TECNOLÓGICO DE MONTERREY

Redes Neuronales

Proyecto de Aprendizaje Profundo

Kevin Jesús Martínez Trinidad // **A00834493**
Miranda Isabel Rada Chau // **A01285243**
Juan José Hernández Beltrán // **A00836747**

Profesor
Enrique Conant Pablos

Grupo: 101

04 de septiembre de 2023

Índice

Introducción

Con este proyecto intentamos generar modelos de redes neuronales profundas que nos permitirán clasificar cada imagen tratada en la categoría correspondiente. Para la realización de este proyecto vamos a utilizar la base de datos Fashion_MNIST. Esta base de datos está conformada por imágenes de diferentes prendas de ropa y sus etiquetas. Está base de datos tiene 60,000 datos para entrenar a un modelo y 10,000 datos para probarlo. Cada una de las imágenes tiene un tamaño de 28x28 pixeles. En este proyecto crearemos dos redes neuronales diferentes para clasificar las imágenes de la base de datos, y se compararán para ver cuál tuvo un mejor rendimiento al clasificarlas. Como parte de este proyecto fue necesario investigar para encontrar más tipos de redes neuronales. Además de los vistos en clase. Vamos a desarrollar dos redes neuronales, probarlas con la base de datos que mencioné anteriormente y discutir cuál de estas dos redes es mejor para clasificar a las prendas de ropa.

Metodología y Resultados

Para la elaboración de este proyecto tuvimos que elegir dos tipos de redes neuronales diferentes. En nuestro caso, decidimos utilizar Redes Convolucionales, una de las redes neuronales vistas en el curso, y Redes neuronales residuales, la cual investigamos para este proyecto.

Elegimos a las redes convolucionales ya que estás son muy buenas identificando patrones en los datos, lo cual nos es útil debido a la naturaleza de las imágenes de Fashion MNIST, en donde la forma de cada prenda es una de las únicas maneras de diferenciar entre los diferentes artículos de ropa. Debido a esta característica de las redes convolucionales, creemos que esta puede tener un buen rendimiento al clasificar a las diferentes prendas de ropa. En el caso de las redes neuronales residuales, las elegimos porque estas también buscan identificar patrones en las imágenes. Además, estas redes son flexibles y eso ayuda a que puedan clasificar imágenes apropiadamente.

Antes de comenzar a crear las redes neuronales tuvimos que realizar un preprocesamiento para los datos. En ambos modelos, se normalizaron los datos para que estos se encontraran en un rango de 0 a 1. Después de esto fue necesario utilizar one-hot encoding para que las etiquetas de las imágenes fueran numéricas. Nosotros decidimos definir un vector binario que indicará la clase a la que pertenece cada una de las imágenes. Decidimos tratarlo de esta manera porque esto permitiría que al final del modelo se pueda representar la probabilidad de que la observación pertenece a la clase. Esto nos puede ayudar a identificar la precisión de nuestro modelo.

Modelo Convolucional

Nuestro modelo convolucional está compuesto por 12 capas, de estas 5 son convolucionales, 4 de pooling, 1 de flatten, 2 densas y una de salida. Para entrenar a este modelo probamos varias combinaciones diferentes de los hiperparámetros con la intención de encontrar el mejor. Específicamente, en los experimentos cambiamos la cantidad de épocas, el optimizador, el costo, el número de neuronas en las capas, entre otros parámetros. A continuación, se puede ver una tabla donde se exponen los diferentes experimentos realizados y el rendimiento correspondiente a cada uno.

Tabla de Resultados Modelo Convolucional

Prueba	Épocas	Tamaño del batch	Optimizador	Costo	Capas Convolucionales	Capas Pooling	Capas Densas	Tiempo	Accuracy
1	5	16	Adam	Categorical crossentropy	1. 32, (3,3), 1, same, relu. 2. 16, (3,3), 1, same, relu.	1. (2,2),2, valid 2. (2,2),2, valid.	1. 64, relu	15 min	0.9118
2	5	8	Adam	Categorical crossentropy	1 32, (3,3), 1, same, relu. 2. 32, (5,5), 1, same, relu.	1. (2,2),2, valid 2: (2,2),2, valid.	1: 128, relu	2.5 min	0.9139
3	5	16	Adam	Categorical crossentropy	1 64, (3,3), 1, same, relu. 2: 32, (5,5), 1, same, relu.	1: (2,2),2, valid 2: (2,2),2, valid	1: 128, relu	3 min	0.9023
4	5	32	RMSProp	Categorical crossentropy	1 64, (3,3), 1, same, relu. 2: 32, (5,5), 1, same, relu. 3: 16, (3,3), 1, same, tanh	1: (2,2),2, valid 2: (2,2),2, valid 3: (2,2),2, valid	1: 128, relu	3.5 min	0.9079
5	10	64	Adam	Categorical crossentropy	1 64, (3,3), 1, same, relu. 2: 32, (5,5), 1, same, relu. 3: 16, (3,3), 1, same, tanh	1: (2,2),2, valid 2: (2,2),2, valid 3: (2,2),2, valid	1: 128, relu	6.5 min	0.9119
6	10	128	Adam	Categorial crossentropy	1 32, (3,3), 1, same, relu. 2: 32, (5,5), 1, valid, relu 2: 16 (5,5), 1, same, relu.	1: (2,2),2, valid 2: (2,2),2, valid 3: (2,2),2, valid	1: 256, relu 2: 128, relu	9 min	0.9068

