Chest X-Ray classification for Pediatric Pneumonia

Guilherme De Sequeira Antunes

December 2023

Abstract

Pediatric Pneumonia classification is one of the most important first steps to stop the vast amount of child death every year. Given a dataset of chest x-rays, the aim is to estimate if the neural network can accurately predicted which of the file contains an image of chest x-ray with pneumonia. Using diverse models, the approach used proved to be a work in progress, whereas half of the models didn't perform as expected, the other half had very promising results.

Contents

1 Introduction

Acute respiratory infections are responsible for 20% of the deaths of children under 5 years old[5]. Approximately 150 million new cases of pneumonia occur annually among children younger than 5 years of age worldwide. This accounts for approximately 10-20 million hospitalizations [7].

Pneumonia kills more children than any other infectious disease, about 2000 children every single day, a death every 43 seconds[4]. Almost all of these deaths are preventable. Given the crucially identification of every case on its earliest, with the robust power of deep learning I tried to develop a model to detect pediatric pneumonia from Chest X-ray images.

1.1 Motivation

Pediatric Pneumonia is the leading cause of child mortality worldwide, emphasizing the crucial need for an early prediction and accurate diagnosis. Automated solutions in cooperation with the traditional methods may help with timely insights, improving the speed time and accuracy of the diagnosis potentially saving more lives, allowing healthcare professionals to focus on other complex aspects of patient care. This solution may also provide to enhance the healthcare efficiency meeting the demands of a growing pediatric population, equally coming as a financial sustainability for healthcare providers and furthermore contributing to the non-stop and hopefuly forever evolving medical research and education.

1.2 State Of The Art

Pediatric pneumonia detection has witnessed significant advancements in recent years, driven by the intersection of medical imaging and cutting-edge technologies. Machine Learning and Deep Learning techniques like convolutional neural networks (CNN) is a characterist example. [3] [2] [8]

2 Materials and Methods

2.1 Approach

To establish a robust foundation for our Pediatric Pneumonia detection project, was conducted an exhaustive literature review, exploring diverse methodologies employed in the classification of pediatric chest X-ray images. The investigation encompassed a spectrum of machine learning and deep learning techniques specifically tailored for medical image analysis.

The literature review revealed a multitude of approaches in the realm of detecting respiratory conditions. Noteworthy methodologies covered the analysis of medical images, encompassing chest X-rays.

Acknowledged for their effectiveness, architectures like U-Net demonstrated significant capabilities in accurately classifying chest X-ray images related to Pediatric Pneumonia. Moreover, this investigation highlighted the success of pre-trained models, specifically VGG-16 and ResNet50, in achieving promising results in the classification of pediatric chest X-rays[9]. The research question set was “Which model, among VGG16 and ResNet50, performs better in the classification of chest X-ray images ?”.

Although, in this experiment, the two models refered previously were the main ones used, to complement the project, however, the author decided to enlogate this approach to other models. Changing the research question to “Which model, among VGG16, ResNet50, Inception V3 and EfficientNetB0 performs better in the classification of chest X-ray images ?”

2.2 Dataset

The dataset used for this project was from Kaggle database called ”Pediatric Pneumonia Chest X-ray”.
https://www.kaggle.com/datasets/andrewmvd/pediatric-pneumonia-chest-xray/data.

This dataset contained a total of 5856 images, divided in two directories, namely test directory, containing 234 files of normal image x-rays and 390 files of image x-rays with pneumonia, and train directory containing 1349 files of normal image x-rays and 3883 files of pneumonia x-rays images. Techniques such as data augmentation were used to enhance the robustness and generalization ability of the model, allowing it to learn more effectively and improve its performance. Other techniques, such as random rotation, rescaling, shear range, zoom range, horizontal flip and rotational flip were used for this data.

