# Introduction

Pneumonia is an infection that inflames the air sacs in one or both lungs. The air sacs may fill with fluid or pus. causing cough with phlegm or pus, fever, chills, and difficulty breathing. A variety of organisms, including bacteria, viruses, and fungi, can cause pneumonia. Pneumonia can range in seriousness from mild to life-threatening. It is most serious for infants and young children, people older than age 65, and people with health problems or weakened immune systems. Each year, pneumonia affects about 450 million people globally (7% of the population) and results in about 4 million deaths. With the introduction of antibiotics and vaccines in the 20th century, survival has greatly improved. Nevertheless, pneumonia remains a leading cause of death in developing countries, and also among the very old, the very young, and the chronically ill. Pneumonia often shortens the period of suffering among those already close to death. Diagnosis is usually made based on recent health history (such as surgery, a cold, or travel exposures) and the extent of the illness. Pneumonia is typically diagnosed based on a combination of physical signs and often a chest X-ray. Chest X-ray (CXR) is a projection radiograph of the chest used to diagnose conditions affecting the chest, its contents, and nearby structures. Chest radiographs are the most common film taken in medicine. This project aims to diagnose whether someone has been infected with pneumonia based on the chest X-ray of the person. And we also want to classify whether the pneumonia is caused by bacteria or viruses based on the image.

# Problem Statement

The objective of our project is to analyze the patient’s chest X-ray and identify whether the patient has pneumonia or not and classify the type of the pneumonia based on the image using CNN and other deep learning method.

# Challenges

**Imbalance Dataset that may lead to Biased Data**

The dataset that we have is imbalance. With the number of positive (Pneumonia) data having more samples than normal data. This may be a problem and may introduce bias in the classification. As Convolutional Neural Network works best when training data is balance, this problem may hinder the potential of the models that are proposed. We handle this problem be assigning class weight for each class. Detailed steps will be explained further.

**There might be not enough data for creating a good model**

As we know, models that are based on Neural Network is data hungry. Meaning that they have to be fed many data so that we can get good results. The dataset contains total of around 5000 images. A model that relies only on this data might be not enough to create a good classifier because of lack of data. This problem might be affecting the overall performance of the model.

## RELATED WORKS

We found some of projects that used similar dataset. And they have some of relation to this project.

*Paper 1*

**Saraiva et al.** provide a technology approach and methodology for automatically and quickly classifying Pneumonia. The author chose and developed a classification of patient chest X-ray pictures that can be used to determine whether the patient has pneumonia or not. The classification stage is divided into two substages, the first of which is conducted by a Convolutional Neural Network and the second by a Multilayer Perceptron. The author believes that the method covered ensures comprehensive image recognition coverage.

Other works classify the Pneumonia with different model, different algorithms, and from different sources. **Saraiva et al.** utilize two learning models, Convolutional Neural Network and Multilayer Perceptron. The input of the test images separated by the k-fold algorithm is conducted in the phase of predicting test data, and the accuracy is recorded. The technique is performed 5 times, each time altering the test and training photos after calculating the k-fold. The author also employed the dropout approach in the fully connected layer of the Convolutional Neural Network to shorten training time and avoid overfitting. At each training iteration, a specific number of neurons from a layer are randomly removed and re-added to the next iteration.

*Paper 2*

**Puneet Gupta** believes that deep learning has the ability to achieve substantial characteristics in image classification tasks and to deliver medically better versions in image analysis. The author attempted to create optimum deep learning CNN models that can detect and categorize pneumonia conditions efficiently. The study consists of optimal CNN models as well as experimental examination of each model for the detection and classification of pneumonia illnesses. Deep Learning features are collected from images and used for pneumonia classification in x-rays. For accuracy comparison, the author employed both the transfer learning approach and CNN from scratch.

**Puneet Gupta** used the Convolutional Neural Networks model and the Transfer Learning principle. Transfer Learning allows the model to save to the original parameter values of a previously trained model, resulting in an effective score without the use of intensive processing effort. The author employed the transfer learning models VGG16 and VGG19. In addition, the author used a customized Convolutional Neural Network with a Max-Pooling layer. By selecting only the maximum values from the kernel matrix, this layer reduced the values to half of their original value.

*Paper 3*

**Xiang yu et al.** thought that one common problem in the detection of pneumonia by CAD systems is the lack of large-scale public datasets. While many works reported high performance, the numbers of images in the testing datasets are too few to form convincing results. Also, many of the works simply transferred the state-of-the-art CNNs to the classification task without further exploration of structural optimization while the reported performance heavily relies on the pre-determined parameters and hardware. To solve the stated issues, the author proposed CGNet framework to detect pneumonia based on large scale public X-ray datasets.

Unlike traditional ML methods that utilize manually designed algorithms for ad hoc tasks, **Xiang yu et al.** use deep learning-based algorithms that have a higher superiority on generalization. Feature extraction plays a key role in the classification task, which directly determines the overall performance of the following classifiers. Primarily, the author implements feature extraction by deploying transfer learning technique.

*Paper 4*

**Fernandes et al.** aims to develop a method for detecting and diagnosing pneumonia through X-ray images. The research aims to demonstrate the feasibility of building an automatically estimated convolutional neural network (CNN) topology that performs satisfactorily within the problem.