					3: 8, (3,3), 1, valid, relu 5: 8, (3,3), 1, same, tanh				
7	10	128	Adam	Categorical Crossentropy	1: 32, (3,3), 1, same, relu. 2: 32, (3,3), 1, valid, relu.	1: (2,2),2, valid	1: 256, relu	8.5 min	0.9214
8	8	128	Adam	Categorical Crossentropy	1: 35, (3,3), 1, valid, relu. 2: 50, (3,3), 1, valid, relu.	1: (2,2),2, valid	1: 256, relu	8.5 min	0.9229
9	12	128	Adam	Categorical Crossentropy	1: 35, (3,3), 1, valid, relu. 2: 50, (3,3), 1, valid, relu.	1: (2,2),2, valid Además se agregaron 2 capas batch normalization y un dropout de 0.7.	1: 256, relu	39 min	0.9252

Durante la búsqueda de hiperparámetros tomamos en consideración el tipo de modelo utilizado, el tamaño de los datos y la complejidad del problema para eficientizar lo más posible la búsqueda de los mejores. Nos dimos cuenta de que, al agregar demasiadas capas o bloques de capas, el problema comenzaba a sufrir de sobre-entrenamiento, reduciendo el desempeño que se tenía, razón por la cual después del intento 6 se decidió por eliminar muchas de las capas con las que se contaba. Situación similar ocurrió con el número de filtros de las capas convolucionales.

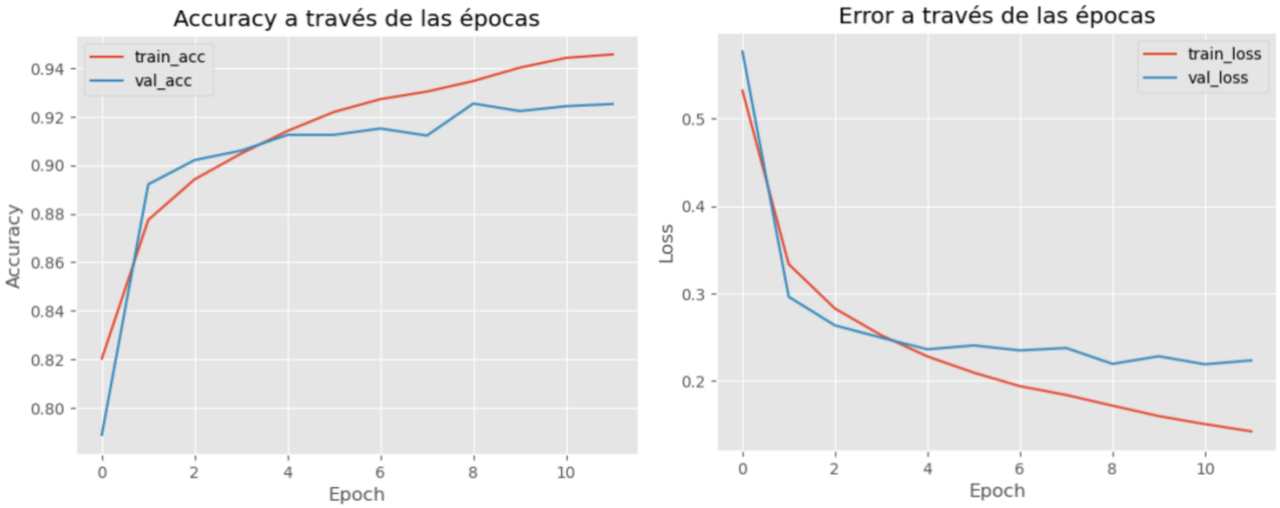
Por la razón antes mencionada se optó por utilizar únicamente dos capas convolucionales, cada una seguida de una capa de batch normalization para que los datos fueran analizados de una manera más efectiva. También se decidió agregar una capa de dropout, con lo cual se logró reducir el overfitting considerablemente.