2.3 Data Preprocessing

The script began with the loading of the Pediatric Pneumonia Chest X-rays dataset using Kaggle API, where the images were resized to each model's needs, so 299 x 299 images in the Inception V3 model, 224 x 244 images in the VGG16 model, in the ResNet50 model and in the EfficientNetB0 model. In order to augment the diversity of the training dataset and to improve the model generalization, some techniques related to data augmentation were implemented, after which were normalized to standardized the pixel intensity, facilitating stable and efficient model training. These steps, collectively, contribute to a more robust dataset, ensuring that the models could learn patterns in a more effective way without being slowed down by variations of image size or intensity.

2.4 Data Augmentation

Data augmentation played a crucial role in the models' ability to generalize patterns in different scenarios. By exposing the model to these augmented images ensures that becomes less sensitive to variations encountered in the different scenarios of the data and able to capture essential features associated with Pediatric Pneumonia.
The techniques implemented were the following:

2.4.1 Random rotation

This techniques applies random rotation in order to simulate variations in the positioning of the patients during an x-ray imaging, ensuring that the model becomes invariant to different orientations and learns features effectively despite the initial alignment of the x-ray. The rotation was up to 20 degrees.

2.4.2 Shear Transformations

Shear transformations replicates potential distortions in x-ray images, helping the model becoming more robust to such distortions in real-life scenarios. The distortion used was up to 20%.

2.4.3 Zooming

Random zooming was applied to the images to simulate different levels of proximity during a x-ray imaging, helping the model to learn to recognize patterns in chest x-rays in regardless of the scale. The zooming applied had a range of 20%.

2.4.4 Horizontal Flip

This technique was employed to create mirror images where in certain instances the patient x-ray may be facing the opposite direction, enriching the model to become indifferent to left or right orientation.

```python
#use a single output node with sigmoid activation
model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Figure 1: ResNet50 layers

