Proyecto 4 - Inteligencia Artificial

*Link del repositorio

Chahua Salgueran, Luis Eduardo
Ciencia de la computación
Universidad de Ingeniería y Tecnología
Lima, Perú
luis.chahua@utec.edu.pe

Riveros Soto, Andres Jaffet
Ciencia de la Computación
Universidad de Ingeniería y Tecnología
Lima, Perú
andres.riveros@utec.edu.pe

Huby Tuesta, Jean Paul
Ciencia de la Computación
Universidad de Ingeniería y Tecnología
Lima, Perú
jean.huby@utec.edu.pe

***Palabras clave*—*Deep learning*, CNN, *Pytorch*, *epoch*, *batch normalization*, *dropout*.**

***Resumen*—El objetivo principal de este proyecto es desarrollar un modelo de *deep learning* basado en redes neuronales convolucionales(*CNN*) para la clasificación automática de imágenes de rayos X en 4 diferentes tipos de clase(COVID, NO-COVID, Opacidad y Neumonía Viral). Para ello, se utilizó el dataset *COVID-19 Radiography Database* proporcionado por la plataforma *Kaggle*, el cual presenta 21165 imágenes divididas en las 4 clases definidas anteriormente.**

I. Introducción

La gran cantidad de volúmenes de datos y la aparición de las $GPU's$ hizo posible que la inteligencia artificial pueda tomar una mayor presencia en estos últimos años. Gracias a ello, actualmente existen muchas aplicaciones de la *AI* a diversos campos como las ventas, la ingeniería, la ciencias exactas, etc. Uno de las subáreas más conocidas de la *AI* es el *Machine Learning*.

El objetivo principal del *Machine Learning* es la predicción de resultados utilizando diversas técnicas como la regresión lineal, regresión multivariada, técnicas de *clustering*, etc. Dentro de este campo existe el subcampo llamado *Deep Learning*, en el cual se utilizan las llamadas redes neuronales. Esto permite obtener modelos con una mayor complejidad y así poder obtener realizar tareas más complejas con un porcentaje alto de *accuracy*.

Para ello, se utilizan las llamadas redes *Multilayer Perceptron* o *MLP*, que están compuestas de varias capas de neuronas y cuyo objetivo es, como se mencionó anteriormente, añadirle mayor complejidad al modelo mediante la cantidad de capas y neuronas. Sin embargo, el uso de *MLP* puede ser muy costoso si nos enfocamos en la eficiencia, puesto que la cantidad de parámetros que se utilizan va de manera proporcional al producto de la cantidad de neuronas de 2 capas consecutivas.

Además, si trabajamos con imágenes como *input*, no se toma en cuenta la posición de los píxeles en la imágen, lo cual hace que no se consideren las carácterísticas espaciales. Una técnica para considerar estos aspectos es la llamada *Convolutional Neural Network* o *CNN*. Es por ello que en este proyecto, se aplicará el uso de las *CNN*.

Para mayor entendimiento, acceder al repositorio donde se podrán ver las implementaciones.

II. Marco Teórico

II-A. Convolucion

Para entender estas redes primero necesitamos definir la operación de convolución.La operación de convolución es simplemente la multiplicación entre una matriz de filtro (kernel) y una sección de la matriz de una imagen, el resultado se guarda en la coordenada respectiva de la matriz resultante tal como se puede ver en la figura 1.

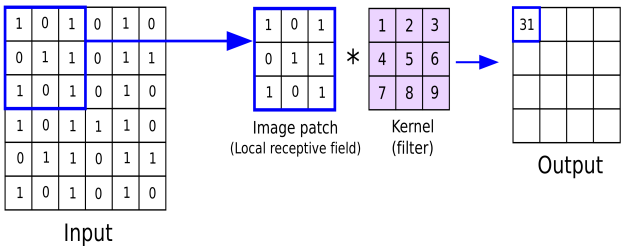

Figura 1. *Convolution*

La manera en como esta operación es realizada se define por parámetros como *stride* y el tamaño del kernel, así cuanto valores decidimos agregar a la imagen original como *padding*.

