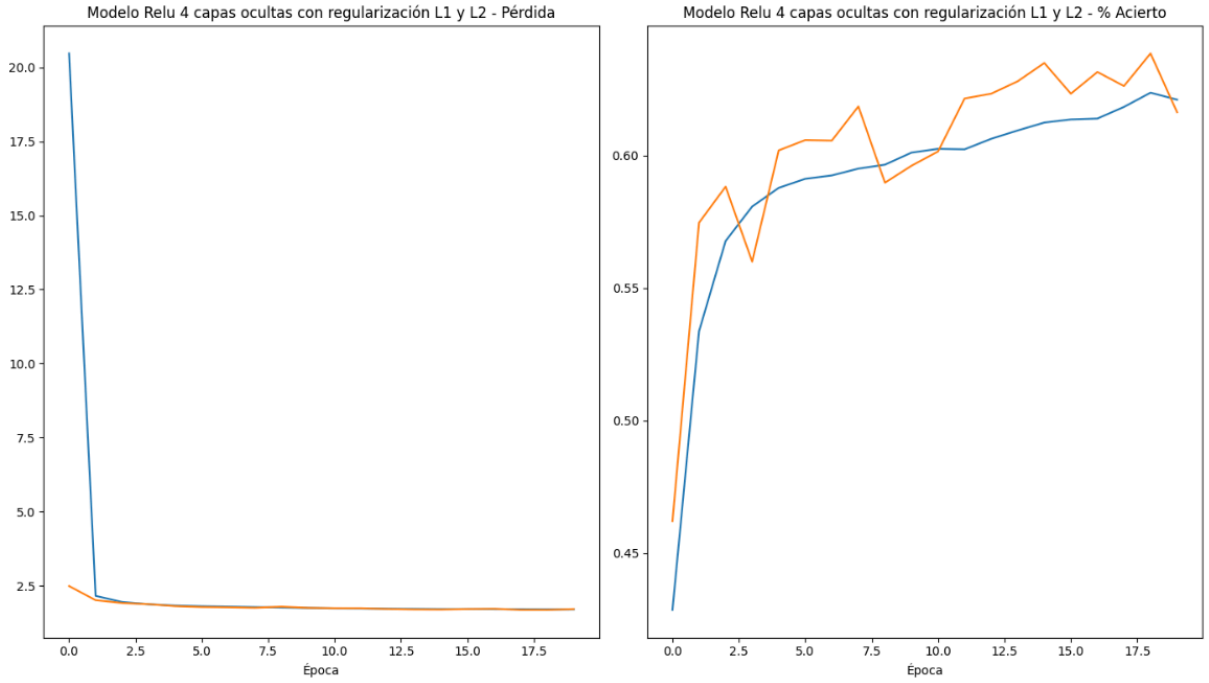
2. Usaremos L2 con coeficiente 0.01 la cual agrega un término a la función de pérdida del modelo que penaliza los pesos de los parámetros de la red en función de su magnitud al cuadrado.



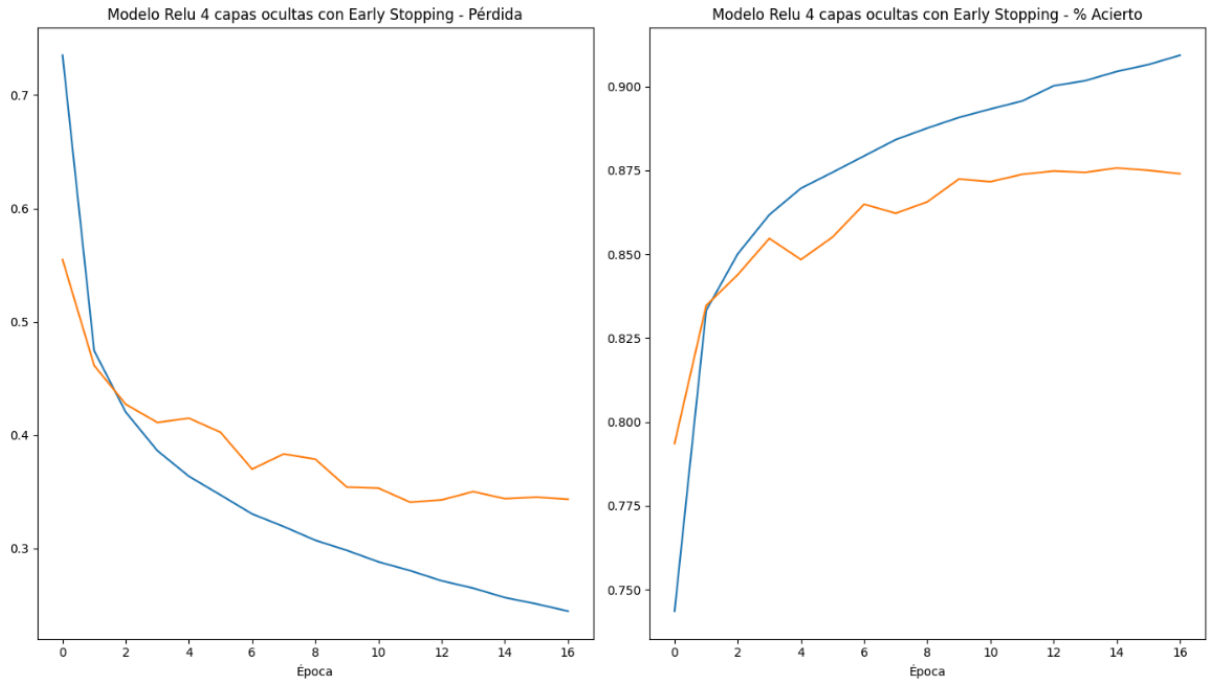
3. Dropout de 0.02 entre capas, la cual consiste en aleatoriamente apagar un cierto porcentaje de neuronas en una capa determinada.