```python
#layers for classification on top of EfficientNetB0
model.add(Flatten())
model.add(Dense(24, activation='relu'))
model.add(Dropout(0.5))  # dropout layer for regularization
model.add(Dense(1, activation='sigmoid'))  # layer for binary classification
```

Figure 2: EfficientNetB0 layers similar to the ones use for VGG16 and Inception V3

2.5 Models

Most models used a similar coding style for the definition of the layers, consisting on flattening the multi-dimensional output into a flat vector, a dense layer of 24 units and a rectified linear unit that would add the capacity of learning complex patterns, a dropout layer in order to prevent overfitting and an output layer for binary classification (Figure 2). The only model who didn't quite follow this approach was ResNet50, due to specific characteristics that it carries, so instead of the flatten layer, the Global Average Pooling was used, converts the spatial features into a vector by taking the average of each feature map and with a similar dense layer of one unit and a sigmoid activation function for the binary output (Figure 1).

2.5.1 VGG16

The VGG16 or Visual Geometry Group 16-layer) architecture is a deep convolutional neural network (CNN) designed for image classification. Developed by the Visual Geometry Group at the University of Oxford, VGG16 is renowned for its simplicity and effectiveness[10]. (Figure 3)

2.5.2 ResNet50

ResNet50, short for Residual Network with 50 layers, is a powerful deep convolutional neural network architecture designed to address the challenges of training very deep networks. Developed by Microsoft Research, ResNet50 introduces the concept of residual learning, enabling the training of exceptionally deep neural networks with improved performance and ease of optimization.[11] (Figure 4)

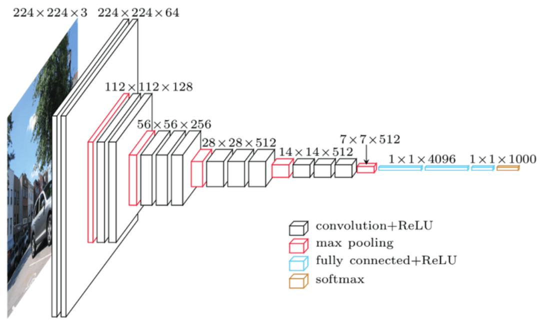

Figure 3: VGG16 model architecture. Comprises 16 layers, consisting mainly of convolutional layers with small 3x3 filters, followed by max-pooling layers for spatial downsampling

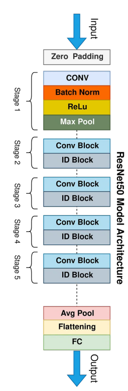

Figure 4: ResNet50 Model Architecture. Consists of 48 convolutional layers along with additional layers for preprocessing and post-processing. It achieved significant success in the ImageNet Large Scale Visual Recognition Challenge, outperforming previous architectures and demonstrating superior accuracy in image classification tasks

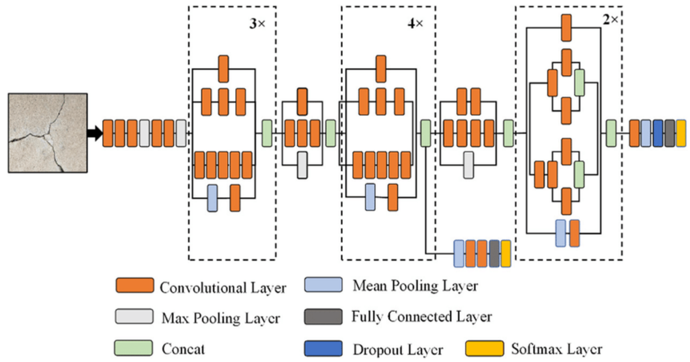

Figure 5: Inception V3 model's architecture. Has a total of 159 layers encompassing both trainable and non-trainable layers in the architecture

2.5.3 Inception V3

InceptionV3, developed by Google, is a deep convolutional neural network architecture designed for image classification and recognition tasks. It is part of the Inception family, which is characterized by its innovative use of inception modules introducing several enhancements to improve accuracy and efficiency[1].(Figure 5)