**Fernandes et al.** believes that estimating a convolutional neural network’s architecture is a challenging task, but it provides a lightweight, robust, and comprehensive method for solving a problem when done correctly. Because it is an exhaustive task, the literature recommends that basic or primitive information must be shared from any previously completed detection or recognition task. Thus, knowledge transfer from a pre-trained network is used in the author’s work as a starting point for estimating the proposed architecture.

*Paper 5*

**Rahman et al.** believe that the computer-aided diagnostic tool can significantly help the radiologist to take more clinically useful images and to identify pneumonia with its type immediately after acquisition. Fast classification will open up other avenues of application for this CAD tool, more particularly in the airport screening of pneumonia patients.

**Rahman et al.** present a deep-CNN-based transfer learning approach for the automatic detection of pneumonia and its classes. Four different popular CNN-based deep learning algorithms were trained and tested for classifying normal and pneumonia patients using chest x-ray images. The author used AlexNet, ResNet, DenseNet, and SqueezeNet as the transfer learning of their models.

The papers focused on how to diagnose Pneumonia based on Chest X-Ray Images and classify the Pneumonia type of it. The papers cover issues that is similar for our proposed study. Not only by technical means, but also from methodology perspective. In our case, diagnose Pneumonia based on chest X-Ray images using transfer learning. And classifying Pneumonia type based on the images. All of these papers help us in getting the general idea of the project. They give many solutions to our problem. Like choosing optimizer, data augmentation, data modeling, choosing transfer learning model, and many more. These related papers are our guidance for assembling the baseline models.

## IMPORTANCE AND IMPACTS

Our aim is to classify Chest X-Ray images to three types of classifications. We believe that this kind of project will help thousands of people who are feeling that they are in symptoms of pneumonia. This project can cause people to reduce their budget of seeing a doctor. People can also save their time from scheduling an appointment. They can straight get a timely treatment based on the classification that they got.

# Data Collection

**DATA DESCRIPTION**

We are using data that is provided from Kaggle. The dataset contains images of chest X-Ray that is focused on pneumonia diseases. The dataset itself was obtained from Mendeley that the original dataset contains the chest X-Ray data and Optical Coherence Tomography (OCT) for classification. But for this project, we will be using the chest X-Ray data only. The data was labeled as Normal and Pneumonia. And for the Pneumonia images, they are categorized again as Bacterial or Viral pneumonia. Chest X-ray images were selected from retrospective cohorts of pediatric patients of one to five years old from Guangzhou Women and Children’s Medical Center, Guangzhou. All chest X-ray imaging was performed as part of patients’ routine clinical care. For the analysis of chest x-ray images, all chest radiographs were initially screened for quality control by removing all low quality or unreadable scans. The diagnoses for the images were then graded by two expert physicians before being cleared for training the AI system. In order to account for any grading errors, the evaluation set was also checked by a third expert. Total images that are available in the dataset is 5856 images. The images pixels are around 1200 x 1000 pixels. The distribution of the images is not balanced with the bacterial pneumonia images have the most number out of the three categories. Below is example of the images of the normal, bacterial pneumonia, and viral pneumonia.

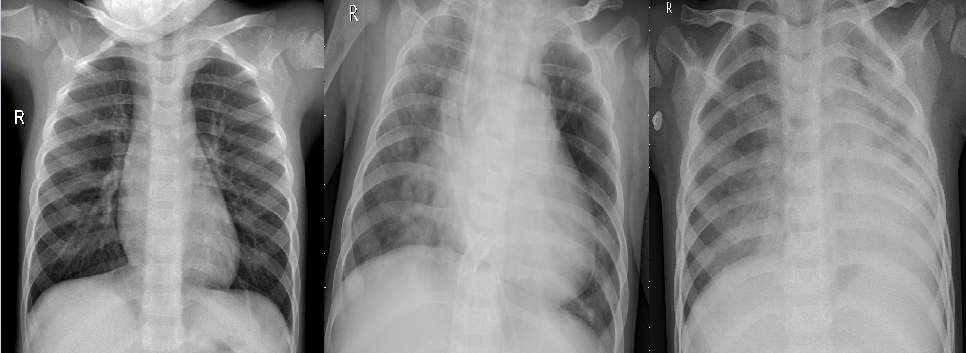


Figure 1: Example of Chest X-Ray, (left) Normal, (center) Bacterial Pneumonia, (right) Viral Pneumonia

**CLASS DISTRIBUTION**

We tried to see if the dataset contains imbalanced data. We then plot the class count in a count plot. Following is the result of the plotting.

**Chart, bar chart

Description automatically generated**

Figure 2: Count Plot of Classes

**A screenshot of a computer

Description automatically generated with medium confidence**

Figure 3: Count of Classes

We can see that the class distribution is not so balanced. We can see that the Bacterial Pneumonia class is dominating the dataset with staggering count of 2780. Followed by Normal and Viral classes with count of 1583 and 1493. We have to balance this data by any means to get the best calculations of the model.

**IMAGE DIMENSIONS**

We tried to know what is the best shape that we can apply to the dataset in order to get the model to process the image as efficient as possible. We created a joint plot to see the sizes of the image that are found in the dataset. Following is the result of the joint plot.