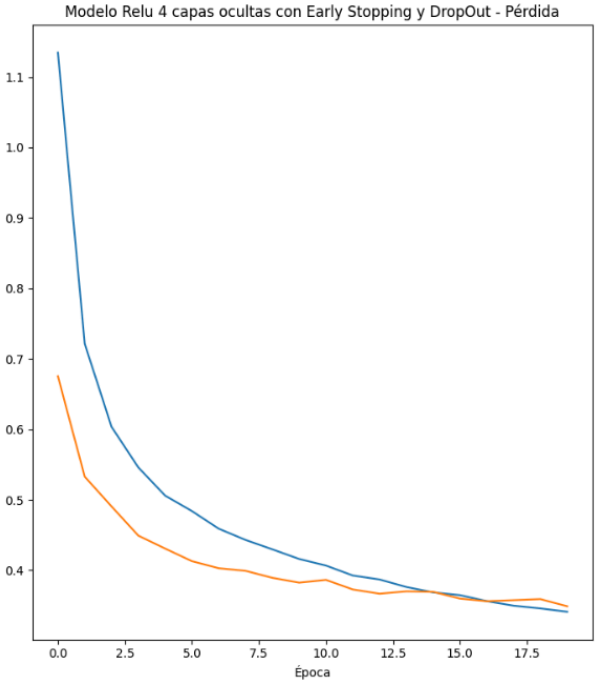
4. El modelo con L1 y L2 ambos con coeficientes de 0.01.

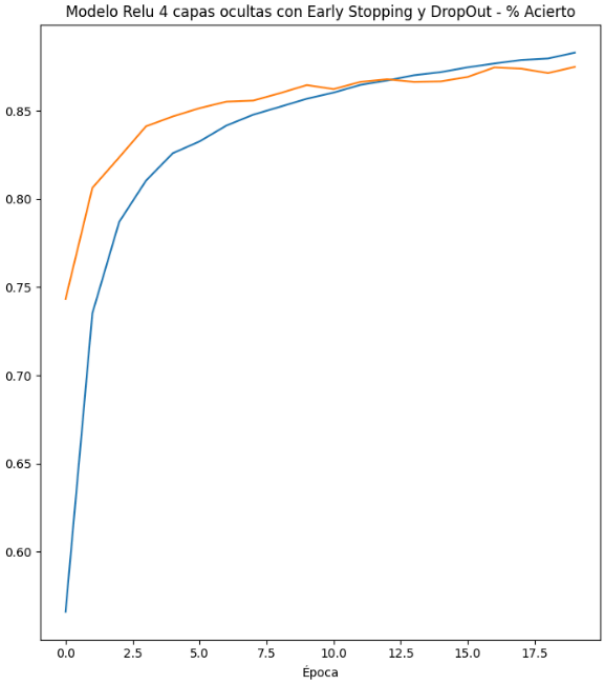


5. Usaremos Early Stopping el cual consiste en monitorear el desempeño del modelo en un conjunto de datos de validación durante el entrenamiento y detener el proceso de entrenamiento cuando el rendimiento del modelo en el conjunto de validación comienza a empeorar, usaremos como monitor 'val_loss' y patience de 5

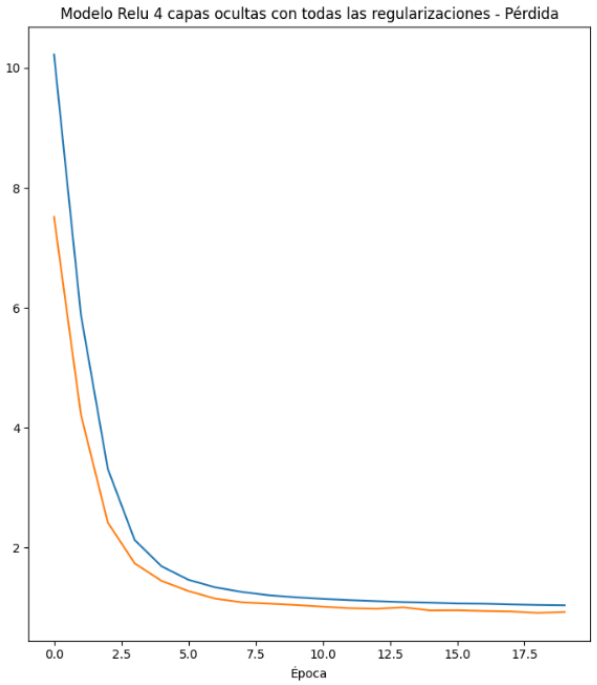


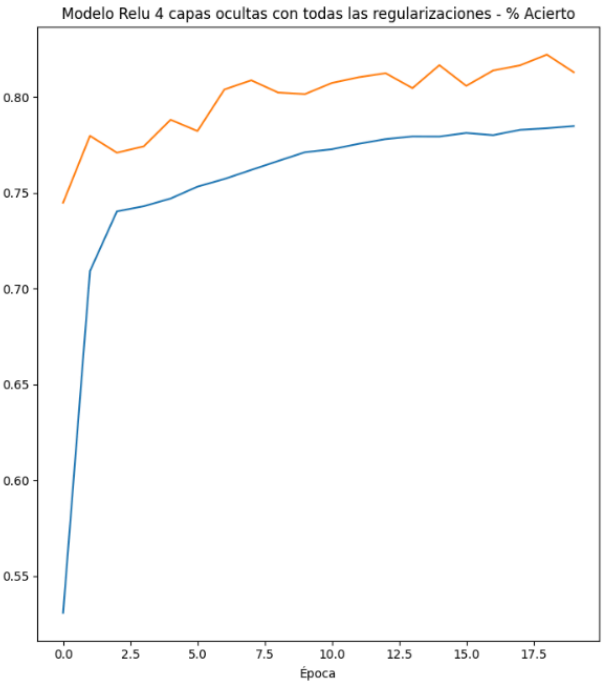
6. Acá usaremos Dropout y Early Stopping