2.5.4 EfficientNetB0

EfficientNetB0, introduced by Google in 2019, represents a breakthrough in convolutional neural network (CNN) architecture, specifically designed to optimize both model efficiency and performance. It introduces a novel compound scaling method that uniformly scales network width, depth, and resolution, achieving superior results with fewer parameters compared to traditional CNNs [6].(Figure 6)

3 Experiments and Performance

All of the experiments were implemented on Google Collab. The starting date of the project was 28/11/2022 and it was finalized on 22/12/2022. The average time spent for the training of the models, was 133 seconds for each model with 15 epochs. The first step of of our project was exploratory analysis, in order to get familiar with our data, their format and their characteristics. The research question set was "Which model, among VGG16 and ResNet50, performs better in the classification of chest X-ray images ?". In order to enrich the project's content and quality the author decided to add some more models such as Inception V3 and EfficientNetB0, to add a bigger comparison term. The changes made had to do with setting the bar a little lower due to the time frame in which the author found himself working, balanced with the lack of experience in this

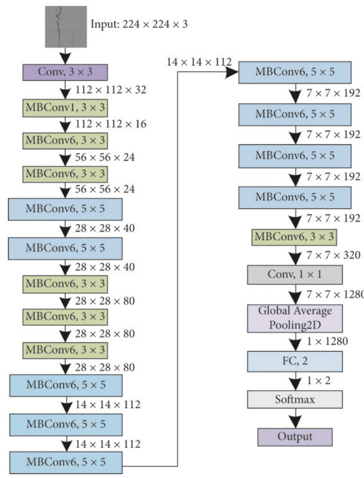

Figure 6: EfficientNetB0 model's architecture. Consists of a total of 314 layers including both trainable and non-trainable parameters in the network architecture

course. Luckily, because the harder you work the luckier you get, the author managed to add some more models affecting the research question to "Which model, among VGG16, ResNet50, Inception V3 and EfficientNetB0, performs better in the classification of chest x-ray images?"

As earlier mentioned in the Data Preprocessing section, several data augmentation techniques were applied. The initial experiment consisted of 25 epochs which was then reduced to 15 due to time related issues. The approach taken to battle the overfitting problems that could and were experienced in the data were handled by a regularization technique denominated by early stopping. The layer units also suffered some changes along the way, ending with a "dense" of 24, which had some impact on the models.

3.1 Performance

After finalizing the experiments the following results were obtained. (Starting from lower accuracy to the highest)

3.2 EfficientNetB0 Performance

EfficientNet starts the results topic with the lowest score. With an accuracy of 62.5% (Figure 8) but a lower loss compared to ResNet50. Making the bold assumption that all of the test cases would be pneumonia 9 didn't got him the highest accuracy in the sheet, despite identifying correctly all the pneumonia cases (Figure 10). It is also important to note that this model stands out for being the one with the lowest training, only four (4) epochs were run in this