**Chart, scatter chart

Description automatically generated**

Figure 4: Joint Plot of Images Dimension

We can see that the images in the dataset ranging from < 500px to > 2000px. The majority of image dimensions fell in ~1500x~1000. But if we are using this shape, the model will be very heavy to compute and will consume many resources. So, we will look another way to determine the best shape to apply to the data. We then tried to see the pixel intensity of an image to see the average pixel distribution in an image. We can see how the pixel varied in one size and get the final shape to apply to the data. Following is the result of the pixel intensity plot.

**Chart, histogram

Description automatically generated**

Figure 5: Pixel intensity plot of image

We can see that the Pixel intensity mean is 156 with standard deviation of 47. We can also see in the plot that the highest pixel intensity is in range of 180px. We can see that the high end of the plot is in 200px. So, we will try to use this shape to be applied to our data.

# Data Preprocessing

**DATA IMBALANCE**

We know from the previous plot, our data is in imbalanced state because of the bacterial class that is dominating in the data. We use class weight method to assign individual weight for each of the classes. This step helps us to gain nonbiased weight towards the class that is the majority in the dataset. We get the weight by dividing the class count to the total of images. Following is the result of class weight for our image data.



Figure 6: Class Weight for each class

**DATA AUGMENTATION**

We use data augmentation in the image flow to see if we can classify image in different shape. We use ImageDataGenerator from Keras. This step is crucial because this step can help the model to get the best weight out of an image. This step can also make sure that our model can handle any kind of image shape that will increase the robustness of our model. This step can clarify and enhance the image further because we can augment the data any way we want. From zooming to rotation. Keras ImageDataGenerator can handle all of those tasks. For our model, we used following parameters to our image for our preprocessing.

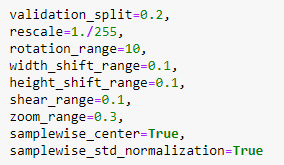


Figure 7: Image Augmentation Parameters

Parameters used:

Validation split = Fraction of images reserved for validation

Rescale = rescaling factor.

Rotation\_range = Degree range for random rotations.

Width\_shift\_range = fraction of total width

Height\_shift\_range = fraction of total height

Shear\_range = Shear angle in counter-clockwise direction in degrees

Zoom\_range = Range for random zoom.

samplewise\_center = Set each sample mean to 0.

samplewise\_std\_normalization = Divide each input by its std.

And following is the result of images after preprocessing.

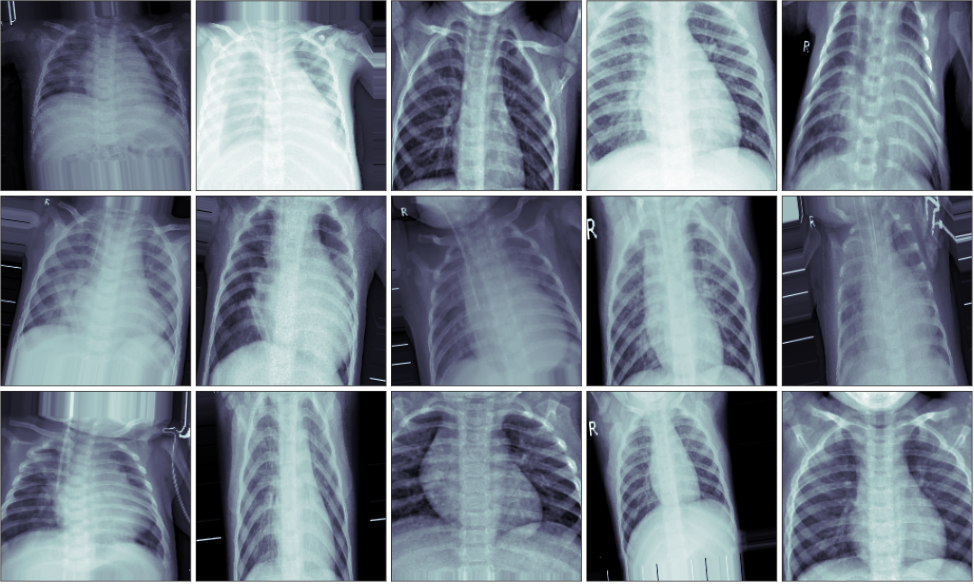


Figure 8: Images after Preprocessing

We also tried to use image data that is not augmented. So, we tested the image as it is. We only rescale them to get the array as normalized as possible. Here are the sample images that we used that are not augmented and still as it is in the dataset.

**A picture containing text, different

Description automatically generated**

Figure 9: Normalized non-Augmented data image

After performing all of these preprocessing steps. We are ready to use these images to the next step. That is Modelling the Neural Network.

# Methodology

**MODELS USED**

**Convolutional Neural Network (CNN)**

Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. Convolutional Neural Network is a specialized neural network for processing data that has an input shape like a 2D matrix like images. CNN's are typically used for image detection and classification. When there is problem that involves image identification, ConvNets is usually used and has become the standard solution to deal with the identification.

**Transfer Learning**

We are also using Transfer Learning Technique as the approach to the identification. Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. By applying transfer learning to a new task, one can achieve significantly higher performance than training with only a small amount of data. Transfer learning is so common that it is rare to train a model for an image or natural language processing-related tasks from scratch. Transfer learning has the benefit of decreasing the training time for a neural network model and can result in lower generalization error.

Below is the Transfer Learning Architecture that we used for the images.