Lo que es especialmente útil de las operaciones de convolución es su capacidad para poder detectar *feautures* que pueden aparecer a lo largo de una imagen. Si un *feauture* puede ser detectado en una coordenada (x1,y1) también podrá ser detectada en una coordenada (x2,y2) en la imagen. Este atributo de la convolución hace que sea muy conveniente para la detección en imágenes.

II-A1. Pooling: La idea principal del *pooling* es reducir la dimensión de la matriz y aceptar algunas características basadas en algunas suposiciones, de forma similar a un filtro aplicado al mapa de características. Se usan diferentes técnicas

de *pooling*, como *max pooling*, *min pooling*, *mean pooling*, etc. En la arquitectura del proyecto se utiliza *max pooling*, de tamaños distintos n * n. Este tamaño n * n de la matriz de *pooling* se coloca en los píxeles de la imagen y se extrae el píxel máximo de la imagen en matrices n * n. En la figura 2, se puede observar un ejemplo para una matriz de *pooling* de 2 x 2.

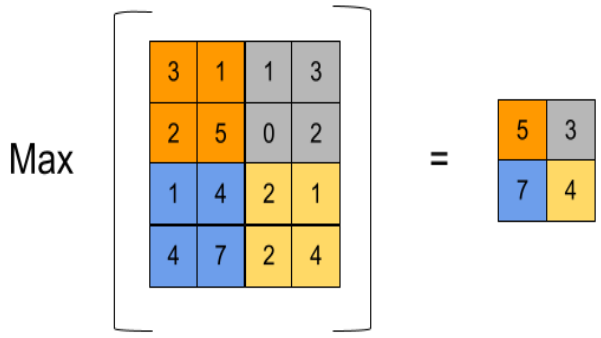

Figura 2. *Max-Pooling*

II-B. Convolutional Neural Network (CNN)

Las redes convoluciones son unas especias de redes que se especializan en el procesamiento de data cuya topología es importante, en donde los datos vecinos influyen para el valor de la data actual. Una red neuronal tradicional convertiría la data en un vector, perdiendo la data de los sus vecinos. El caso mas común de esto son en la data que proviene de imágenes.

Hay dos operaciones esencial para las redes *CNN*: *Pooling* y *Convolución*. Sirviendo para detectar *features* y luego reducir la dimensiones de las imágenes. Lo cual sirve para poder detectar los diversos *features*.

Esta colección de *features* pueden ser usado después por una capas neuronales mas tradicionales para poder clasificar imagen atreves de esos *features*. La combinación de estos elementos sirven para crear una arquitectura como en la que es mostrada en la figure 3.

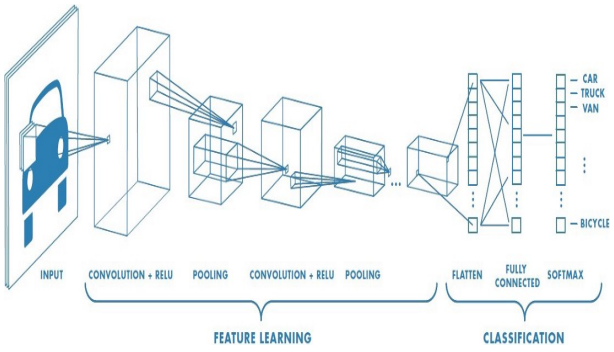

Figura 3. *CNN network*

Lo que aprende la red CNN es no solo como detectar clasificar las imágenes atreves de los *features*, sino que también pueden ser capaces de aprender los valores de los kernels para poder detectar estos.

II-B1. Activate Function: Son expresiones matemáticas usadas para encontrar el resultado de la red neuronal. Existen varios tipos de funciones de activación, como ReLU, Leaky ReLU, Sigmoid, etc. En donde ReLU es la más usada porque se obtiene valores positivos para entrada de valores positivos, caso contrario, cero. Además, se usa para reducir el problema del desvanecimiento de la gradiente.

II-C. Dropout

Es una técnica en que las neuronas seleccionadas al azar se ignoran durante el entrenamiento del conjunto de datos. Lo que significa que la contribución a la activación de neuronas posteriores se elimina temporalmente en el paso hacia adelante y no se aplica ninguna actualización de peso a la neurona en el paso hacia atrás.