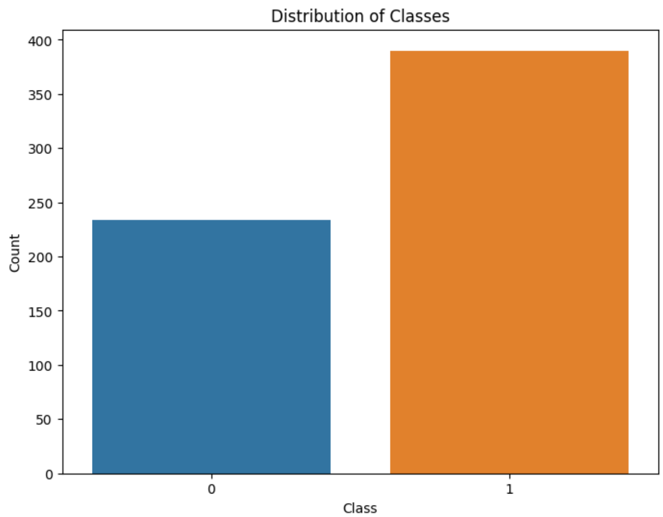

Figure 7: The initial dataset counted with 390 cases of pneumonia and 234 normal cases

code due to the early stopping feature which means that the model stopped improving after four epochs avoiding future degradation. Without the feature the results may have been worse.

3.3 ResNet50 Performance

ResNet50 scored 63,78% of test accuracy (Figure 11) and a loss of 2.515. ResNet50 came shy on the detection phase. In the bold assumption of the majority (616/624 total x-rays) of the cases to be considered as pneumonia (Figure 12). ResNet50, while dectecting nearly almost of the pneumonia cases, wasn't able to differentiate properly the normal x-rays from pneumonia, diagnosing the normal patients with pneumonia (Figure 13) and belongs to the half that ran all the 15 eppochs.

3.4 Inception V3 Performance

InceptionV3 had a test accuracy of 87,18% (Figure 14). The test loss was 0,3525. Nearly the same cases of normal and pneumonia were detected by the Inception V3 model (Figure 15) as the dataset (Figure 7). Altough, this accuracy maybe misleading, not containing the difference between true and false positives. Out of the 390 pneumonia cases, managed to predict 266 correctly and 73 normal cases out of 234 cases (Figure 16), all of this running only 6 epochs before the early stopping feature intervened. Came as an unexpected guest and finished second best.

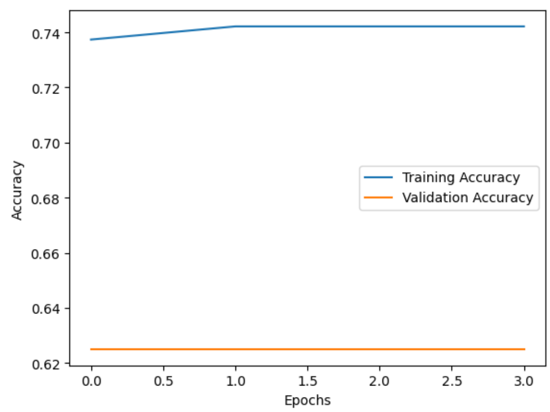

Figure 8: EfficietnNetB0 had an accuracy of 62.5% and a loss of o.6616

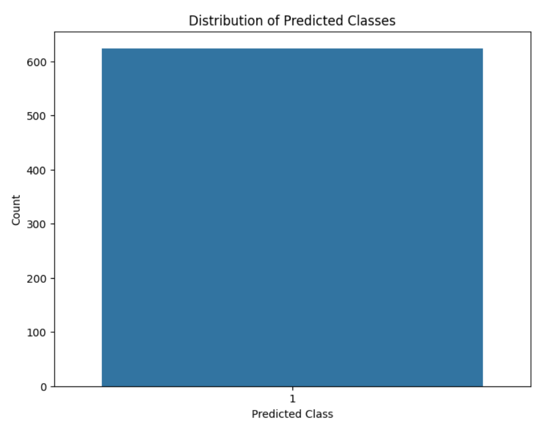

Figure 9: EfficietnNetB0 predicted that all of the cases would be pneumonia

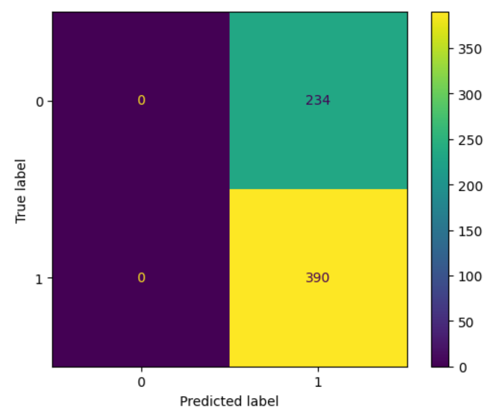

Figure 10: EfficietnNetB0 Had an 100% rate labeling the pneumonia cases, on the other hand a 0% rate aknowledging the normal ones.

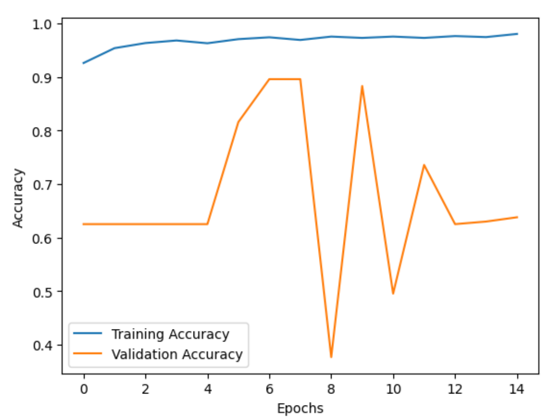

Figure 11: ResNet50 scored 63,78% of test accuracy and a loss of 2.515

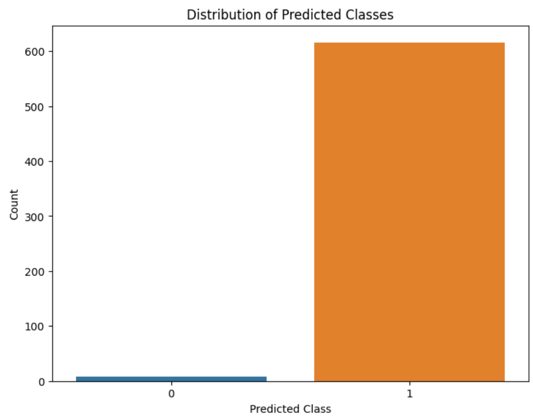

Figure 12: ResNet50 came shy on the detection phase. In the bold assumption of the majority (616/624 total x-rays) of the cases to be considered as pneumonia

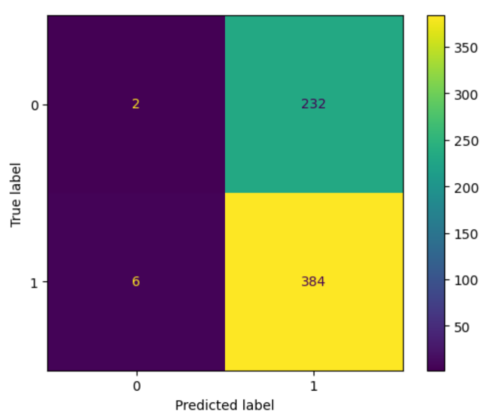

Figure 13: ResNet50, While dectecting nearly almost of the pneumonia cases, wasn't able to differentiate properly the normal x-rays from pneumonia, diagnosing the normal patients with pneumonia.

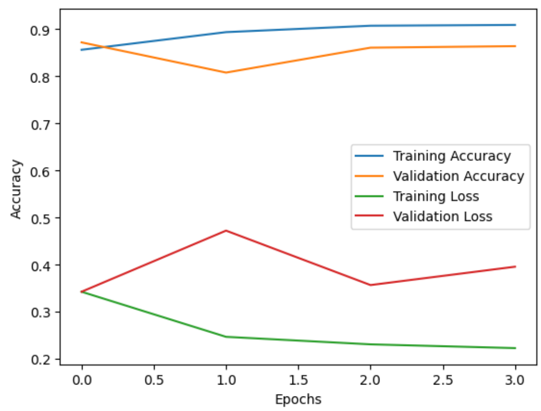

Figure 14: InceptionV3 had a test accuracy of 87,18%. The test loss was 0.3525

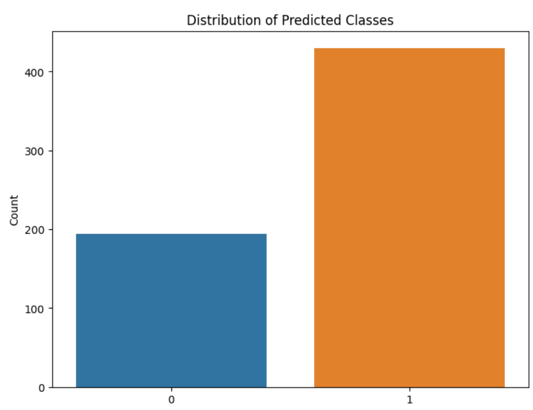

Figure 15: Nearly the same cases of normal and pneumonia were detected by the Inception V3 model. Altough, this graphic maybe misleading, not containing the difference between true and false positives

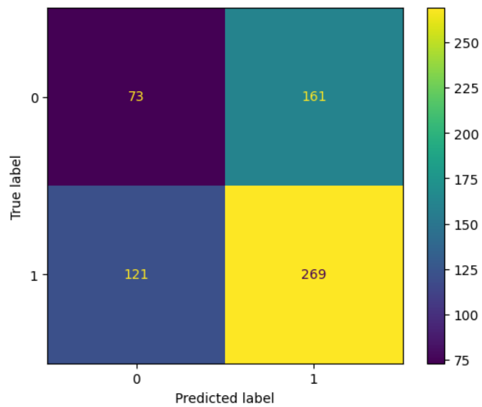

Figure 16: Out of the 390 pneumonia cases, Inception V3 managed to predict 266 correctly and 73 normal cases out of 234

3.5 VGG16 Performance

