Reporte Implementación de un modelo de deep learning.

Se decidió abordar el problema de clasificación de radiografías en el pecho de pacientes con neumonía y pacientes sanos.

Este dataset se recuperó de la plataforma de kaggle en la siguiente liga: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

Muchas de las decisiones que se tomaron para entrenar este modelo se basaron en el artículo de (Lee & Lim, 2022), en el cual se abordó un problema similar con radiografías de pecho para pacientes con COVID-19.

Técnicas de regularización.

En este artículo se mencionan cuatro cosas importantes para el éxito de su modelo:

- DataAugmentation: Uno de los problemas que se enfrentaron en la clasificación de imágenes de radiografías fue el desbalance que existía entre ambas clases, por lo que optaron por generar nuevas imágenes utilizando una normalización de la intensidad de la imagen, y un random crop en las distintas imágenes.
- Modelos pre-entrenados: Tras realizar pruebas con diferentes métricas con distintas arquitecturas, concluyeron que la mas adecuada para este problema en particular era utilizar la arquitectura DenseNet201
- Early Stopping: Se menciono que utilizaron un método para parar el entramiento de la red cuando veían que la precisión dentro del set de validación no mejoraba pasando cierto número de epochs, y se recuperaban los mejores pesos.
- Castigar los pesos de la clase desbalanceada: En el artículo se menciona que utilizaron un método de ajustar los pesos de las clases dependiendo de la cantidad que existían de cada una, lo cual les arrojo mejores resultados.

Para el problema de clasificación entre pacientes con neumonía y pacientes normales no existía un desbalance tan grande por lo que no se tuvieron que utilizar las técnicas de regularización como castigar los pesos para las clases desbalanceadas.

Las técnicas de regularización que si se utilizaron para este problema fueron las siguientes.

- DataAugmentation: Se utilizo el siguiente código para cambiar las características de las imágenes dentro del set de entrenamiento.

```
train_datagen = ImageDataGenerator(
            rescale=1./255,
            brightness_range=(brightness_range[0],
brightness_range[1]),
            zoom_range=[zoom_range[0],zoom_range[1]]
```

```
        )
```

Y algunos ejemplos de como las imágenes quedaron después de los cambios.

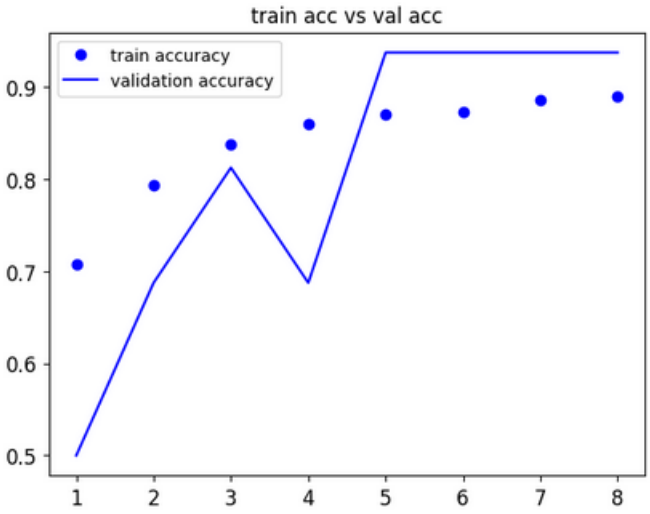
- Modelos pre-entrenados: Aunque en el artículo se sugería utilizar la arquitectura de DenseNet201, se opto por utilizar una versión más ligera de esta misma la cual es DenseNet169. Se agregaron 4 capas más, y la última capa de DenseNet169 se habilito para que se pudiera entrenar. Al final el modelo se ve de la siguiente manera.

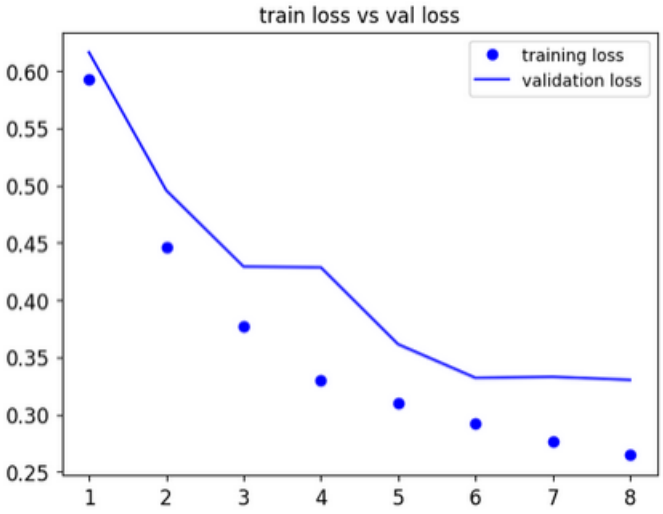
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet169 (Functional)    (None, 7, 7, 1664)        12642880