*VGG16*

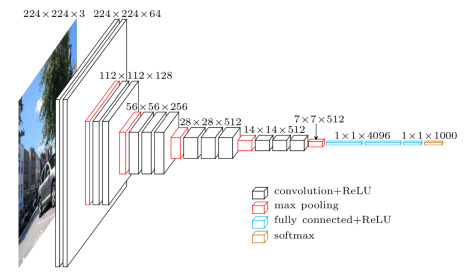
**

Figure 10: VGG Architecture

VGG16 is a type of CNN (Convolutional Neural Network) that is considered to be one of the best computer vision models to date. The creators of this model evaluated the networks and increased the depth using an architecture with very small (3×3) convolution filters, which showed a significant improvement on the prior-art configurations. They pushed the depth to 16–19 weight layers making it approximate 138 trainable parameters. Most unique thing about VGG16 is that instead of having a large number of hyperparameter they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and MaxPool layer of 2x2 filter of stride 2. VGG16 is object detection and classification algorithm which can classify 1000 images of 1000 different categories with 92.7% accuracy. It is one of the popular algorithms for image classification and is easy to use with transfer learning.

*DenseNet*

Diagram, engineering drawing

Description automatically generated

Figure 11: DenseNet Architecture

DenseNet is one of the new discoveries in neural networks for visual object recognition. DenseNet is quite like ResNet with some fundamental differences. ResNet uses an additive method (+) that merges the previous layer (identity) with the future layer, whereas DenseNet concatenates (.) the output of the previous layer with the future layer. DenseNet was developed specifically to improve the declined accuracy caused by the vanishing gradient in high-level neural networks. In simpler terms, due to the longer path between the input layer and the output layer, the information vanishes before reaching its destination.

*Xception*

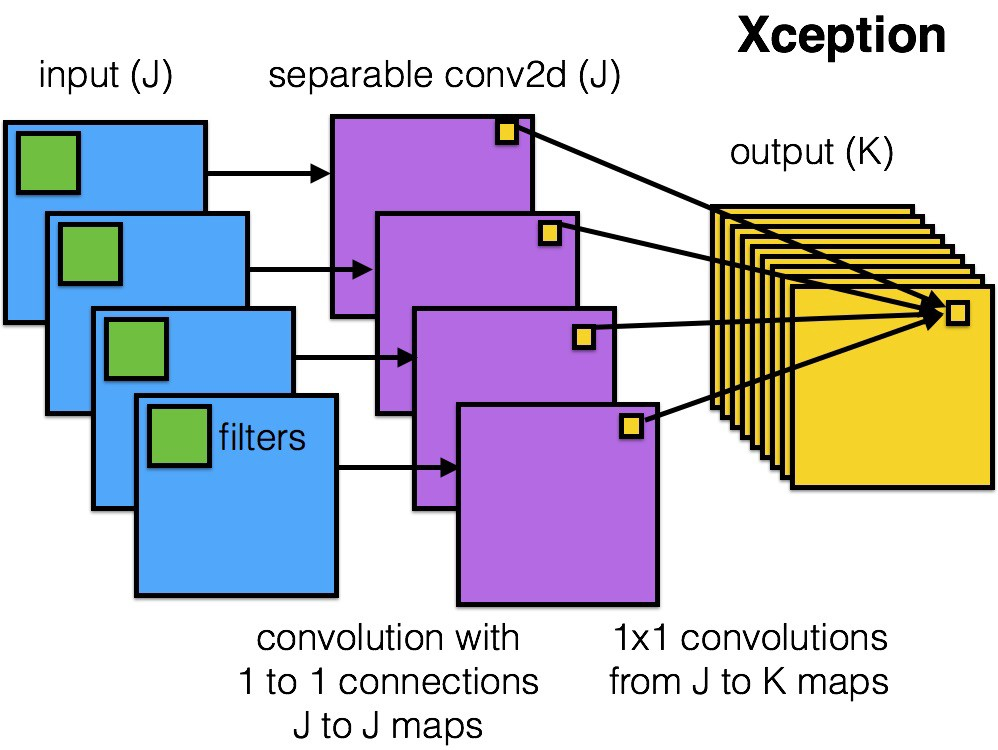


Figure 12: Xception Architecture

Xception is a deep convolutional neural network architecture that involves Depth wise Separable Convolutions. Xception stands for “extreme inception”, it takes the principles of Inception to an extreme. In Inception, 1x1 convolutions were used to compress the original input, and from each of those input spaces we used different type of filters on each of the depth space. Xception just reverses this step. Instead, it first applies the filters on each of the depth map and then finally compresses the input space using 1X1 convolution by applying it across the depth. This method is almost identical to a depth wise separable convolution, an operation that has been used in neural network design as early as 2014. There is one more difference between Inception and Xception. The presence or absence of a non-linearity after the first operation. In Inception model, both operations are followed by a ReLU non-linearity, however Xception doesn’t introduce any non-linearity.