The model VGG16 secured the first place on this list with 91.19% accuracy (Figure 17) and scoring also the lowest loss. Not only the accuracy was incredible as it also managed to exceed the opponents on all the other graphics presented. Predicting numbers very similar (Figure 18) to the ones presented on the test directory of the dataset (Figure 7). Having also the most balanced accuracy on the true label test (Figure 13). Run all the 15 epochs.

4 Conclusion

4.1 Conclusion

Despite having a high accuracy in half of the models, the other half came short of what was expected. We can also add that there's the alarming fact that none of the models was able to accurately separate true positive cases of pediatric pneumonia from true negatives, which lowers the real accuracy of the models, which suggests further investigation as well as a more directed fine tuning and time to adjust each of these models to fulfill their top level capacity. This can also be related to the slight dataset imbalance, despite not having a huge difference between classes, it may be significant, leading the models to evaluate the files in a biased way. Some models, for instance, EfficientNetB0 may not be suited at all for this type of experiments, as on the other hand ResNet50 with some more runs and some tuning adjustments in the layers may have achieved a slight better performance. Given the results, the future seems to be promising to medical fields if the right amount of time and dedication is applied, since some models actually came quite close to an amount of true label predictions, the accuracy could be further improved and give the models a real-life use for

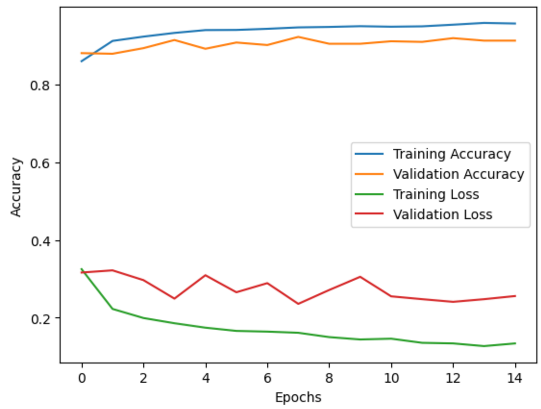

Figure 17: VGG16 scored a stunning 91.19% on test accuracy with the lowest loss of 0.256

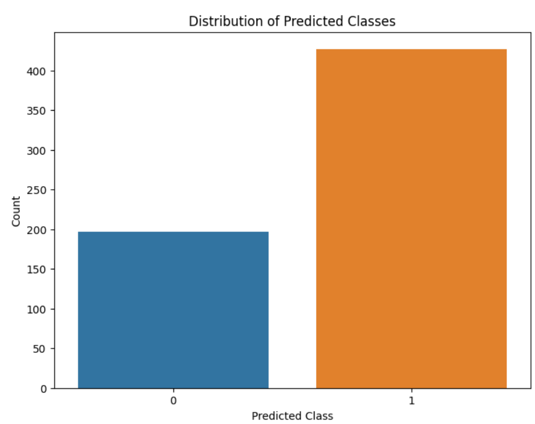

Figure 18: VGG16 detected almost the same cases as the ones presented by the dataset

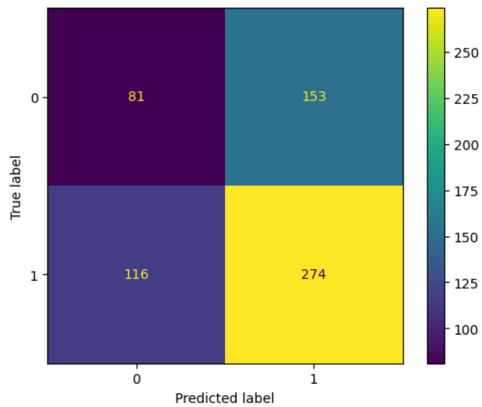

Figure 19: Out of the 390 pneumonia cases, VGG16 managed to predict 274 correctly and 81 normal cases out of 234

pediatric pneumonia cases.

5 Discussion

5.1 Starting Point

When I was transfered for this elective, I had no knowledge of biomedical image analysis and knew very little about deep learning for image and video processing. Following this course gave me the chance to learn new theoretical concepts and get familiar with state of the art techniques. Regarding the programming part, my programming skills were also in beginning steps, but all this concepts helped improving them much more, adding to the fact of all the knowledge acquired about deep learning.

5.2 Issues Faced