7. Acá usaremos todas las regularizaciones antes usadas con los mismo valores





8. Resumiendo

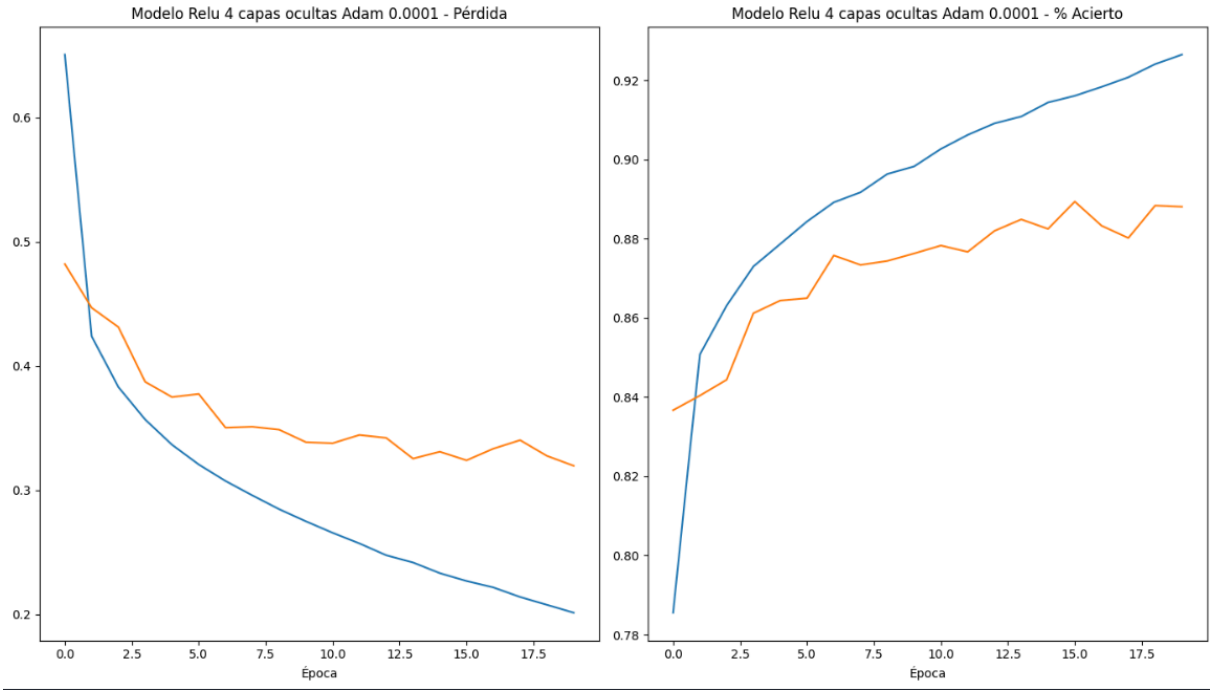
	Nombre	Tiempo de ejecución	Perdida	Accuracy	Val_Accuracy	Val_Loss
2	Modelo Relu 4 capas ocultas	64.837876	0.227586	0.915517	0.8783	0.345055
4	Modelo Tanh 3 capas ocultas	59.431098	0.273793	0.900517	0.8783	0.341514
11	Modelo Relu 4 capas ocultas con dropout	69.670721	0.339347	0.880833	0.8764	0.344645
1	Modelo Relu 3 capas ocultas	63.941934	0.244174	0.911617	0.8758	0.349339
0	Modelo Relu 2 capas ocultas	52.780692	0.285140	0.895983	0.8751	0.349939
14	Modelo Relu 4 capas ocultas con Early Stopping y DropOut	70.874614	0.341163	0.882967	0.8749	0.348947
13	Modelo Relu 4 capas ocultas con Early Stopping	54.999362	0.244772	0.909300	0.8740	0.343446
3	Modelo Tanh 2 capas ocultas	52.307445	0.295721	0.892817	0.8731	0.359482
5	Modelo Tanh 4 capas ocultas	65.857923	0.261160	0.905500	0.8727	0.349506
6	Modelo Sigmoid 2 capas ocultas	52.039589	0.465559	0.835700	0.8232	0.491043
10	Modelo Relu 4 capas ocultas con regularización L2	68.942703	0.932188	0.832383	0.8219	0.956483
15	Modelo Relu 4 capas ocultas con todas las regularizaciones	81.048001	1.035950	0.784900	0.8130	0.924754
7	Modelo Sigmoid 3 capas ocultas	59.673061	0.575680	0.791267	0.7841	0.587976
8	Modelo Sigmoid 4 capas ocultas	64.388945	0.750073	0.728850	0.7312	0.756893
9	Modelo Relu 4 capas ocultas con regularización L1	71.276685	1.450034	0.695317	0.7006	1.417843
12	Modelo Relu 4 capas ocultas con regularización L1 y L2	76.625782	1.703504	0.621150	0.6164	1.712871

Notamos que los mejores resultados son cuando no usamos parámetros.

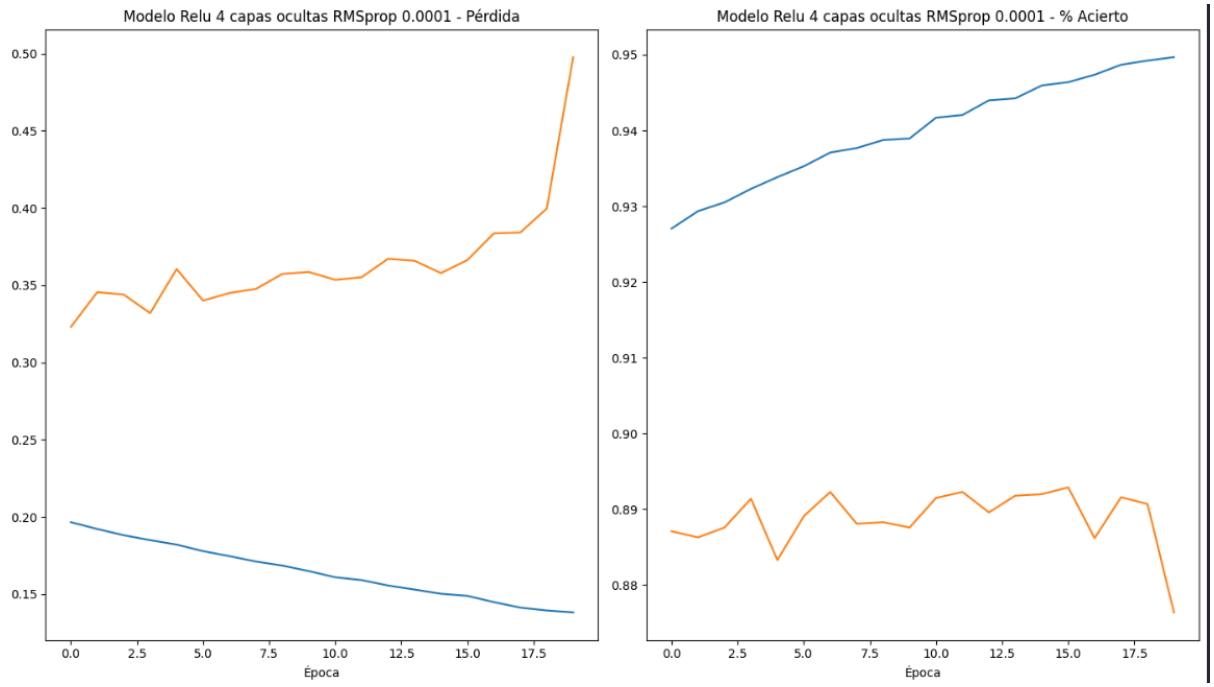
- **Detalle de ajustes de redes**

Usaremos como ajuste del modelo Relu con 4 capas con los siguientes optimizadores