Además de los ajustes efectuados en la tabla, se realizaron otros antes y durante esta que no fueron incluidos, con la finalidad de no saturar de información. Algunos de los insights que obtuvimos de estas y que nos guiaron para llegar a la combinación final de hiperparámetros presentados en la última fila de la tabla fueron los siguientes:

- Un número de épocas mayor a 12 parecía sobreentrenar el modelo o mejorarlo insignificantemente, por lo que decidimos utilizar principalmente un número de épocas de 8 o 10, al final siendo de 12.
- Un número demasiado pequeño o grande de batch empeoraba el desempeño, por lo que se decidió dejarlo en 128.

- Se probaron con la mayoría de los optimizadores que ofrece la librería de Keras, y aunque había otros que obtuvieron desempeños muy buenos como RMSprop o Adamax, ninguno superó en este aspecto a Adam, el optimizador más utilizado.
- Se experimentó con una buena parte de las funciones de error categóricas disponibles en Keras, sin embargo, la mayoría de ellas reducía el desempeño del modelo significativamente. Únicamente la divergencia de Kullback-Leiblier logró tener un desempeño cercano a la entropía categórica cruzada, esta última siendo la que se decidió utilizar.
- Los parámetros de la capa de pooling no se modificaron, pues desde el inicio vimos que si cambiábamos el tipo de pooling, el desempeño empeoraba considerablemente.
- Si aumentábamos mucho la taza de aprendizaje, el desempeño se volvía definitivamente peor, por lo que se decidió disminuirla, aunque eso significara aumentar el número de épocas. Al final la tasa de aprendizaje utilizada fue de 0.0005.
- Mientras mayor fuera la tasa de dropout, menos se sobreentrenaba el modelo, y como este era un problema importante, la tasa terminó siendo de 0.7.

El modelo descrito en la última fila resultó ser aquel con mejor rendimiento, tardando relativamente poco tiempo en entrenarse (aunque mucho más comparado con las otras pruebas). A continuación, se muestran las gráficas de precisión y error de los datos de entrenamiento y prueba a través de las épocas.



En base a las gráficas que se muestran en esta sección, se puede ver que el modelo muestra una precisión más alta en el entrenamiento, que en la validación. Sin embargo, debido a que la diferencia entre las dos medidas no es tan alta, no consideramos que exista un sobre ajuste en nuestro modelo. Por lo tanto, podemos decir que esta red neuronal sí tiene un

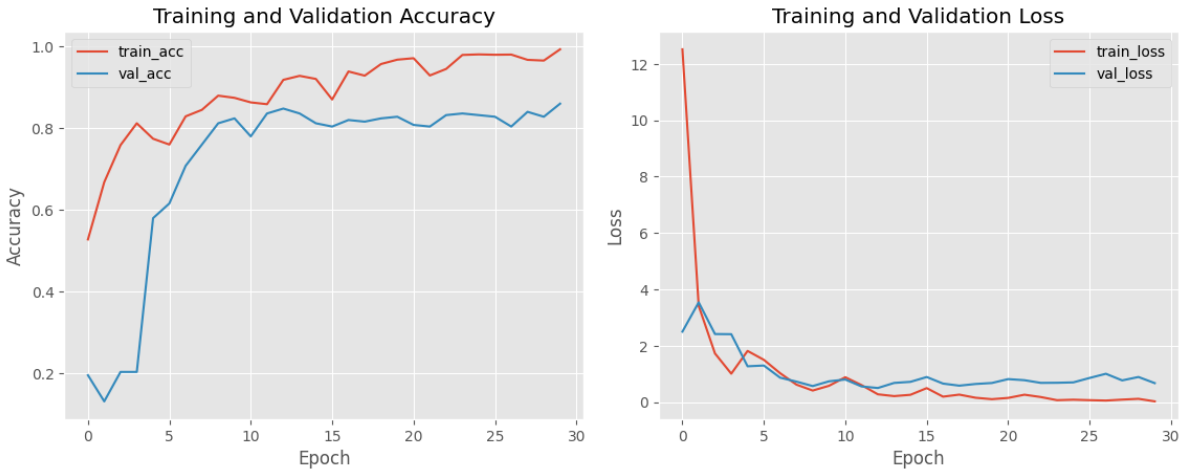
buen rendimiento, ya que tiene una precisión del 92%, y por ende se puede decir que es un buen clasificador para estas imágenes.