During the realization of the project I faced some challenges, I started by running the code on google colab but some issue with the GPUs wouldn't allow me to run the code properly taking nearly 2 hours for each epoch to be processed, time when I was experiencing with 25 epochs, I then switched to Jupyter Notebook where I faced the same challenges but where the running time of each epoch was less than the previous with also less epochs, around 10 epochs with about 1h52mins each. I managed to fix this issues and to run the code properly on google colab, adjusting the epochs at the begining to 25 and later reduce to 15, since I could only run one of the codes at the time. No simultaneous code run was done during this entire project. The second main challenge I faced was the overfitting of the data that was controlled by the techniques mentioned above, such as data augmentation and early stopping. I also had to redo most of the code for two of the pre-models used due to some unsaving issue related again with my google colab in a late stage of the project. Before the the main challenge there was another one that came across, the limit of GPU usage of google colab, it didn't allowed me to run anymore code with the GPU usage, making it impossible to run the final adjustments made to every single code. Another challenging requirement was finding the right dense to add, since the evaluation accuracy spiked in very different epochs for the models. However, the main challenge that remains to be be refered was joining this course when it was in an ending phase, when I got transferred to this course the final presentation schedule was already settled for few weeks after, leaving me a bit anxious about the all situation, it required a lot of dedication but made me make this project and experiment that I would probably would not believe I could achieve if someone told me I would in the beginning.

5.3 Outside Help

For the challenges that I faced I tried to make the most out of all the available sources I had in hand, from colleagues who had previously taken the course, to colleagues who were currently taking the course to AI languages such as ChatGPT. Despite of none of them had a major impact but small advises, I cannot not credit this to them. I faced some coding errors that I could not understand and for my own good I searched the internet, but some errors had to be explained by the language model used, others where helped from friends, whom by brainstorming the way I wanted to do things, over small encounters that we usually have. Even though, not being sure if I would have found a solution for all of them, I know that I would have found a solution for most of them, unfortunately given the short time that I had to solve this issues due to the late transfer to the course, I tried to optimize it collecting information from my colleagues. I can however take credit for the most code written, not entirely because of some adaptions that had to be done, acknowledging the fact that I am using pre-trained models and not models entirely made by me, therefore, would not be concise or truthful to the events collecting the credits entirely to myself. Despite all of this, however, I can assure that I spent many many hours that can certainly be translated to days or perhaps weeks trying to develop these models and the outside consultation was not deeply founded to even consider not taking credit for my own work.