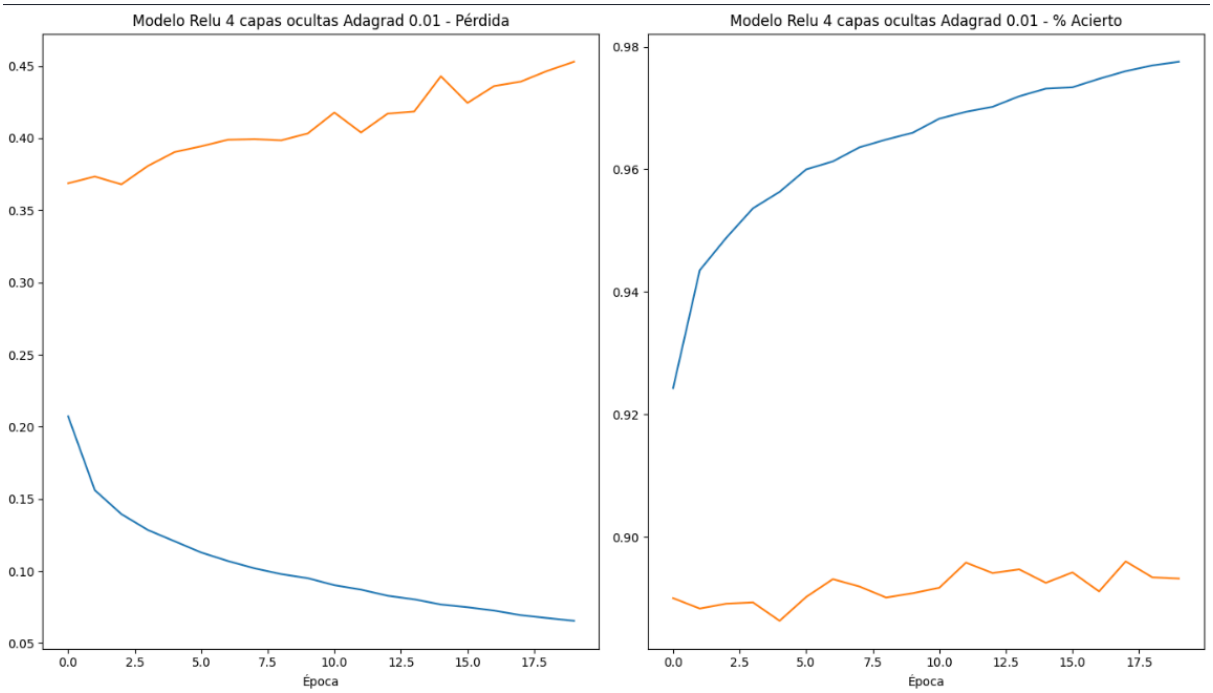
1. Adam con learning_rate=0.0001, beta_1=0.9, beta_2=0.999



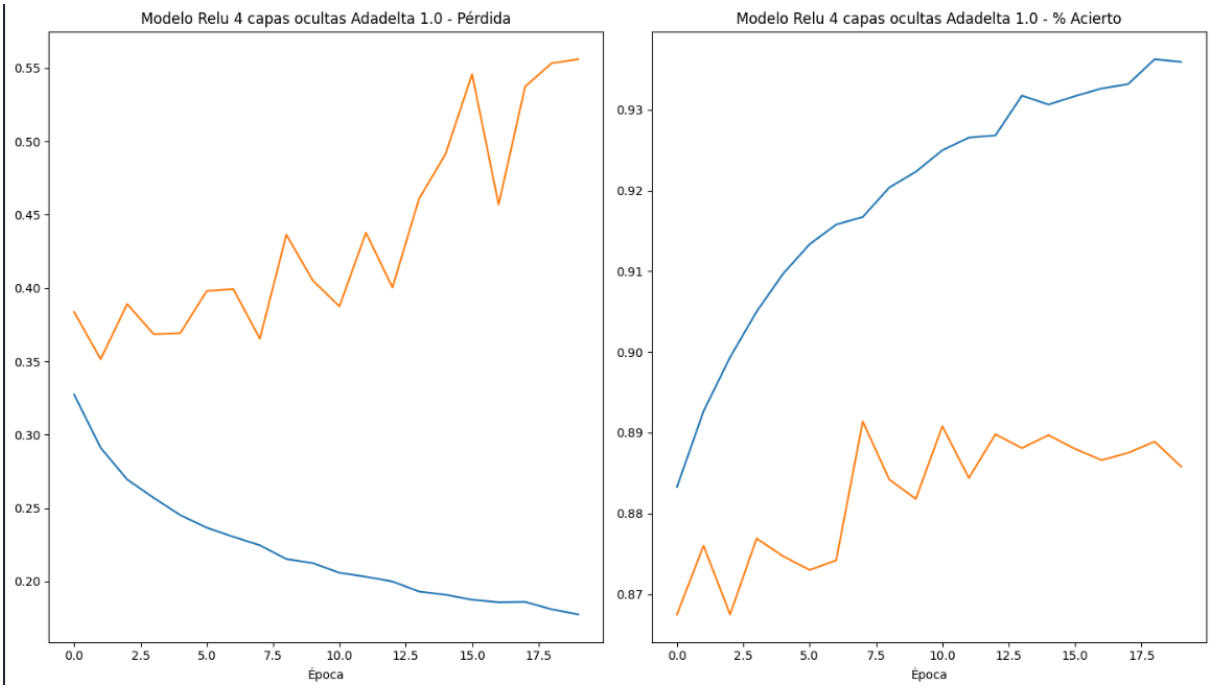
2. RMSprop con learning_rate=0.0001, rho=0.9