 global_average_pooling2d_1  (None, 1664)              0
  (GlobalAveragePooling2D)

 dropout_1 (Dropout)         (None, 1664)              0

 dense_2 (Dense)             (None, 1664)              2770560

 dense_3 (Dense)             (None, 1)                 1665

=================================================================
Total params: 15415105 (58.80 MB)
Trainable params: 2775553 (10.59 MB)
Non-trainable params: 12639552 (48.22 MB)
_________________________________________________________________
```

- EarlyStopping: Para esta parte se utilizó una función de tensorflow la cual nos permitía mantener un seguimiento de la precisión dentro del set de validación y determinar si el entrenamiento debería de parar cuando este no mejorara dentro de un número de paciencia que nosotros estimamos.

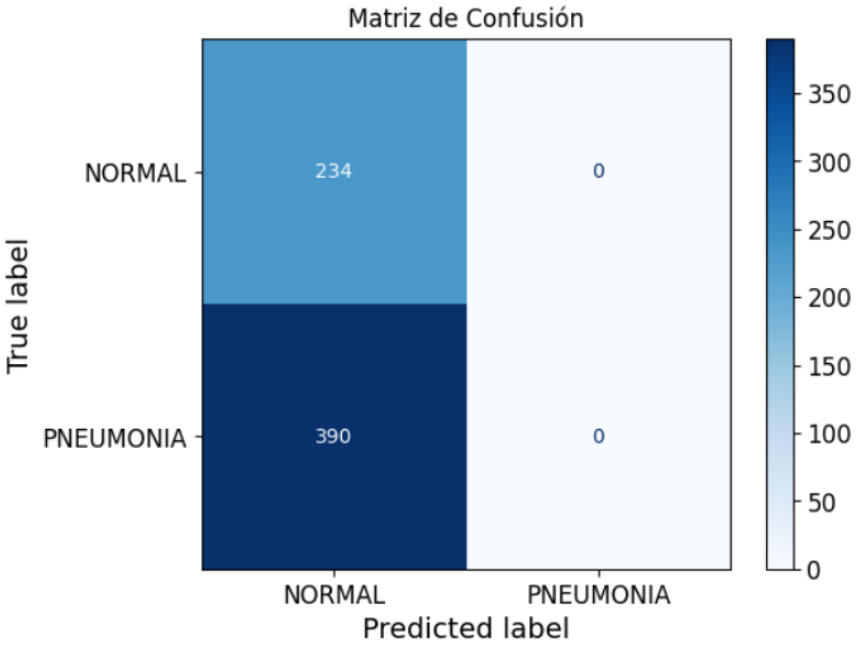
```
early_stopping = EarlyStopping(
            monitor='val_acc',
            patience=patience,
            restore_best_weights=True
        )
```

Resultados.






```
39/39 [==============================] - 54s 1s/step - loss: 1.9059 - acc: 0.6218

test acc :
 0.6217948794364929
```



Se obtuvieron buenos resultados dentro del set de entrenamiento y en el de validación, pero no en el de prueba, esto puede indicar que se está generando un sobreajuste.

Áreas de oportunidad.

Dentro del artículo se hace referencia a que se utilizaron 100 epochs para poder entrenar esta red. Dentro de las limitaciones de mi hardware solo me permitía realizar 8 epochs en un tiempo razonable, probablemente una de las razones por las que no se obtuvo un buen resultado en el set de prueba haya sido ese. De igual manera no genere nuevos datos en el set de entrenamiento debido a las limitaciones que me encontré. Para futuras iteraciones se probara entrenar la red un sistema operativo diferente en el que se pueda tener activo CUDA, esto con el fin de alcanzar más epochs, y poder generar un dataset más grande con las funciones de DataImageGenerator.

Referencias:

- Lee, C. P., & Lim, K. M. (2022). COVID-19 diagnosis on chest radiographs with enhanced deep neural networks. Diagnostics, 12(8), 1828. https://doi.org/10.3390/diagnostics12081828