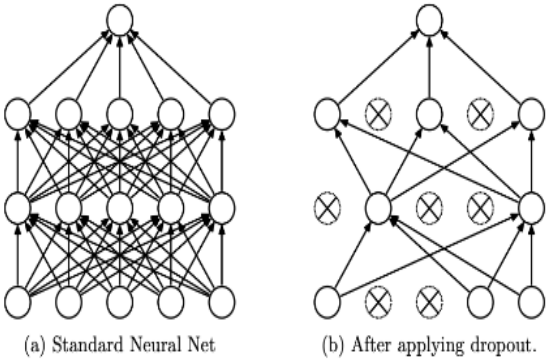

Figura 4. *Dropout*

II-D. Batch Normalization

Es un método que mejora el entrenamiento en rapidez y estabilidad en redes neuronales profundas. Y consiste en normalizar los vectores de activación de las capas ocultas usando la media y varianza del *batch* actual. Y este paso de normalización puede ser aplicado antes o después de la función no lineal.

II-E. Early Stopping

Es una forma de regularización usado para evitar *overfitting*. Y se desarrolla almacenando y actualizando los mejores parámetros actuales durante el entrenamiento, y cuando las actualizaciones de parámetros ya no producen una mejora (después de un número de iteraciones), deja de entrenar y se obtiene los últimos mejores parámetros del modelo.

III. Experimentos

III-A. Exploración de la data

El *dataset* proporcionado contiene imágenes de rayos X divididos en 4 clases(COVID, NO-COVID, Opacidad y Neumonía Viral). Asimismo, contiene también las máscaras de las imágenes. Este *dataset* fue dividido de forma aleatoria en 70 %

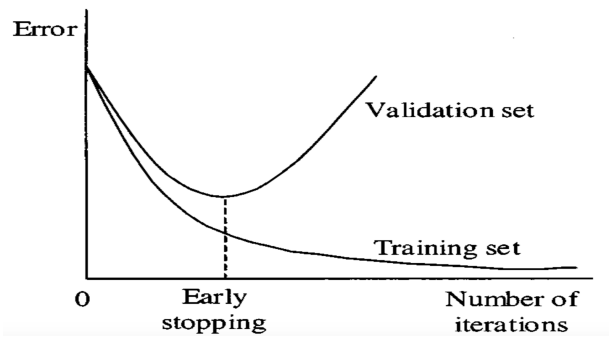

Figura 5. *Early Stopping*

para el *training*, 20 % para el *validation* y 10 % para el *test*. Cabe resaltar que en esta división se conservó la distribución inicial de datos entre las 4 clases. Es decir, tanto para el *training*, *validation* y *test* la proporción de imágenes de las 4 clases es iguales. Luego de cada gráfica, se mostrará el resultado final del *validation loss* y *test*.

Cabe mencionar que no se aplico la mascara a las imágenes de las radiografías debido a que en experimentos preliminares se noto una baja notoria en el rendimiento de los modelos. Además, las imágenes se leyeron con escala de grises debido que no todas las imágenes estaban en el formato RGB.

III-B. Consideraciones

El modelo base que se usara para la red es el que fue dado en la consigna, se puede visualizar la arquitectura de este modelo en la figura 6.

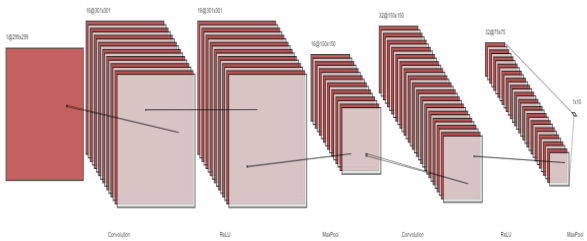

Figura 6. *Model Arquitecture*

Se probaran diversas redes neuronal, usando como referencia el modelo base, y técnicas de regularización para poder maximizar la métrica de *accuracy* de la predicción. Se adicionarán 0, 1 y 2 capas de convolución adicionales al código dado, y para cada uno de ellos se experimentará agregando *batch normalization* en cada capa, *batch normalization* y *dropout* en cada capa y *batch normalization* y *dropout* solo en la primera capa. Además, en cada experimento se aplicó *Early Stopping* para evitar *overfitting*. El orden en el cual se usara las técnicas de regularización es el siguiente:

1. Convolution
2. Droput
3. Relu
4. Max Pool
5. Batch Norm

El orden de estas operaciones es tema de discusión, así que un cambio de estas puede representar una mejora o empeora de los resultados.Otros factores a notar es que se uso un optimizado Adam, y se tomo un *Cross Entropy Loss* para el calculo de la perdida. El primero, debido que no queremos que el modelo se quede estancado en mínimos locales; mientras que el segundo se es para medir la perdida dado a que es clasificador de múltiples categorías.