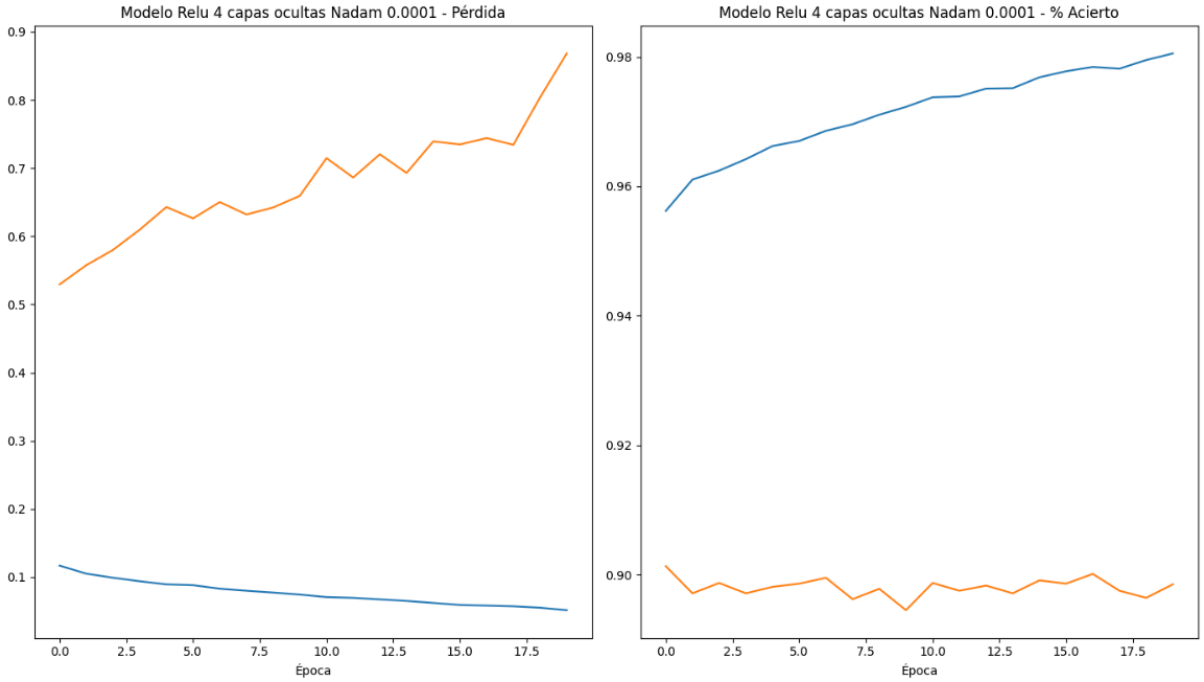
3. Adagrad learning_rate=0.01



4. Adadelta learning_rate=1.0, rho=0.95



5. Nadam learning_rate=0.0001, beta_1=0.9, beta_2=0.999



6. Resumiendo

	Nombre	Tiempo de ejecución	Perdida	Accuracy	Val_Accuracy	Val_Loss
20	Modelo Relu 4 capas ocultas Nadam 0.0001	93.528831	0.051581	0.980517	0.8985	0.868317
18	Modelo Relu 4 capas ocultas Adagrad 0.01	83.014182	0.065539	0.977550	0.8932	0.452978
16	Modelo Relu 4 capas ocultas Adam 0.0001	90.817668	0.201232	0.926517	0.8881	0.319545
19	Modelo Relu 4 capas ocultas Adadelta 1.0	95.813186	0.177562	0.935933	0.8858	0.555951
2	Modelo Relu 4 capas ocultas	64.837876	0.227586	0.915517	0.8783	0.345055
4	Modelo Tanh 3 capas ocultas	59.431098	0.273793	0.900517	0.8783	0.341514
17	Modelo Relu 4 capas ocultas RMSprop 0.0001	82.564986	0.138265	0.949683	0.8764	0.497750
11	Modelo Relu 4 capas ocultas con dropout	69.670721	0.339347	0.880833	0.8764	0.344645
1	Modelo Relu 3 capas ocultas	63.941934	0.244174	0.911617	0.8758	0.349339
0	Modelo Relu 2 capas ocultas	52.780692	0.285140	0.895983	0.8751	0.349939
14	Modelo Relu 4 capas ocultas con Early Stopping y DropOut	70.874614	0.341163	0.882967	0.8749	0.348947
13	Modelo Relu 4 capas ocultas con Early Stopping	54.999362	0.244772	0.909300	0.8740	0.343446
3	Modelo Tanh 2 capas ocultas	52.307445	0.295721	0.892817	0.8731	0.359482
5	Modelo Tanh 4 capas ocultas	65.857923	0.261160	0.905500	0.8727	0.349506
6	Modelo Sigmoid 2 capas ocultas	52.039589	0.465559	0.835700	0.8232	0.491043
10	Modelo Relu 4 capas ocultas con regularización L2	68.942703	0.932188	0.832383	0.8219	0.956483
15	Modelo Relu 4 capas ocultas con todas las regularizaciones	81.048001	1.035950	0.784900	0.8130	0.924754
7	Modelo Sigmoid 3 capas ocultas	59.673061	0.575680	0.791267	0.7841	0.587976
8	Modelo Sigmoid 4 capas ocultas	64.388945	0.750073	0.728850	0.7312	0.756893
9	Modelo Relu 4 capas ocultas con regularización L1	71.276685	1.450034	0.695317	0.7006	1.417843
12	Modelo Relu 4 capas ocultas con regularización L1 y L2	76.625782	1.703504	0.621150	0.6164	1.712871

● Justificación de la solución

Notamos que los mejores modelos son con los automatizadores Nadam learning rate de 0.001y beta_1=0.9, beta_2=0.999 por ende escogeremos este

● Conclusiones con respecto al trabajo realizado

A partir de los datos proporcionados, podemos observar varios aspectos clave en el rendimiento de los diferentes modelos de redes neuronales probados para una tarea de clasificación usando deep learning.

1. **Mejor Rendimiento General**: El modelo 'Modelo Relu 4 capas ocultas con dropout Adam 0.001' ha logrado el mejor equilibrio en términos de precisión (Accuracy) y pérdida de validación (Val_Loss), con una precisión de aproximadamente 0.896 y una pérdida de 0.335 usando menos tiempo. Esto indica que la combinación de la función de activación ReLU, cuatro capas ocultas, dropout y el optimizador Adam parece ser la más efectiva entre las probadas.

2. **Impacto del Dropout y Optimizadores**: Los modelos con dropout generalmente muestran un mejor equilibrio entre la precisión y la pérdida de validación, lo que sugiere que el dropout ayuda a mitigar el sobreajuste. Además, el uso de diferentes optimizadores como Nadam, Adam, y Adadelta con tasas de aprendizaje similares también influye significativamente en el rendimiento, siendo Nadam el más efectivo en este caso.