*ResNet*

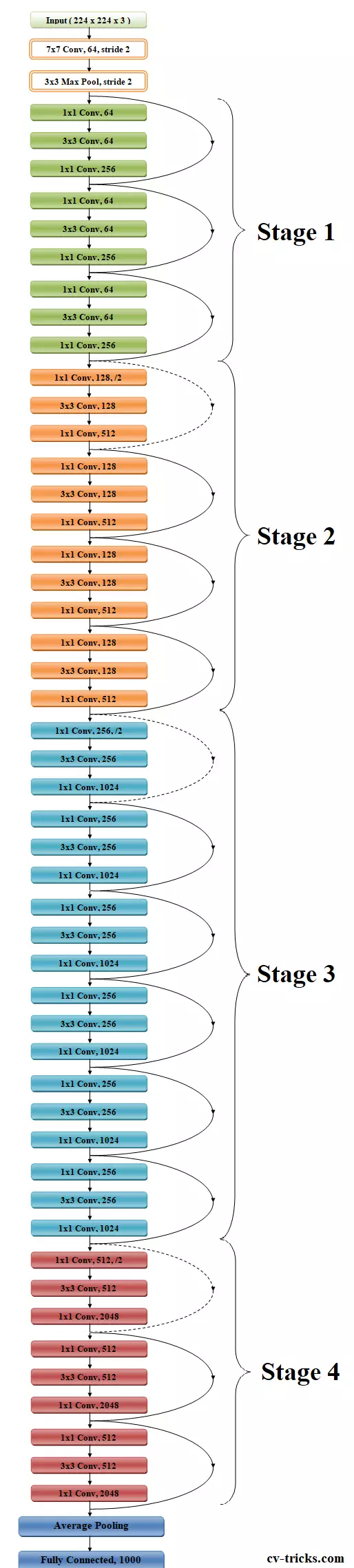


Figure 13: Resnet Architecture

ResNet, short for Residual Network is a specific type of neural network that was introduced in 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. Mostly in order to solve a complex problem, it stacks some additional layers in the Deep Neural Networks which results in improved accuracy and performance. The intuition behind adding more layers is that these layers progressively learn more complex features. For example, in case of recognizing images, the first layer may learn to detect edges, the second layer may learn to identify textures and similarly the third layer can learn to detect objects and so on. But it has been found that there is a maximum threshold for depth with the traditional Convolutional neural network model. The skip connections in ResNet solve the problem of vanishing gradient in deep neural networks by allowing this alternate shortcut path for the gradient to flow through. The other way that these connections help is by allowing the model to learn the identity functions which ensures that the higher layer will perform at least as good as the lower layer, and not worse.

*Inception*

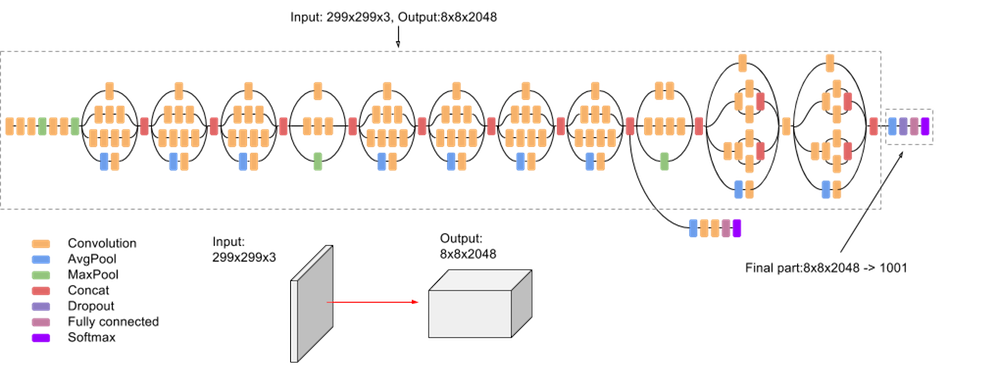
**

Figure 14: Inception Architecture

InceptionV3 is an image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years. The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concatenations, dropouts, and fully connected layers. Batch normalization is used extensively throughout the model and applied to activation inputs. Loss is computed using Softmax.

**Neural Network Architecture**

*Baseline CNN architecture*

*Table

Description automatically generated with medium confidence*

Figure 15: Baseline CNN architecture

As we can see in figure 15, this is the architecture of our Baseline CNN model. We can see that the total trainable parameters that are available is ~7 million. We can see that there are total of 19 layers for the baseline model. We can also see the input layer; the input shape is the same as the image size that we assigned before. And for the output layer, 3 neurons are for the total class we want to predict. The layers that are used for the baseline model are:

* Conv2D

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs.

* MaxPooling2D

Downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool\_size) for each channel of the input.

* Flatten

Layer to flattens the input to 1 dimensional shape.

* Dropout

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.

* Dense

Regular hidden densely connected layer.

*Transfer Learning Model Architecture*

*Table

Description automatically generated*

Figure 16: Transfer Learning model architecture

We can see in figure 16, this is the architecture of our Transfer Leaning model. We can see that despite the total parameters is ~23 million, the total trainable parameters are only ~1 million. This is because we froze the base functional layer so that the functional layer cannot be trained in our model. We extend the model by adding more hidden layer after getting the output from pooling layer. We can see that there are total of 15 layers for the transfer learning model. The layers that are used for the transfer learning model are:

* Functional base layer

Base layer from the transfer learning step. The ones that we use are VGG16, DenseNet, ResNet, InceptionV3, Xception

* GlobalAveragePooling2D

Global average pooling operation for spatial data.

* Batch\_Normalization

Layer that normalizes its inputs. Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

Dropout

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.

Dense

Regular hidden densely connected layer.

**Activation Function**

*Rectified Linear Unit (ReLU)*

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. reason why ReLU works so well is when there are enough of them, they can approximate any function just as well as other activation functions like sigmoid or tanh. We use ReLU for all the hidden dense layers in the models.