5.4 What I learnt

Implementing this project gave me useful insights into machine learning methodologies like transfer learning, I learnt about data augmentation and how it can save the day, I also realized what it means overfitting of a model and how to solve it and I studied about various evaluation techniques like confusion matrix and cross validation. However, the most important thing that I learnt was how interesting that this field is and left me with a truly amazing question. How can I this knowledge to something that startles me?

6 References

References

[1] Luqman Ali, Fady Alnajjar, Hamad Al Jassmi, Munkhjargal Gocho, Wasif Khan, and M Adel Serhani. Performance evaluation of deep cnn-based crack detection and localization techniques for concrete structures. *Sensors*, 21(5):1688, 2021.

[2] Roaa Alsharif, Yazan Al-Issa, Ali Mohammad Alqudah, Isam Abu Qasmieh, Wan Azani Mustafa, and Hiam Alquran. Pneumonianet: Automated detection and classification of pediatric pneumonia using chest x-ray images and cnn approach. *Electronics*, 10(23):2949, 2021.

[3] Enes Ayan, Bergen Karabulut, and Halil Murat Ünver. Diagnosis of pediatric pneumonia with ensemble of deep convolutional neural networks in chest x-ray images. *Arabian Journal for Science and Engineering*, pages 1–17, 2022.

[4] Colleen Murray and Holly Newby. Data resource profile: United nations children's fund (unicef). *International journal of epidemiology*, 41(6):1595–1601, 2012.

[5] World Health Organization et al. Indicator metadata registry details. *World Health Organization. https://www. who. int/data/gho/indicator-metadata-registry/imr-details/158*, 2022.

[6] Asra Momeni Pour, Hadi Seyedarabi, Seyed Hassan Abbasi Jahromi, and Alireza Javadzadeh. Automatic detection and monitoring of diabetic retinopathy using efficient convolutional neural networks and contrast limited adaptive histogram equalization. *IEEE Access*, 8:136668–136673, 2020.

[7] Igor Rudan, Lana Tomaskovic, Cynthia Boschi-Pinto, and Harry Campbell. Global estimate of the incidence of clinical pneumonia among children under five years of age. *Bulletin of the World Health Organization*, 82(12):895–903, 2004.

[8] Yash S Saboo, Saarthak Kapse, and Prateek Prasanna. Convolutional neural networks (cnns) for pneumonia classification on pediatric chest radiographs. *Cureus*, 15(8), 2023.

[9] Jianshe Shi, Yuguang Ye, Daxin Zhu, Lianta Su, Yifeng Huang, and Jianlong Huang. Comparative analysis of pulmonary nodules segmentation using multiscale residual u-net and fuzzy c-means clustering. *Computer Methods and Programs in Biomedicine*, 209:106332, 2021.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. *Dive into deep learning.* Cambridge University Press, 2023.