3. **Influencia de la Regularización**: Los modelos con regularización L1 y L2 tienden a tener una mayor pérdida en la validación, lo que podría indicar que la cantidad de regularización aplicada puede haber sido demasiado restrictiva, afectando negativamente el aprendizaje del modelo.
4. **Comparación entre Funciones de Activación**: Los modelos que utilizan la función de activación ReLU tienden a superar a aquellos con Tanh y Sigmoid

en la mayoría de los casos, mostrando mejor precisión y menor pérdida en la validación.

Para la tarea específica de clasificación, la configuración del modelo utilizando ReLU con cuatro capas ocultas, dropout y el optimizador Nadam provee un rendimiento robusto y eficiente. Sin embargo, es crucial ajustar y experimentar con diferentes tasas de dropout y técnicas de regularización para optimizar aún más el rendimiento y evitar el sobreajuste. Estos resultados ofrecen una base sólida para futuras iteraciones y pruebas con el fin de refinar el modelo para aplicaciones específicas.

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

Para mejorar el rendimiento del modelo de clasificación de imágenes, se recomienda explorar arquitecturas de redes neuronales más especializadas. Una primera mejora sería implementar una Red Neuronal Convolucional (CNN), reconocida por su efectividad en tareas de visión por computadora gracias a su habilidad para detectar patrones espaciales y jerárquicos en imágenes. Podemos comenzar con una estructura básica que incluya capas convolucionales con filtros de varios tamaños, como 3x3 y 5x5, seguidas de capas de pooling y densas, optimizando la captura de características a distintas escalas.

Además, la transferencia de aprendizaje ofrece un gran potencial. Utilizando modelos preentrenados en grandes datasets, como VGG o ResNet, y realizando un ajuste fino de las últimas capas, podemos adaptar eficazmente el modelo a las categorías específicas del conjunto de datos Fashion-MNIST. Esto no solo podría reducir significativamente el tiempo de entrenamiento, sino también mejorar la precisión del modelo.

Otra estrategia es el aumento de datos, que consiste en expandir el conjunto de entrenamiento mediante transformaciones como rotaciones, escalados y volteos horizontales de imágenes. Esto se puede gestionar fácilmente con herramientas disponibles en librerías como TensorFlow o PyTorch, mejorando la generalización del modelo y ayudando a prevenir el sobreajuste.

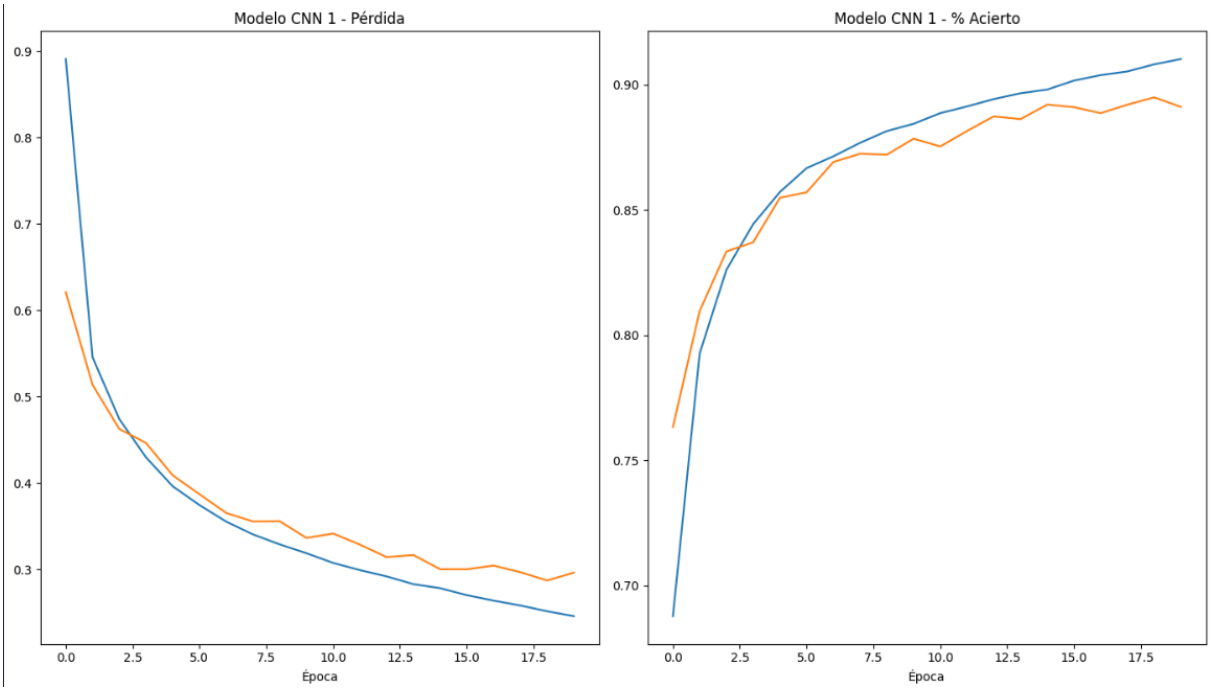
Finalmente, la optimización hiperparamétrica automatizada puede ser crucial. Herramientas como Hyperopt o Optuna permiten buscar de manera eficiente la mejor configuración de los parámetros del modelo, incluyendo la tasa de aprendizaje, el número de capas y unidades por capa, y la configuración de dropout. Esto optimiza el rendimiento del modelo, eliminando la necesidad de ajustes manuales prolongados y a menudo imprecisos.

Al combinar estas técnicas avanzadas, el proyecto no sólo elevará su capacidad actual sino que también establecerá una base sólida para futuras investigaciones y aplicaciones en la clasificación de imágenes con deep learning.