*SoftMax*

The softmax function converts a vector of K real values into a vector of K real values that total to one. The softmax translates input values that are positive, negative, zero, or higher than one into values between 0 and 1, letting them to be evaluated as probabilities. If one of the inputs is small or negative, the softmax converts it to a small probability; if an input is high, it converts it to a large probability, but it always remains between 0 and 1. We use softmax for all the output layer in the models because we have 3 classes to predict and we want the output to appear as the probabilistic value.

**Cost/Loss function**

*Categorical Cross Entropy*

Cross-entropy is a measure of the difference between two probability distributions for a given random variable or set of events. Entropy is the number of bits required to transmit a randomly selected event from a probability distribution. A skewed distribution has a low entropy, whereas a distribution where events have equal probability has a larger entropy. Cross-entropy is the default loss function to use for multi-class classification problems. It is common for categorical cross entropy to be used when dealing with multiclass problem. Thus, we apply this as our loss function.

**Optimizer used**

*Adam*

The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. Many believe that Adam is the best optimizer because Adam takes benefits from two optimizers: RMSprop and AdaGrad. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance). Thus, we use this optimizer for our models.

*RMSprop*

RMSProp, is an extension of gradient descent and the AdaGrad version of gradient descent that uses a decaying average of partial gradients in the adaptation of the step size for each parameter. The use of a decaying moving average allows the algorithm to forget early gradients and focus on the most recently observed partial gradients seen during the progress of the search, overcoming the limitation of AdaGrad. RMSProp is designed to accelerate the optimization process, e.g., decrease the number of function evaluations required to reach the optima, or to improve the capability of the optimization algorithm, e.g., result in a better final result. We want to compare RMSprop with Adam. To see which of these two optimizers that works best for our image data.

**Early Stopping**

Early Stopping literally means stop training when a monitored metric has stopped improving. We use this to prevent the model to have decreased performance. We monitor it through validation accuracy. With set patience of 3 epochs. We also set to restore best weights back if the latest weights are worse than the best value.

**Training Parameters**

Epoch: 20 Steps

Batch Size: total number of images // 64

Validation data: 20% split from training data

Learning Rate (for optimizers): 0.0001

# Results and INterpretation

**RESULTS OF THE BEST AND WORST MODEL**

**Table of Comparison of time and accuracy**

**Table

Description automatically generated**

Figure 17: Table of Comparison between models

We created a comparison table to see the performances for all the models in terms of precision, recall, f1, accuracy, ROC-AUC score, and training time. In figure 17, we can see the results from various models that we have tested. We tested total of 14 models. In which they are differentiated from the optimizer that we use. Which is Adam and RMSprop. Typically, we are interested to see the F1 and accuracy score when we want to evaluate classifier model.

In terms of macro average F1 score, we can see that Baseline CNN with non-augmented images and Adam optimizer has the highest score among all models with 77%. And the VGG with RMSprop optimizer has the lowest score with 54%. We can conclude that in terms of balance between precision and recall, VGG RMSprop is the worst of all the models.

Next metric that we want to see is the accuracy score. Again, we can see that Baseline CNN with non-augmented images and Adam optimizer has the best accuracy out of all models with 80% accuracy. And Inception with Adam optimizer being the worst model with accuracy of 66%. We can conclude that based on the comparison tables, it is better to use RMSprop for Inception model rather than Adam for the data that we use because the accuracy is higher by 7%.

In terms of ROC-AUC score. We can see similar results as we get from the F1 evaluation. We can see Baseline CNN with non-augmented images and Adam optimizer being the highest scoring model and VGG with RMSprop optimizer being the lowest scoring model. We use AUC to measure the ability of classifier to distinguish between classes and is used as a summary of the ROC curve. We will breakdown this score more in the ROC plot and confusion matrix.

In terms of training time, we can see that using scaled non-augmented data gives us the fastest training process by much. With only less than 2 minutes, our baseline CNN models, when using non-augmented data, can give us the best results among all models. Note that we are using EarlyStopping for training process. So, faster training time may be in subject of the EarlyStopping that immediately cuts the process after the training process doesn’t improve the model anymore. we will break this down more in the accuracy and loss plot. We can also see that the Densenet with RMSprop being the slowest with the training process.

In terms of comparison between Adam and RMSprop, we can see variations between the results. Sometimes, Adam yields better F1 and accuracy scores. Sometimes, RMSprop outshines Adam. Sometimes, they both give same results. So, we conclude that both optimizers perform the same for our data.

**Accuracy and Loss Plot**

Best model

**Chart, line chart

Description automatically generated**

Figure 18: Accuracy and Loss plot of Best Model

Figure 18 gives us the insight of accuracy and loss plot of the best model that we have. We can that the best model went through 15 epochs. And also, we can see from the accuracy evolution plot, even though the validation accuracy is the highest from all models, we can see that the accuracy stopped evolving after 12th epoch. And tended to go down afterward. The highest jump happened in the second epoch. We can see that the training and validation accuracies are both aligned nicely. Meaning that the biases from training data influenced the validation data.

Worst model

Chart, line chart

Description automatically generated

Figure 19: Accuracy and Loss plot of Worst Model

The worst model went through only 4 epochs. That explains why the model performed poorly. It didn’t get to have adequate training time. We can see that there’s a major accuracy drop from the second epoch. And the third and fourth epoch couldn’t reach the previous best accuracy. That’s why the EarlyStopping cut the training process prematurely. We can see that the accuracy lines don’t align with each other. Meaning that the biases are not enough to influence the validation data.