III-B1. Adicionando 0 capas de convolución: Para la experimentación con la clase base se usaron las variaciones propuestas, el resultado de estas se encuentra en las gráficas de abajo:

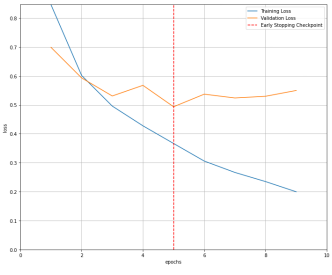

Figura 7. Sin Batch Normalization y Dropout

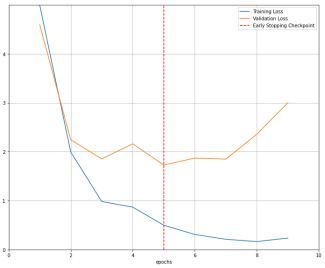

Figura 8. Batch Normalization

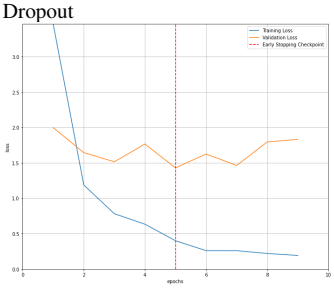

Figura 9. Batch Normalization y Dropout en cada capa

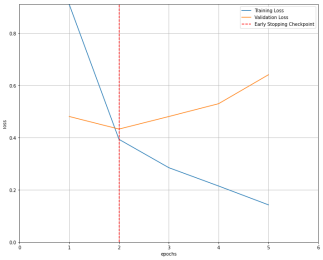

Figura 10. Batch Normalization y Dropout en la primera capa

El resultado de *accuracy* que se uso del modelo obtenido para cada variación se puede observar en la siguiente tabla I. No hay una mejora en ninguno de los casos, incluso agregando un factor regularizador no parece mejorar los modelos. Así mismo, si observamos la gráficas, podemos ver que el modelo encuentra el mejor resultado en *epochs* cercanos.

Cuadro I
RESULTADOS DE LOS *test* Y *validation*

	Val Acc	Test Acc
Without BatchNorm-Dropout	83.21	83.34
BatchNorm	83.12	83.15
BatchNorm-Dropout	82.41	82.82
1st layer BatchNorm-Dropout	83.89	83.77

Por lo que podemos concluir, que el modelo tiene un alto bias. Es decir, que se necesita agregar complejidad para que este pueda detectar features mas complejos.

III-B2. Adicionando 1 capa de convolución: Se puede observar los resultados de la gráfica en de los resultados de la siguiente manera:

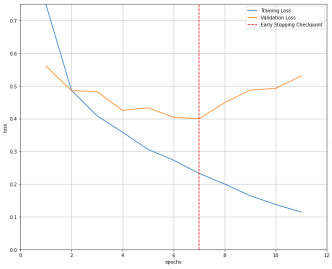

Figura 11. Sin Batch Normalization y Dropout

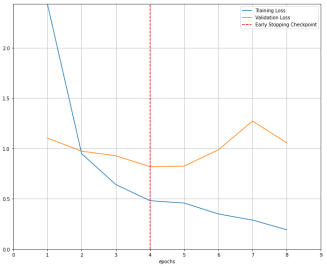

Figura 12. Batch Normalization

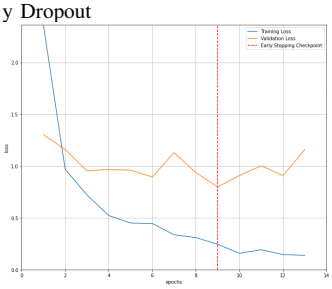

Figura 13. Batch Normalization y Dropout en cada capa

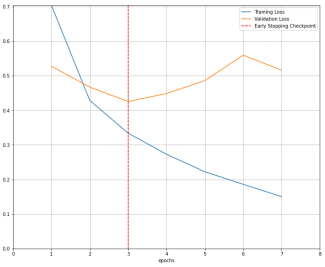

Figura 14. Batch Normalization y Dropout en la primera capa

Como se puede observar en la tabla II los resultados de los modelos muestran un resultados mas positivos comparados con el modelo base. Cabe notar que en este modelo también se ve un empeoramiento de los regularizadores. Así mismo, la precision sigue siendo todavía muy baja por lo que se puede llegar a una conclusión similar a la del modelo base.

Cuadro II
RESULTADOS DE LOS *test* Y *validation*

	Val Acc	Test Acc
Without BatchNorm-Dropout	86.39	85.73
BatchNorm	83.85	84.39
BatchNorm-Dropout	84.20	83.92
1st layer BatchNorm-Dropout	84.46	84.82

III-B3. Adicionando 2 capas de convolución: Los resultados de la gráficas se pueden observar a primera vista una mejora con respecto a la reducción de perdida dentro del conjunto de datos de validación.

De la misma manera, se puede apreciar que en todas las experimentaciones, mostradas en la tabla III, se logró un *accuracy* mayor al 85 %. Además, se obtuvo un mejor resultado cuando se agrega *batch normalization* y *dropout* solo en la primera capa. Cabe resaltar que los resultados tanto en el *validation accuracy* como en el *test accuracy* son bastante similares.

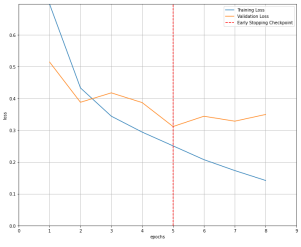

Figura 15. Sin Batch Normalization y Dropout

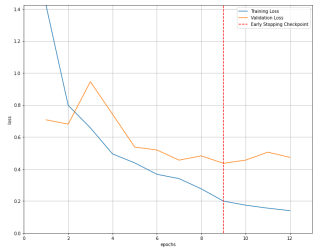

Figura 16. Batch Normalization

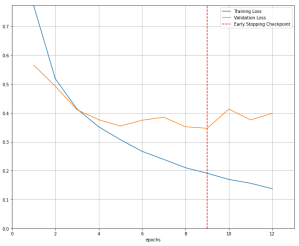

Figura 17. Batch Normalization y Dropout en cada capa

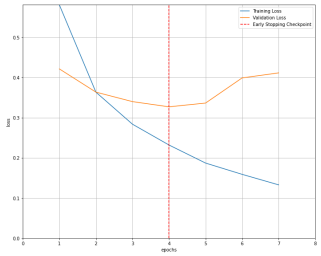

Figura 18. Batch Normalization y Dropout en la primera capa

Cuadro III
RESULTADOS DE LOS *test* Y *validation*

	Val Acc	Test Acc
Without BatchNorm-Dropout	87.940	88.263
BatchNorm	87.611	87.261
BatchNorm-Dropout	87.826	87.723
1st layer BatchNorm-Dropout	88.810	89.456

III-B4. Adicionando 2 capas de convolución y 1 capa lineal: Para el ultimo experimento, decidimos usar el modelo que mejor resultado nos dio con respecto a capas convoluciones. Al agregar una capa lineal aumentamos la complejidad del modelo. El resultado de los experimentos se ven en la siguiente gráfica III-B4.

Los resultados del *accuracy* de los experimentos se puede observar en la tabla IV. En donde observamos unos resultados similares al del modelo con 2 capas de convoluciones adicionales, con excepción del modelo con *BatchNorm-Dropout* que muestra los peores resultados. Y el modelo con solo *BatchNorm*, mostrando el mejor resultado hasta ahora, con 90.98 %.