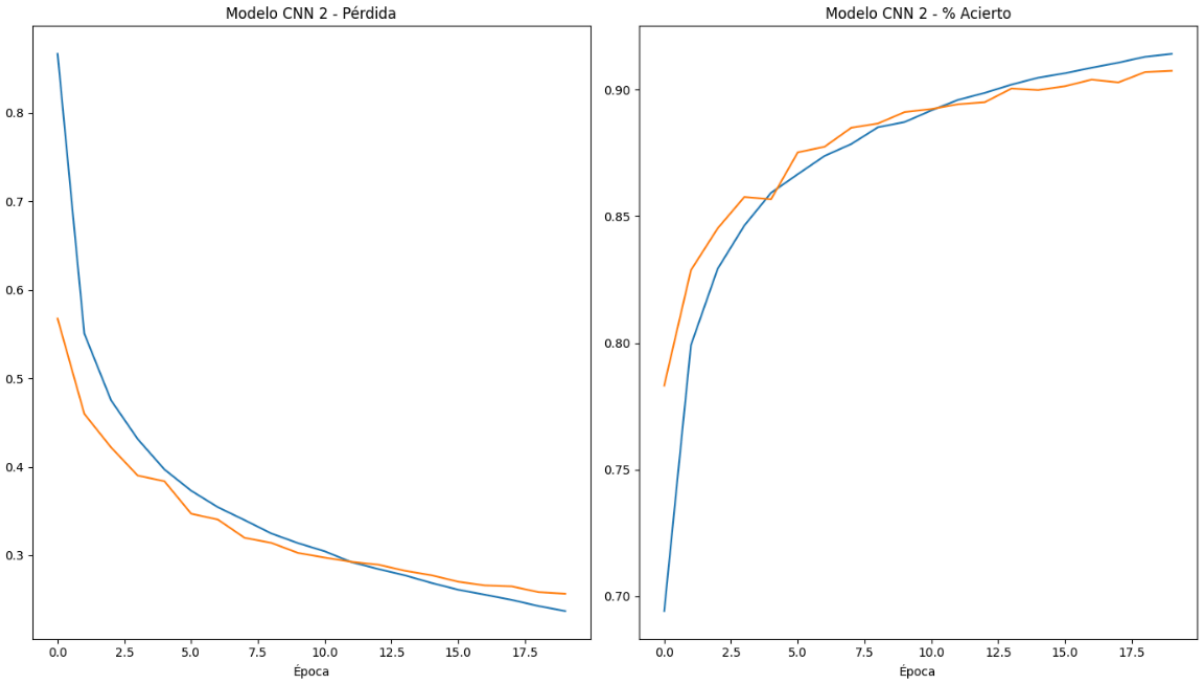
● Parte 5. Redes Convolucionales

A continuación crearemos 3 modelos de redes neuronales convolucionales usando los hiperparametros mencionados anteriormente Adam(learning_rate=0.0001, beta_1=0.9, beta_2=0.999), los cuales tiene los siguientes modelos

a) El modelo model1 es una red neuronal convolucional que consta de tres capas convolucionales: la primera con 32 filtros de tamaño 3x3 y activación ReLU, seguida de una capa de agrupación máxima con ventana de 2x2; la segunda capa convolucional tiene 64 filtros de tamaño 3x3 con activación ReLU, seguida nuevamente de una capa de agrupación máxima con ventana de 2x2; finalmente, la tercera capa convolucional tiene 64 filtros de tamaño 3x3 con activación ReLU. Después de las capas convolucionales, hay una capa Flatten para convertir los mapas de características 2D en un vector 1D, seguida de una capa densa con 128 unidades y activación ReLU, otra capa densa con 64 unidades y activación ReLU, y finalmente una capa de salida con 10 unidades y activación softmax para clasificar imágenes en 10 categorías diferentes.

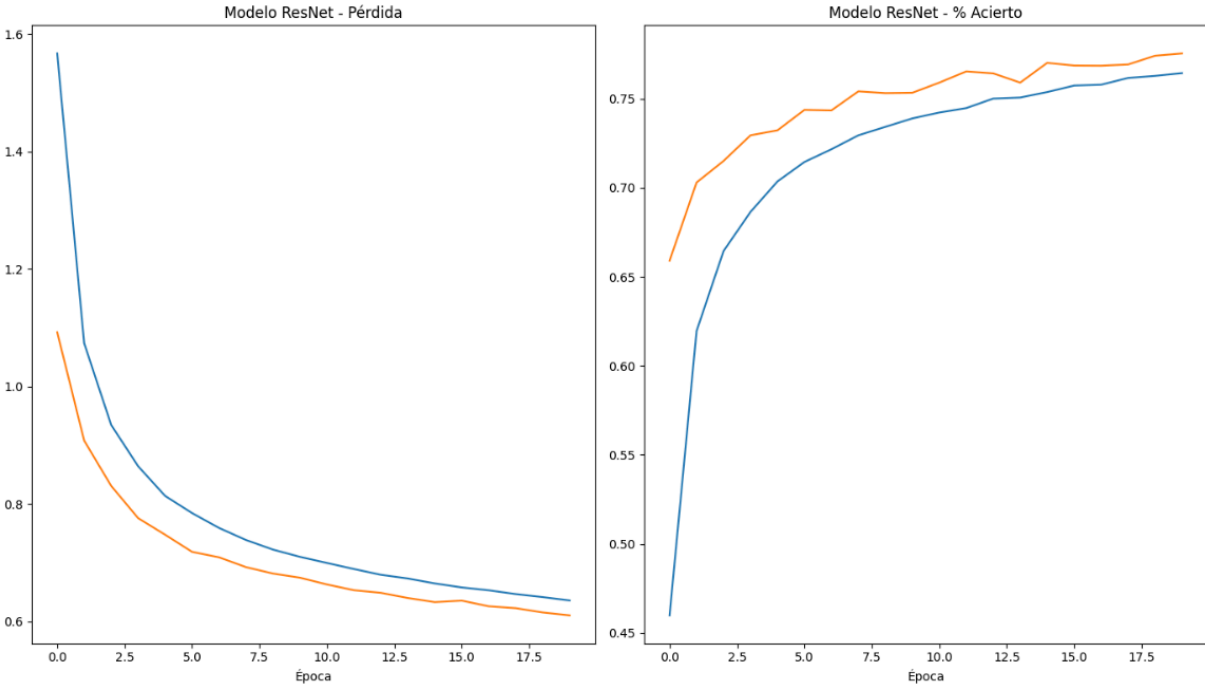


b) El modelo model2 es una red neuronal convolucional compuesta por tres capas convolucionales: la primera con 64 filtros de tamaño 3x3 y activación ReLU, seguida de una capa de agrupación máxima con ventana de 2x2; la segunda capa convolucional tiene 128 filtros de tamaño 3x3 con activación ReLU, seguida nuevamente de una capa de agrupación máxima con ventana de 2x2; finalmente, la tercera capa convolucional tiene 128 filtros de tamaño 3x3 con activación ReLU. Después de las capas convolucionales, hay una capa Flatten para convertir los mapas de características 2D en un vector 1D, seguida de una capa densa con 128 unidades y activación ReLU. Además, se incluye una capa de dropout con una tasa de 0.5 para regularizar el modelo y prevenir el sobreajuste. La capa de salida consta de 10 unidades con activación softmax para clasificar imágenes en 10 categorías diferentes.



c) Model3 usando resnet, primero está la preparación de Datos donde las imágenes de entrenamiento y prueba se redimensionan de 28x28 a 32x32 utilizando la función cv2.resize, luego las imágenes se convierten a 3 canales, utilizando la función np.repeat para replicar los canales de la imagen en 3 dimensiones (rojo, verde, azul).