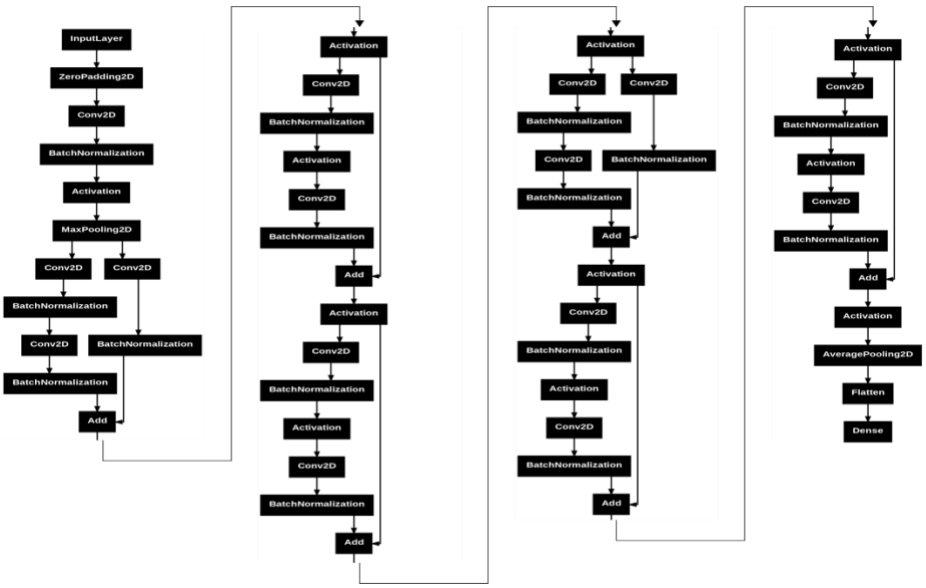
Modelo Residual

En el caso de la red neuronal residual, tuvimos que programar un bloque de identidad y un bloque convolucional. Estas son algunas de las estructuras necesarias para crear una red neuronal de este tipo. Estas estructuras involucraron a varias de las funciones que aprendimos durante el curso como la normalización de lotes, filtros, capas convolucionales, entre otras. Después establecimos la arquitectura de nuestra red. En este caso, creamos una red usando "zero-padding" y varias capas de diferentes tipos incluyendo capas convolucionales, densas, pooling y de flatten.

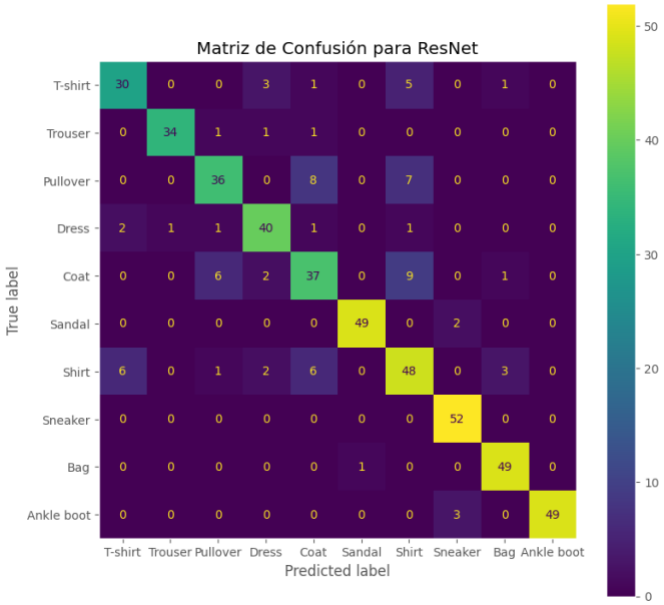
Después de establecer una arquitectura tentativa basada en otras encontradas en artículos de investigación que utilizan este tipo de redes (2), fue posible comenzar con el entrenamiento. Con el fin de encontrar la mejor arquitectura para nuestro modelo, se hicieron varias pruebas con diferentes parámetros. Algunos de los parámetros que cambiaron fueron las épocas, el número de lotes, el optimizador, el número de capas, entre otros.

El principal problema que se presentó fue el overfitting, el cual se combatió reduciendo la complejidad del modelo y, en otros casos, añadiendo capas de dropout (aunque este método se descartó al final porque no fue tan efectivo).