Cuadro IV
RESULTADOS DE LOS *test* Y *validation*

	Val Acc	Test Acc
Without BatchNorm-Dropout	88.41	88.74
BatchNorm	90.76	90.98
BatchNorm-Dropout	83.12	82.10
1st layer BatchNorm-Dropout	88.74	90.12

IV. DISCUSIÓN

Se observa que existe un patrón en los resultados finales de *accuracy* en cada uno de los CNN, por ejemplo, en los resultados obtenidos aplicando ningún regularizador y usando

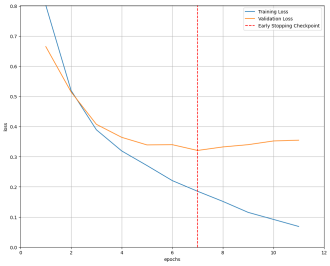

Figura 19. Sin Batch Normalization y Dropout

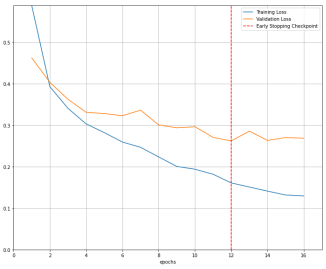

Figura 20. Batch Normalization

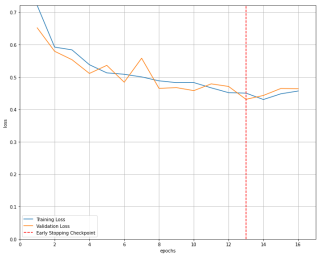

Figura 21. Batch Normalization y Dropout en cada capa

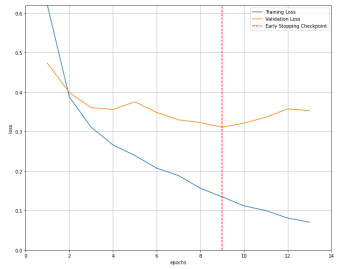

Figura 22. Batch Normalization y Dropout en la primera capa

regularizadores en la primera capa se obtuvo un *accuracy* relativamente superior con respecto a los mismos CNN que hacen uso de *batch normalization* o *batch normalization y dropout* en cada una de sus capas. En el caso de usar *dropout* en cada capa, este podría estar insertando mucho ruido en el modelo resultando en que el modelo no entrene correctamente. Además, el orden aplicado a cada una de las redes neuronales es distinto a lo común que suele usarse, y me refiero a colocar *batch normalization* al final de cada capa en lugar de colocarlo entre la convolución y la función de activación. Estos dos factores, *dropout* en cada capa y el orden de uso de *batch normalization*, pueden afectar el entrenamiento del modelo.

Con respecto al modelo con dos capas adicional CNN y dos capas lineales, se muestra un comportamiento similar al discutido por los demás modelos, además de mostrar los mejores resultados con respecto. Siendo notorio, el efecto negativo *Dropout* en el modelo. Mientras que *BatchNorm* parece tener un efecto positivo en la capas lineales.

V. Conclusiones

En general, podemos ver que todos los modelos arrojan un *accuracy* cercano al 85 % en cada uno de los CNN con o sin regulizadores. Sin embargo, un modelo arroja valores por encima del 85 %, que es el CNN con dos capas adicionales de *convolution* y *pooling*. Dentro de esta CNN el modelo en donde se agrega solo en la primera capa tanto el *batch normalization* como el *dropout* presentan un mayor accuracy que es 89 %. Note además que el valor del *accuracy* en promedio se va incrementando conforme aumenta la cantidad de capas. Además, se pudo observar que, al momento de agregar 1 capa lineal adicional se incrementó el valor del *accuracy* hasta 90.98 %. Además, se puede apreciar que se obtuvieron mejores resultados cuando no se agrega *dropout* en alguna de las capas convoluciones.

Referencias

- Mayank Mishra, *Convolutional Neural Networks, Explained*, *Towards Data Science* (2020, Aug 26).https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939
- Sumit Saha, *A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way*, *Towards Data Science* (2018, Dec 15). https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
- Sanskar Hasija, *HyperParameter Tuning: Fixing High Bias(Underfitting) in Neural Networks*, *Medium* (2021, Jul 18). https://medium.com/mlearning-ai/hyperparameter-tuning-fixing-high-bias-underfitting-in-neural-networks-5184ead3cbed
- Jason Brownlee, *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*, *Machine Learning Mastery* (2019, Jul 5).https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/