Después se utiliza ResNet50 con pesos pre entrenados desde la base de datos ImageNet, las capas de ResNet se congelan para evitar que se actualicen durante el entrenamiento, esto se realiza para construir un modelo secuencial que consiste en la base de ResNet50 seguida de una capa de agrupación global, una capa densa con 256 unidades y activación ReLU, una capa de dropout con tasa del 0.5 y una capa de salida con activación softmax para clasificar imágenes en 10 categorías.



- **Resultados finales**

	Nombre	Tiempo de ejecución	Perdida	Accuracy	Val_Accuracy	Val_Loss
22	Modelo CNN 2	500.212085	0.237054	0.914067	0.9074	0.256560
20	Modelo Relu 4 capas ocultas Nadam 0.0001	93.528831	0.051581	0.980517	0.8985	0.868317
18	Modelo Relu 4 capas ocultas Adagrad 0.01	83.014182	0.065539	0.977550	0.8932	0.452978
21	Modelo CNN 1	185.272544	0.245923	0.910317	0.8912	0.296225
16	Modelo Relu 4 capas ocultas Adam 0.0001	90.817668	0.201232	0.926517	0.8881	0.319545
19	Modelo Relu 4 capas ocultas Adadelta 1.0	95.813186	0.177562	0.935933	0.8858	0.555951
2	Modelo Relu 4 capas ocultas	64.837876	0.227586	0.915517	0.8783	0.345055
4	Modelo Tanh 3 capas ocultas	59.431098	0.273793	0.900517	0.8783	0.341514
11	Modelo Relu 4 capas ocultas con dropout	69.670721	0.339347	0.880833	0.8764	0.344645
17	Modelo Relu 4 capas ocultas RMSprop 0.0001	82.564986	0.138265	0.949683	0.8764	0.497750
1	Modelo Relu 3 capas ocultas	63.941934	0.244174	0.911617	0.8758	0.349339
0	Modelo Relu 2 capas ocultas	52.780692	0.285140	0.895983	0.8751	0.349939
14	Modelo Relu 4 capas ocultas con Early Stopping y DropOut	70.874614	0.341163	0.882967	0.8749	0.348947
13	Modelo Relu 4 capas ocultas con Early Stopping	54.999362	0.244772	0.909300	0.8740	0.343446
3	Modelo Tanh 2 capas ocultas	52.307445	0.295721	0.892817	0.8731	0.359482
5	Modelo Tanh 4 capas ocultas	65.857923	0.261160	0.905500	0.8727	0.349506
6	Modelo Sigmoid 2 capas ocultas	52.039589	0.465559	0.835700	0.8232	0.491043
10	Modelo Relu 4 capas ocultas con regularización L2	68.942703	0.932188	0.832383	0.8219	0.956483
15	Modelo Relu 4 capas ocultas con todas las regularizaciones	81.048001	1.035950	0.784900	0.8130	0.924754
7	Modelo Sigmoid 3 capas ocultas	59.673061	0.575680	0.791267	0.7841	0.587976
23	Modelo ResNet	2084.453433	0.635425	0.764450	0.7755	0.610073
8	Modelo Sigmoid 4 capas ocultas	64.388945	0.750073	0.728850	0.7312	0.756893
9	Modelo Relu 4 capas ocultas con regularización L1	71.276685	1.450034	0.695317	0.7006	1.417843
12	Modelo Relu 4 capas ocultas con regularización L1 y L2	76.625782	1.703504	0.621150	0.6164	1.712871

Los modelos convolucionales (CNN) como Modelo CNN 2 y Modelo CNN 1 muestran un buen equilibrio entre tiempo de ejecución y precisión de validación, con Modelo CNN 2 obteniendo la mayor Val_Accuracy de 0.9074.

Las redes neuronales con varias capas ocultas y diferentes optimizadores también muestran resultados prometedores. Por ejemplo, el Modelo Relu 4 capas ocultas Nadam 0.0001 tiene un alto Accuracy de 0.9805, pero su Val_Accuracy es ligeramente inferior a la de los modelos CNN.

El Modelo ResNet, a pesar de su mayor tiempo de ejecución, tiene un rendimiento inferior en términos de precisión de validación (Val_Accuracy de 0.7755), lo que puede indicar la necesidad de un ajuste fino adicional o más épocas de entrenamiento.

Los modelos con funciones de activación Sigmoid y Tanh generalmente muestran menor rendimiento comparado con los modelos ReLU y CNN, lo que resalta la efectividad de ReLU en redes profundas. Además, la regularización L1 y L2 no siempre mejora el rendimiento, como se observa en los modelos con regularización que obtienen menor precisión y mayor pérdida.

En conclusión, los modelos convolucionales y las redes profundas con optimizadores adecuados como Adam y Nadam muestran el mejor rendimiento en la tarea de clasificación de imágenes Fashion-MNIST.

- **Verificación de las mejoras propuestas**

1. *Implementar una Red Neuronal Convolucional (CNN)*
Estado: Cumplido. Se han implementado dos modelos CNN con diferentes arquitecturas y capas.

2. Transferencia de aprendizaje
Estado: Cumplido. Se han implementado ResNet50 pre entrenada y ajustado las últimas capas para adaptarla al conjunto de datos Fashion-MNIST.

3. Aumento de datos
Estado: No cumplido. No hay implementación de técnicas de aumento de datos.
Sugerencia: Puedes agregar aumento de datos utilizando ImageDataGenerator de Keras.

4. Optimización hiperparamétrica automatizada
Estado: No cumplido. No se ha implementado ninguna herramienta para la búsqueda de hiperparámetros debido a la escasez de recursos.
Sugerencia: Es posible integrar herramientas como Optuna para automatizar la búsqueda de los mejores hiperparámetros.