**Accuracy, Recall, Precision, F1**

Best model

**Table

Description automatically generated**

Figure 20: Classification Report of Best Model

From this report, we can see various of metrics from the best model. We can see that the recall for BACTERIAL and NORMAL classes are both good by above 90%. But when it comes to VIRAL class, the recall dropped drastically from the other two classes. We can see that the precision from normal class is way higher than both pneumonia classes. Meaning that the model can classify NORMAL images pretty good. But when classifying both of pneumonia classes, the model still struggles to differentiate them. The performance is also reflected on the F1 score. We can see that for NORMAL class, the model did a good job with 91% score. And did a poor performance when classifying both pneumonia classes.

Worst model

Table

Description automatically generated

Figure 21: Classification Report of Worst Model

As we can see from the classification report from the worst model. The pattern of scores is quite different from the best model. All of the scores for VIRAL class are 0%. Meaning that the model couldn’t classify VIRAL class even for 1 image. The NORMAL class scores are reduced by 5%. And it seems that the model can classify BACTERIAL class better than the best model because the recall score is 99%.

**ROC of AUC**

Best model

**Chart, line chart

Description automatically generated**

Figure 22: ROC Plot of Best Model

Figure 22 is the ROC plot of our best model. We can see that the NORMAL class curve is almost reaching the edge with score of 0.94. Meaning that our best model can classify NORMAL class images with a great performance. And for the pneumonia classes, which is BACTERIAL and VIRAL, we can see that the curve is more leaning to the middle. And the scores are both 0.82 and 0.71. With BACTERIAL class above VIRAL. This means that our best model still struggles to differentiate between BACTERIAL and VIRAL images. With VIRAL class being the harder one to classify.

Worst Model

Chart, line chart

Description automatically generated

Figure 23: ROC plot of Worst Model

We can see from the ROC plot from our worst model, that for the NORMAL class, the model still can differentiate it pretty well. Even though it is not as good as the best model. With the score of 0.9. The curve also tends to lie towards the edge. However, we can see from our pneumonia classes, both classes don’t perform well enough. We can see that the line for VIRAL class goes perfect straight with score of 0.5. Meaning that our worst model couldn’t classify VIRAL images even for just 1 image.

**Confusion Matrix**

Best model

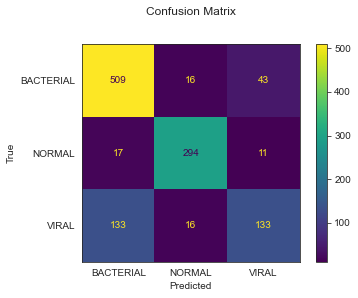
****

Figure 24: Confusion Matrix of Best Model

From the confusion matrix for best model, we can see that it is still pretty difficult for the model to classify the VIRAL images. We can see that the model mistook VIRAL class for BACTERIAL class by 133 images. This number is pretty high compared to other errors that are less than 50 images. we can also see that the errors for NORMAL class are the lowest from all errors.

Worst model

**Chart, treemap chart

Description automatically generated**

Figure 25: Confusion Matrix of Worst Model

From the confusion matrix, we can see that the model performed poorly. We can see that the model can’t even classify the VIRAL image in the test data. The model got zero in predicting VIRAL images and mistook all of it for BACTERIAL class. There’s also nothing that the model predicted for VIRAL class. The result for NORMAL class is slightly reduced in the worst model. We can conclude that for this worst model, the model’s bias for VIRAL class didn’t influence the model enough.

# Discussion of Results

**Conclusion**

1. Based on the models that we have tested, the baseline CNN models without data augmentation are the best performing with the fastest training time also.
2. We got variation of results between Adam and RMSprop. So, we can’t say which one is better from these two optimizers.
3. We can see that for all models that we have tested, the models struggle to differentiate BACTERIAL and VIRAL class.
4. VGG with RMSprop optimizer is the model that yield worst results. But we conclude that, because the training process was cut prematurely by the Earlystopping, maybe if the training process was held for longer, the performance will be much better.
5. Even though the models struggle to classify between two pneumonia classes, the models can classify NORMAL class pretty well. We conclude that, if we make this project a binary problem between NORMAL and PNEUMONIA only, the accuracies for all models will be much better.
6. All of the models were trained by short epochs and EarlyStopping. If the epochs were longer and EarlyStopping mechanism was set longer in patience, it maybe resulting in better performing models.

**Future Improvement**

We realize that our model is not perfect. And there’s still many rooms for us to develop our model and our scope of project. For now, our model can be used to classify Chest X-Ray images to three classes in term of pneumonia detection. We realize that there are many limitations in our model. For next improvement, we are planning to increase the accuracy for prediction between BACTERIAL and VIRAL. We will try to make the models classify between two pneumonia classes better. And we are planning to add more disease classes based on Chest X-Ray images. For example, COVID and TBC data. And last thing that we will do is to apply cross validation. We didn’t apply any validation for any of the model. There is also no hyperparameter tuning performed in all of our models. We think that by applying cross validation, we can increase the accuracies for our models greatly. We believe by accomplishing these tasks, we are able to significantly improve our project.

# References

Saha, S. (2018, December 17). *A comprehensive guide to Convolutional Neural Networks - the eli5 way*. Medium. Retrieved September 24, 2022, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Wikimedia Foundation. (2022, September 4). *Pneumonia*. Wikipedia. Retrieved September 24, 2022, from https://en.wikipedia.org/wiki/Pneumonia

Mayo Foundation for Medical Education and Research. (2020, June 13). *Pneumonia*. Mayo Clinic. Retrieved September 24, 2022, from https://www.mayoclinic.org/diseases-conditions/pneumonia/symptoms-causes/syc-20354204

*Pneumonia*. Johns Hopkins Medicine. (n.d.). Retrieved September 24, 2022, from https://www.hopkinsmedicine.org/health/conditions-and-diseases/pneumonia

Brownlee, J. (2019, September 16). *A gentle introduction to transfer learning for Deep learning*. Machine Learning Mastery. Retrieved September 24, 2022, from https://machinelearningmastery.com/transfer-learning-for-deep-learning/#:~:text=Transfer%20learning%20is%20an%20optimization,that%20has%20already%20been%20learned.

Lendave, V. (2021, September 1). *A comparison of 4 popular transfer learning models*. Analytics India Magazine. Retrieved September 24, 2022, from https://analyticsindiamag.com/a-comparison-of-4-popular-transfer-learning-models/

Fernandes, V., Junior, G. B., de Paiva, A. C., Silva, A. C., & Gattass, M. (2021). Bayesian convolutional neural network estimation for pediatric pneumonia detection and diagnosis. *Computer Methods and Programs in Biomedicine*, *208*, 106259. https://doi.org/10.1016/j.cmpb.2021.106259

Puneet Gupta. (2021). Pneumonia detection using convolutionalneural networks. *January 2021*, *7*(01), 77–80. https://doi.org/10.46501/ijmtst070117

Saraiva, A., Santos, D., Costa, N., Sousa, J., Ferreira, N., Valente, A., & Soares, S. (2019). Models of learning to classify X-ray images for the detection of pneumonia using neural networks. *Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies*. https://doi.org/10.5220/0007346600760083

Rahman, T., Chowdhury, M. E., Khandakar, A., Islam, K. R., Islam, K. F., Mahbub, Z. B., Kadir, M. A., & Kashem, S. (2020). Transfer learning with deep convolutional neural network (CNN) for pneumonia detection using chest X-ray. *Applied Sciences*, *10*(9), 3233. https://doi.org/10.3390/app10093233

Yu, X., Wang, S.-H., & Zhang, Y.-D. (2021). CGNet: A graph-knowledge embedded convolutional neural network for detection of pneumonia. *Information Processing & Management*, *58*(1), 102411. <https://doi.org/10.1016/j.ipm.2020.102411>

Thakur, R. (2020, November 24). Step by step VGG16 implementation in Keras for beginners. Medium. Retrieved November 26, 2022, from <https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c>

Learning, G. (2021, September 23). Everything you need to know about VGG16. Medium. Retrieved November 26, 2022, from <https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918#:~:text=VGG16%20is%20object%20detection%20and,to%20use%20with%20transfer%20learning>

Singhal, G. (2020, May 6). Gaurav Singhal. Pluralsight. Retrieved November 26, 2022, from <https://www.pluralsight.com/guides/introduction-to-densenet-with-tensorflow>

Akhtar, Z. (2021, March 3). Xception: Deep learning with depth-wise separable convolutions. OpenGenus IQ: Computing Expertise &amp; Legacy. Retrieved November 26, 2022, from <https://iq.opengenus.org/xception-model/#:~:text=Xception%20is%20a%20deep%20convolutional,version%20of%20an%20Inception%20module>.

Team, G. L. (2022, March 22). Introduction to resnet or residual network. Great Learning Blog: Free Resources what Matters to shape your Career! Retrieved November 26, 2022, from <https://www.mygreatlearning.com/blog/resnet/>

Google. (n.d.). Advanced guide to inception V3 &nbsp;|&nbsp; cloud TPU &nbsp;|&nbsp; google cloud. Google. Retrieved November 26, 2022, from <https://cloud.google.com/tpu/docs/inception-v3-advanced>

Team, K. (n.d.). Keras Documentation: Getting started. Keras. Retrieved December 1, 2022, from <https://keras.io/getting_started/>

Softmax function. DeepAI. (2019, May 17). Retrieved December 2, 2022, from <https://deepai.org/machine-learning-glossary-and-terms/softmax-layer#:~:text=The%20softmax%20function%20is%20a,can%20be%20interpreted%20as%20probabilities>

Ye, A. (2020, September 2). If rectified linear units are linear, how do they add nonlinearity? Medium. Retrieved December 2, 2022, from <https://towardsdatascience.com/if-rectified-linear-units-are-linear-how-do-they-add-nonlinearity-40247d3e4792>

Brownlee, J. (2020, December 22). A gentle introduction to cross-entropy for Machine Learning. MachineLearningMastery.com. Retrieved December 2, 2022, from <https://machinelearningmastery.com/cross-entropy-for-machine-learning/>

Brownlee, J. (2021, October 11). Gradient descent with RMSProp from scratch. MachineLearningMastery.com. Retrieved December 2, 2022, from https://machinelearningmastery.com/gradient-descent-with-rmsprop-from-scratch/