Después de hacer el modelo, fue posible medir su desempeño. En este caso, se puede ver que el desempeño en general fue bueno, ya que el modelo clasificó bien más del 68% en todas las categorías. Hay algunas prendas que clasifican mejor, una de estas son los "trousers", esto se sabe ya que el modelo clasificó bien el 97% de las imágenes. En general, el modelo tiene una precisión del 85%, consideramos que esta es una medida relativamente buena, aunque es mejor el CNN creado.



En esta matriz de confusión se puede apreciar visualmente el desempeño de esta red. En este caso, se puede ver que la variable con mejor precisión fue la de "trouser", donde solo

hubo un error de clasificación. Esto corrobora lo mencionado anteriormente en cuanto a las métricas calculadas. Con esta matriz también podemos ver cuales prendas le causan más problemas al modelo a la hora de clasificar. En este caso, se puede ver que el modelo suele confundirse entre “coat”, “pullover” y “shirt”.

Discusión

El mejor resultado se obtuvo a utilizando una red neuronal convolucional (92.5% de precisión), lo cual es explicable considerando que su propósito fundamental es el reconocimiento de imágenes. Por su parte, la red neuronal residual, que de igual modo utiliza capas de convolución, está pensada para problemas más grandes y complejos en los que se deben combatir problemas con la expansión de los gradientes (añadiendo sus características funciones residuales, saltos, mapeos de identidad, y bloques residuales/ convolucionales), este “exceso de complejidad” en el modelo se volvió evidente con los problemas de overfitting que presentamos en un inicio. Simplificar la red residual fue el camino a seguir y aun cuando terminaron eliminándose muchas de sus capas (más no las que otorgan sus propiedades “residuales”) y el resultado en el conjunto de prueba fue aceptable (85% precisión), el overfitting seguía estando, en menor medida, presente.

Vale la pena también mencionar que el tiempo de entrenamiento de una red residual es notablemente mayor al de una red convolucional típica, lo cual lo vuelve más tardado para probar con diferentes hiperparámetros y, por lo tanto, más latoso.

Cabe mencionar que, aunque la métrica de desempeño indica que el modelo convolucional es mejor prediciendo la categoría correcta, el modelo residual se entrenó mejor para los datos de entrenamiento, ya que en este su exactitud para encontrarlos es cercana al 100%, lo cual nos indica que este modelo es muy bueno para predecir los datos de entrenamiento, pero no generalizando.

En este problema, y en general (principio de parsimonia), la solución más simple resultó ser la mejor.

Conclusiones

Después de haber realizado este proyecto, fuimos capaces de identificar algunas posibles mejoras en cada uno de nuestros modelos. En el caso de la red neuronal convolucional, creemos que con más tiempo podríamos probar con diferente orden de las capas o incluso combinaciones de tamaños y número de filtros diferentes. Esto mismo sucedió con la red

neuronal residual. También nos hubiera gustado tener la oportunidad de experimentar más con diferentes hiperparámetros y diferentes métricas, pero consideramos que eso también se podría llevar a cabo después si en algún momento tenemos la oportunidad de extender esta investigación.

Este proyecto nos permitió reforzar y practicar lo aprendido en el curso, sobre todo en la creación de redes neuronales y la investigación de diferentes tipos de redes y sus aplicaciones. Este proyecto también nos permitió ver cómo se relacionan las redes entre sí, a pesar de que utilizan diferentes hiperparámetros y estructuras. También consideramos que fue muy útil, ya que nos permitió ver cuál es el verdadero impacto de los hiperparámetros y la cantidad de épocas en el rendimiento de un modelo.

Lo que no nos gustó es que podía llegar a ser tardado especialmente debido a que hicimos tantas pruebas con diferentes hiperparámetros, hubo muchos tiempos de espera debido a la velocidad de nuestras computadoras y la cantidad de pruebas que realizamos.

Por otro lado, lo que más nos gustó de este proyecto fue que tuvimos la oportunidad de aplicar estos modelos con libertad en cuanto a la cantidad de capas, la cantidad de neuronas, la cantidad de épocas, etcétera. Creemos que esto fue muy útil como para ir a aprendiendo más y comprendiendo la importancia de conocer y entender estas redes neuronales y los elementos que las componen.

Referencias

1. Azeem, I. (2023, noviembre 14). *Understanding ResNet Architecture: A Deep Dive into Residual Neural Network*. Medium; Medium. https://medium.com/@ibtedaazeem/understanding-resnet-architecture-a-deep-dive-into-residual-neural-network-2c792e6537a9
2. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778). https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf