Final Report

School of Software, Yunnan University

Student ID	Name	Score
20183290375	**MD MAHFUZUR RAHMAN 罗尼**	

Semester: Spring 2021

Course: Innovative Design and Research

Project: Pneumonia Detection using CNN

Table of content

Summary

In recent years, medical image processing has become a research hotspot in the field of computer vision. At present, the diagnosis of pneumonia imaging types mainly relies on the experience of doctors. The hospital needs to set up special departments and personnel to make judgments. This is time-consuming and laborious. Moreover, some CT images of pneumonia are very similar, and doctors are likely to make mistakes in judgments and cause misdiagnosis. Therefore, a technology that can assist doctors in diagnosis and reduce the rate of misdiagnosis is needed.

Through experiments, it is found that traditional image processing methods have a low recognition rate for pneumonia classification. Image features are mainly selected artificially, which cannot accurately represent the target. Deep learning is a very popular direction in the field of machine learning in recent years. As a representative network of deep learning, convolutional neural networks have the ability to learn features independently, and are invariant to displacement, scaling, and distortion. The convolutional neural network can be trained through a large amount of labeled pneumonia type data, and the characteristics of the pneumonia type can be learned independently to distinguish the type of pneumonia.

This project was created using Pytorch. PyTorch is python native, and integrates easily with other python packages, which makes this a simple choice for researchers. Many researchers use Pytorch because the API is intuitive and easier to learn, and get into experimentation quickly, rather than reading through documentation.

In this experiment first I use CNN to build a model and create model class then do this experiment I use VGG-16 and ResNet-50.

Here, I am using Residual Network (ResNet-50) via transfer learning, a popular deep learning neural network model, to classify Pneumonia chest X-ray scans. And also with VGG-16. The experiment was performed on the pneumonia dataset of “Guangzhou Women and Children’s Medical Center, Guangzhou”. This dataset consists of around 5,000 and 600 images in the training and testing set respectively. It is quite small by today's big era standard, and it presents some challenges such as data imbalance that I will attempt to mitigate. By leveraging the power of transfer learning, we are able to 'transfer' the weights of low-level features (e.g., lines, shapes, etc.) that were detected in a pretrained model. Doing so saves us the need to train a model from scratch, which can be challenging for those who do not have a 'big' enough dataset or computational resources.

The experimental results show that the recognition accuracy of the model on the validation set reaches 95.42%. The accuracy of the VGG-16 was 91.35%; while the RestNet-50 network model trained and got an accuracy of 88.00%. Comparing the result of this test, it is not difficult to find that VGG-16 is better effect in the recognition of CT images of pneumonia.

Background of topic selection

Since December 2019, there have been many cases of pneumonia caused by the new type of coronavirus in Wuhan, Hubei Province. Starting in January 2020, the pneumonia that broke out in Wuhan has spread rapidly throughout Hubei and in the world. Through the complete gene sequencing of the patient's alveolar lavage fluid, a new type of β-coronavirus was discovered, which is significantly different from SARS and MERS-related coronaviruses; but its gene homology with bat SARS-like viruses reaches 85 %; Mainly respiratory symptoms, fever and other clinical manifestations are very similar to other viral pneumonia, can still cause gastrointestinal symptoms. On February 7, 2020, the National Health Commission named the new coronavirus-infected pneumonia "New Coronavirus Pneumonia" (referred to as New Coronary Pneumonia), and issued the "New Coronavirus Pneumonia Diagnosis and Treatment Plan, "Patients with imaging features of pneumonia in suspected cases" are regarded as the clinical diagnostic case standard in Hubei Province. At present, relevant international organizations and institutions have new names for this disease and the virus that causes it, namely COVID-19 (CoronaVirusDisease-19) and SARS-CoV-2. Relevant diagnosis and treatment guidelines at home and everywhere are still being updated. The diagnosis of new coronary pneumonia is based on new coronavirus nucleic acid testing or viral gene sequencing. Because nucleic acid testing has a certain false negative rate, those with negative nucleic acid testing and typical imaging performance still need to be vigilant, especially in COVID-19 During the outbreak of pneumonia, some cases had no clinical symptoms, even throat swab nucleic acid tests were negative, and those with characteristic pneumonia on CT were finally diagnosed as new coronary pneumonia.

The main differential diagnosis of new coronary pneumonia includes cytomegalovirus pneumonia, viral pneumonia such as SARS and MERS, and mycoplasma pneumonia. CT images of the new type of coronavirus pneumonia have certain characteristics. Familiarity with these features will help the early diagnosis, early isolation, and early treatment of new coronary pneumonia, and curb the spread of the new coronary pneumonia epidemic. The identification of pneumonia imaging types currently mainly relies on the experience of doctors. The hospital needs to set up special departments and personnel to make judgments. This is time-consuming and laborious, and it is a very inefficient method in the face of large-scale conditions such as new coronary pneumonia. Moreover, the CT images of some pneumonia are very similar, and doctors are likely to make mistakes in judgments and cause misdiagnosis.

In recent years, there have been many methods in the research of pneumonia recognition, the more common one is the template matching method, which is simple and clear, and easy to get practical applications. However, this model emphasizes that the image must match the template in order to be recognized. This is more difficult for CT images of pneumonia, because everyone's lung images are different, and the same disease only has similar features, but does not appear to be imaged. If there is no fixed template, you can only choose a representative one as the template, and calculate the similarity between the extracted features to determine.

Knowledge about pneumonia

Pneumonia is an inflammatory condition of the lung affecting primarily the small air sacs known as alveoli. Symptoms typically include some combination of productive or dry cough, chest pain, fever and difficulty breathing. The severity of the condition is variable. Pneumonia is usually caused by infection with viruses or bacteria and less commonly by other microorganisms, certain medications or conditions such as autoimmune diseases. Risk factors include cystic fibrosis, chronic obstructive pulmonary disease (COPD), asthma, diabetes, heart failure, a history of smoking, a poor ability to cough such as following a stroke and a weak immune system. Diagnosis is often based on symptoms and physical examination. Chest X-ray, blood tests, and culture of the sputum may help confirm the diagnosis. The disease may be classified by where it was acquired, such as community- or hospital-acquired or healthcare-associated pneumonia.

2.1 Chest imaging test:

X-rays or scans produce an image of the organs and structures in the chest.

- X-rays (radiography) use radiation to produce a 2-D image. Usually done in hospitals, using fixed equipment by a radiographer, they can also be done on portable machines.

- Computed tomography (CT) scans use a computer to merge 2-D X-ray images and convert them to a 3-D image. They require highly specialized equipment and are done in hospital by a specialist radiographer.

- Ultrasound scans use high-frequency sound waves to produce an image. They can be done in hospitals or other healthcare settings, such as a doctor’s office.

Currently, a formal diagnosis of COVID-19 requires a laboratory test (RT-PCR) of nose and throat samples. RT-PCR requires specialist equipment and takes at least 24 hours to produce a result. It is not completely accurate, and may require a second RT-PCR or a different test to confirm diagnosis.

COVID-19 is a respiratory disease. Clinicians may use chest imaging to diagnose people who have COVID-19 symptoms, while awaiting RT-PCR results or when RT-PCR results are negative, and the person has COVID-19 symptoms.

2.2 Imaging examination technology for new coronary pneumonia

Chest DR examination (chest radiograph)-Because DR is an overlapping image, it shows limited early and slight inflammatory changes (especially the interstitial of the lung), and there are many missed diagnoses or false negatives. In critically ill patients in the ICU isolation ward (especially mechanical ventilation with tracheal intubation), bedside DR is necessary. Chest CT examination-tomographic images, no overlapping structure interference; volume acquisition, thin-layer high-resolution image reconstruction technology, for early and slight inflammatory changes in the lungs, whether it is the lung interstitial or the lung parenchyma, it can be very clear Display.

2.3 Imaging manifestations of new coronary pneumonia

In the early stage, new coronary pneumonia presents multiple small patchy shadows and interstitial changes, which are obvious outside the lungs; and then develops multiple ground glass shadows, infiltration shadows and fine mesh changes in both lungs; severe cases may have lung consolidation. CT image manifestations of new coronary pneumonia.

Early stage:

- Small patchy/nodular ground glass shadow GGO;
- Limited lesions, can appear in both lungs, more common in the lower lobe; can also be single lung, single lobe, subsegmental or segmental, distributed under the pleura;
- With or without fine mesh shadow (swelling and thickening of interlobular septal edema), the phenomenon of thickening of small blood vessels is often seen at this time;
- The lesions are distributed along the bronchial vascular bundle.

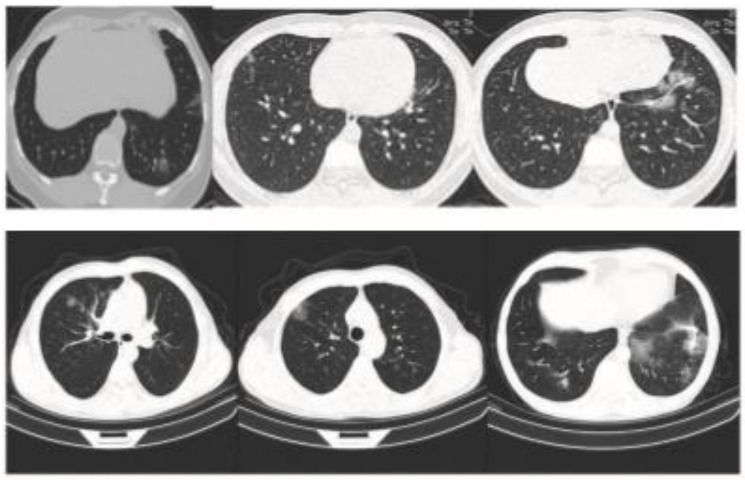


Figure 1 CT impact of early pneumonia

Advanced stage:

The number of lesions increases, the scope of the lesions expands or merges, and the density of the lesions increases.

- Ground glass shadow and consolidation coexist, distributed in large flakes.
- Often both lungs and multiple lobes are involved, and they are distributed asymmetrically under the pleura and extend to the hilar area.
- Aerated bronchial sign can be seen in the consolidation.
- There is subsegmental atelectasis and a little fibrosis.

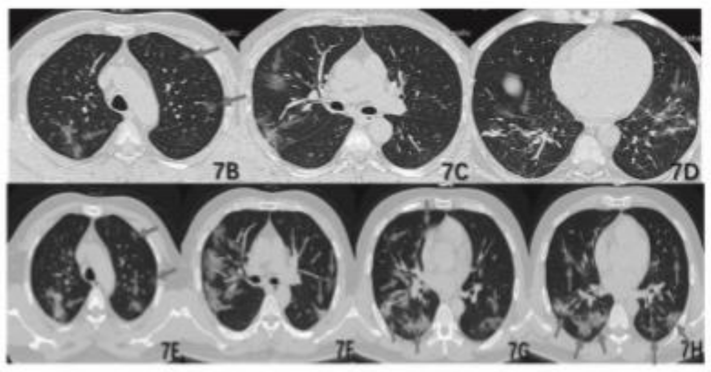


Figure 2 CT impact of advanced pneumonia

Severe stage:

- Diffuse lung lesions, 2/3 of the lung fields are affected by the lesion, showing a "white lung" appearance;
- 48-hour lesion area increases by 50%;
- Consolidation is the main component, combined with GGO, and often accompanied by striped shadows;
- Air bronchial signs can be seen in the consolidation;
- The condition is serious and clinically obvious respiratory failure appears.

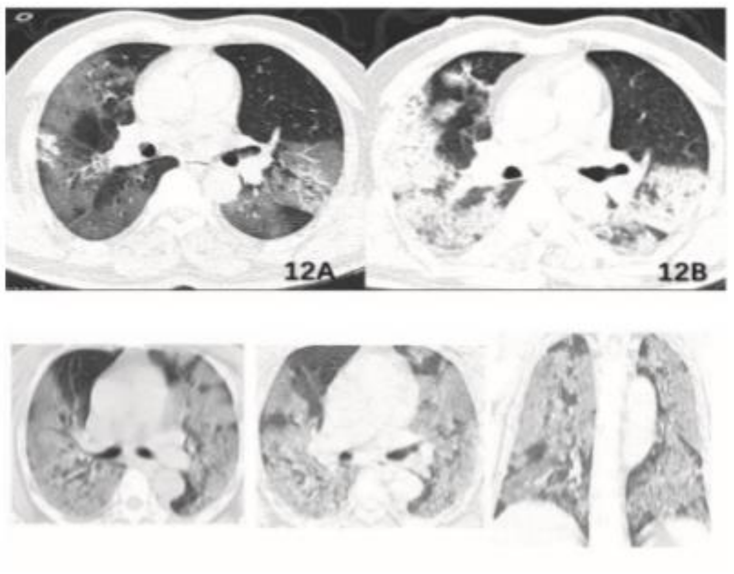


Figure 3 CT impact of severe pneumonia

Outcome after treatment: HRCT of the chest showed that the exudative lesions of both lungs were obviously absorbed, the scope was reduced, and the residual interstitial fibers were changed.

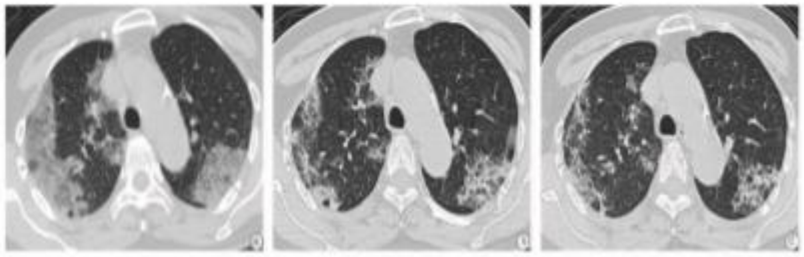


Figure 4 CT impact of pneumonia after treatment

Image recognition related knowledge

Image recognition is an important part of the field of computer vision and artificial intelligence, and its ultimate goal is to make computers have the ability to analyze and understand image content. The development of image recognition has gone through three stages: text recognition, digital image processing and recognition, and object recognition.

3.1 The principle of image recognition technology

There is no essential difference between computer image recognition technology and human image recognition in principle. Human image recognition is based on the classification of the characteristics of the image, and then the image is recognized by the characteristics of each category. When looking at When we arrive at a picture, our brain will quickly sense whether we have seen this picture or a picture similar to it. In this process, our brain will recognize according to the categories that have been classified in the stored memory to see if there is a stored memory with the same or similar characteristics as the image, so as to recognize whether it has seen the image.

Image recognition technology is based on the main characteristics of the image. Each image has its characteristics, such as the letter A has a sharp point, P has a circle, and the center of Y has an acute angle. Research on eye movement during image recognition shows that the line of sight is always focused on the main features of the image, that is, where the contour of the image is the largest or the direction of the contour changes suddenly, where the amount of information is the greatest. Moreover, the scanning route of the eye always turns from one feature to another one by one. It can be seen that in the process of image recognition, the perceptual mechanism must exclude the input redundant information and extract the key information. At the same time, there must be a mechanism responsible for the integration of information in the brain, which can organize the information obtained in stages into a complete perceptual image.

Pattern recognition is an important part of artificial intelligence and information science. Pattern recognition refers to the process of analyzing and processing different forms of information representing things or phenomena to obtain a description, identification and classification of things or phenomena.

3.2 The process of image recognition technology

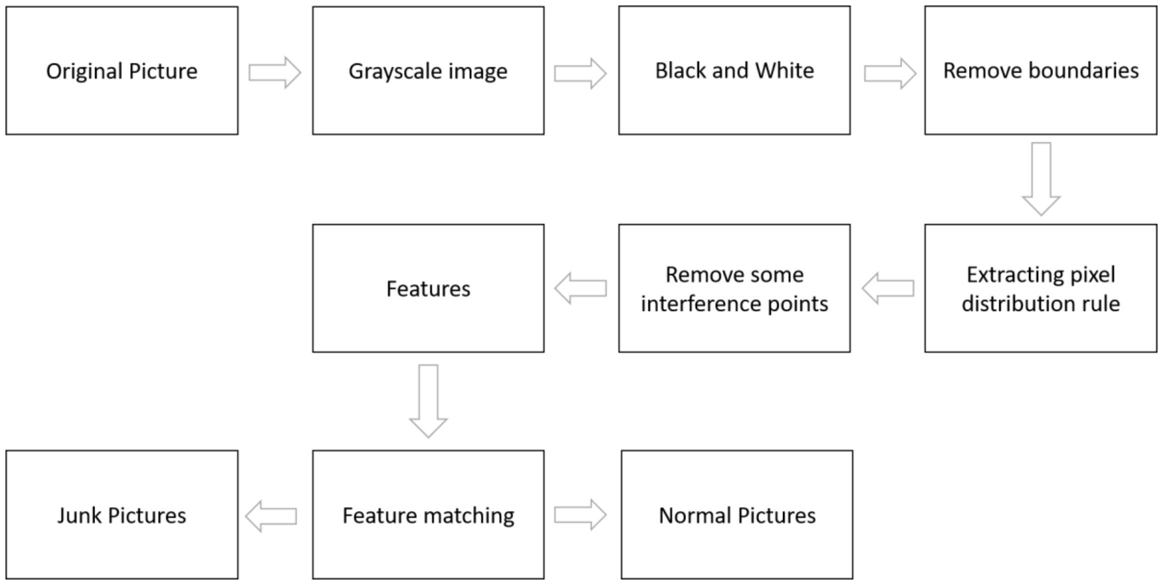


Figure 5 process of image recognition technology

The process of image recognition technology is divided into the following steps: information acquisition, preprocessing, feature extraction and selection, classifier design and classification decision-making.

The acquisition of information refers to the conversion of information such as light or sound into electrical information through sensors. That is to obtain the basic information of the research object and transform it into information that can be recognized by the machine in some way.

Preprocessing mainly refers to operations such as denoising, smoothing, and transformation in image processing, so as to enhance the important features of the image.

Feature extraction and selection refers to the need for feature extraction and selection in pattern recognition. The simple understanding is that the images we study are of all kinds. If we want to distinguish them by a certain method, we must identify them by their own characteristics, and the process of obtaining these characteristics is feature extraction.

The features obtained in feature extraction may not be all useful for this recognition. At this time, useful features must be extracted. This is the feature selection. Feature extraction and selection are one of the key technologies in the image recognition process, so the understanding of this step is the focus of image recognition.

3.3 Classification of image recognition technology

Image recognition technology can be divided into neural network image recognition technology and non-linear dimensionality reduction image recognition technology.

(1) Image recognition technology of neural network

Neural network image recognition technology is a relatively new type of image recognition technology, which is an image recognition method based on traditional image recognition methods and neural network algorithms. The neural network here refers to the artificial neural network, which means that this neural network is not the real neural network possessed by the animal itself, but is artificially generated by humans after imitating the animal neural network.

In the neural network image recognition technology, the neural network image recognition model fused with genetic algorithm and BP network is very classic and has its applications in many fields. In the image recognition system, the neural network system is generally used to extract the features of the image first, and then map the features of the image to the neural network for image recognition and classification.

(2) Image recognition technology with nonlinear dimensionality reduction

Computer image recognition technology is an unusually high-dimensional recognition technology. Regardless of the resolution of the image itself, the data it generates is often multi-dimensional, which brings great difficulties to computer recognition. If you want a computer to have the ability to identify efficiently, the most direct and effective way is to reduce dimensionality. Dimensionality reduction is divided into linear dimension reduction and non-linear dimension reduction. For example, principal component analysis (PCA) and linear singular analysis (LDA) are common linear dimensionality reduction methods, which are simple and easy to understand. However, the overall data set is processed by linear dimensionality reduction, and what is sought is the optimal low-dimensional projection of the entire data set.

It has been verified that this linear dimensionality reduction strategy has high computational complexity and takes up relatively more time and space. Therefore, image recognition technology based on non-linear dimensionality reduction has been produced, which is an extremely effective method for non-linear feature extraction. . This technology can discover the non-linear structure of the image and can reduce its dimensionality without destroying its intrinsic structure, so that the computer's image recognition can be carried out in the lowest possible dimensionality, thus increasing the recognition rate.

Model Architecture

Overall Model Architecture

Pneumonia CT image recognition methods are generally divided into image preprocessing, artificial extraction of designated features, feature enhancement, feature classifier selection, and feature classification. The image preprocessing is to prepare for the next feature extraction operation. Some basic operations are performed on the image, such as grayscale, binarization, noise removal, etc.; artificial selection of designated features is mainly based on the classification target, and the artificial selection of energy Represents a certain feature of the target. Commonly used image features include color features, texture features, shape features, and spatial relationship features; feature enhancement is to selectively make the target of interest clear, suppress areas of interest, and satisfy the target. Needs. Feature classifier selection. Different feature classifiers have different classification effects. Some classifiers have simple structure and fast calculation speed. Some classifiers have high classification accuracy but a large amount of calculation. According to specific practical applications, there are choices Classifiers can often achieve good classification results. Feature classification, use the extracted features to train the classifier to achieve the final classification. The basic block diagram of the pneumonia CT image recognition method is shown below:

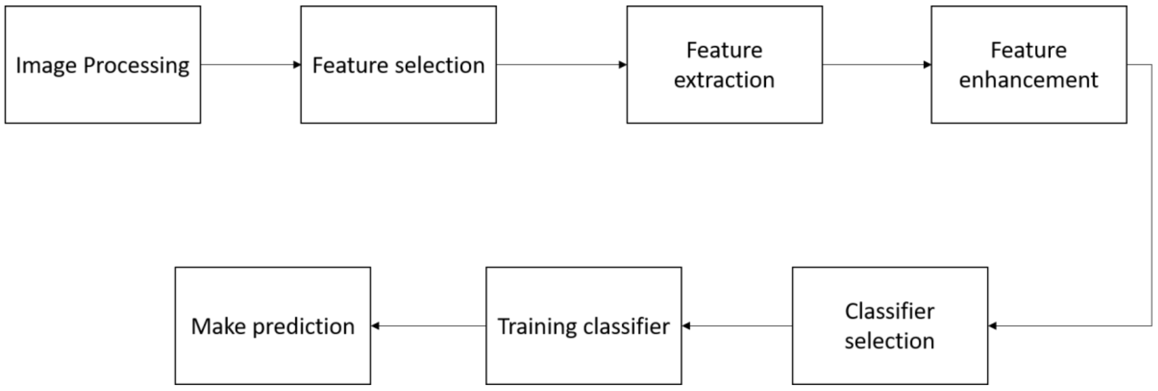


Figure 6 image recognition method of pneumonia

The specific steps of the content to be achieved in this article are similar to the above figure. Pneumonia CT image preprocessing: The purpose of image preprocessing is to remove useless information, such as noise contained in the image caused by some external factors, and to restore some important feature information in preparation for feature extraction or image segmentation. Image preprocessing operations generally include grayscale, smoothing, radiation transformation, image restoration and other operations. Before pneumonia feature extraction, image preprocessing operations are required: The flow chart of the entire system in shown below:

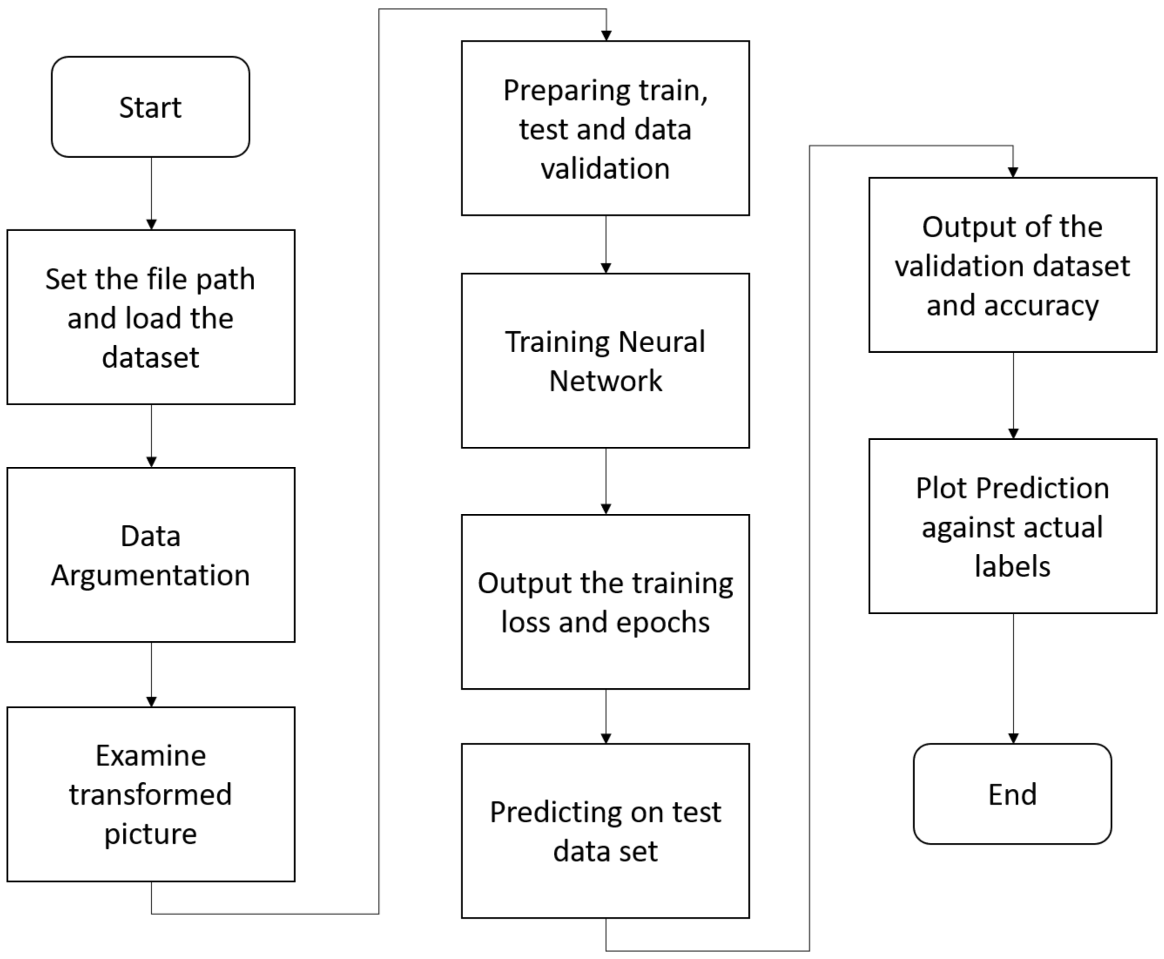
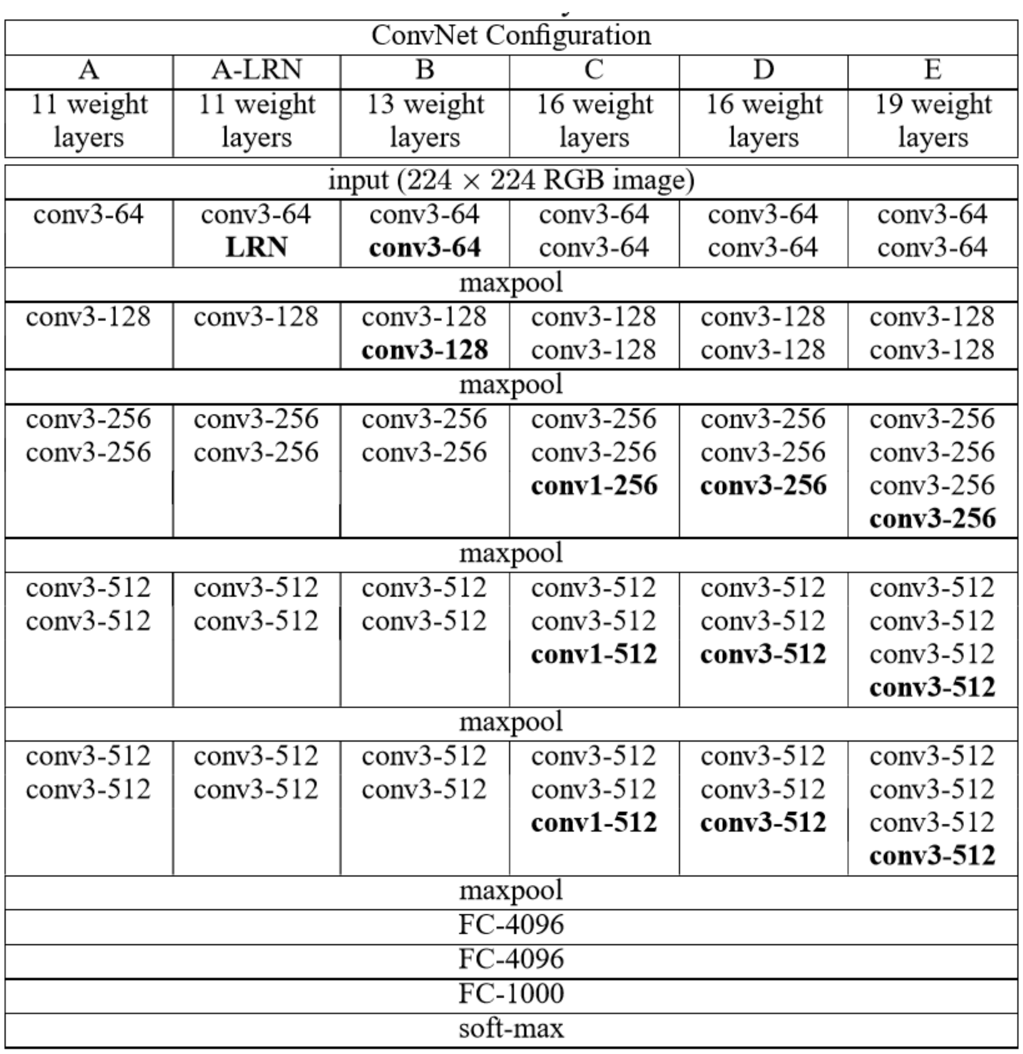

Start
Set the file path and load the dataset
Data Argumentation
Examine transformed picture
Preparing train, test and data validation
Training Neural Network
Output the training loss and epochs
Predicting on test data set
Output of the validation dataset and accuracy
Plot Prediction against actual labels
End

Figure 7 System flow chart

VGG Network

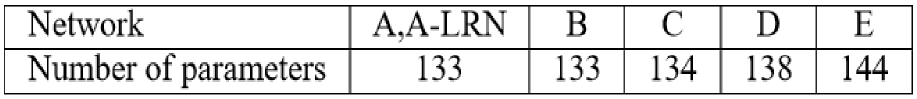
VGG-Net is a deep convolutional neural network developed by researchers from the Computer Vision Group of Oxford University and Google DeepMind. VGG-Net explored the relationship between the depth of the convolutional neural network and its performance. By repeatedly stacking 3*3 small convolution kernels and 2*2 maximum pooling layers, VGG-Net successfully constructed a 16–19-layer deep convolution Neural Networks. Compared with the previous state-of-the-art network structure, VGG-Net has a significant drop-in error rate, and achieved the 2nd place in the ILSVRC2014 competition classification project and the 1st place in the positioning project. At the same time, VGG-Net is very extensible, and the generalization of migration to other image data is very good. The structure of VGG-Net is very simple, and the entire network uses the same size of the convolution kernel size (3*3) and maximum pooling size (2*2). So far, VGG-Net is still often used to extract image features. The model parameters after VGG-Net training are open source on its official website and can be used for retraining on specific image classification tasks (equivalent to providing very good initialization weights), so they are used in many places.

All 3*3 convolution kernels and 2*2 pooling kernels are used in VGG-Net, and the performance is improved by continuously deepening the network structure. In below figure shows the network structure diagram of each level of VGG-Net, and after that figure shows the parameters of each level. There are detailed performance tests from the 11-layer network to the 19-layer network. Although the network at each level from A to E has gradually become deeper, the number of parameters of the network has not increased much, because the number of parameters is mainly consumed in the last three fully connected layers. Although the previous convolution part is very deep, but the number of parameters consumed is not large, but the more time-consuming part of training is still convolution, because of the relatively large amount of calculation. Among them, D and E are what we often call VGGNet-16 and VGGNet-19. C is very interesting. Compared with B, there are several 1*1 convolutional layers. The significance of 1*1 convolution is mainly linear transformation, while the number of input channels and output channels remain unchanged, and no dimensionality reduction occurs. The input of VGG-Net is set to a 224x244 RGB image. Calculate the RGB average of all images on the training set image, and then pass the image as input to the VGG convolutional network, using a 3x3 or 1x1 filter, and the convolution step size is fixed to 1. The VGG fully connected layer has 3 layers. According to the purpose of the total number of convolutional layers + fully connected layers, it can be from VGG11 to VGG19. The least VGG11 has 8 convolutional layers and 3 fully connected layers, and the most VGG19 has 16 convolutional layers+3 fully connected layers. In addition, the VGG network is not followed by a pooling layer after each convolutional layer, but a total of 5 pooling layers, distributed under different convolutional layers.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 8 VGGNet structure diagram of each level



Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Figure 9 VGGNet parameters of each lever

VGGNet has 5 segments of convolution, each segment has 2 to 3 convolutional layers, and at the same time, a maximum pooling layer is connected at the end of each segment to reduce the image size. The number of convolution kernels in each segment is the same, and the later segments have more convolution kernels: 64-128-256-512. There are often multiple identical 3*3 convolutional layers stacked together, which is actually a very useful design. As shown in below, the concatenation of two 3*3 convolutional layers is equivalent to a 5*5 convolutional layer, that is, a pixel will be associated with the surrounding 5*5 pixels. It can be said that the receptive field size is 5* 5. The effect of three 3*3 convolutional layers in series is equivalent to one 7*7 convolutional layer.

In addition, three concatenated 3*3 convolutional layers have fewer parameters than a 7*7 convolutional layer, and only the latter's (3*3*3)/(7*7)=55%. The most important thing is that three 3*3 convolutional layers have more non-linear transformations than one 7*7 convolutional layer (the former can use three ReLU-activation functions, while the latter only has one), which makes the CNN feature The learning ability is stronger.

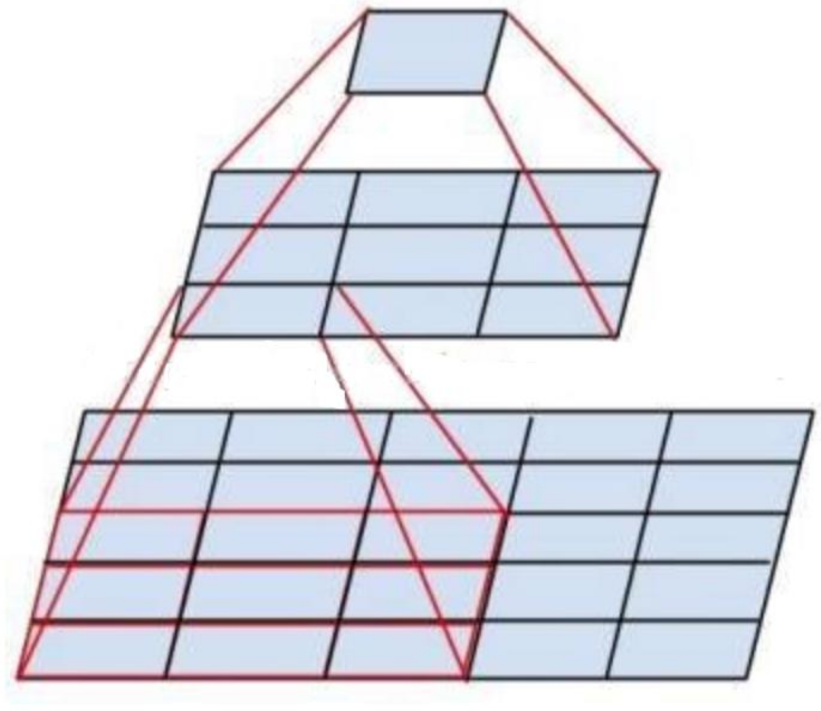
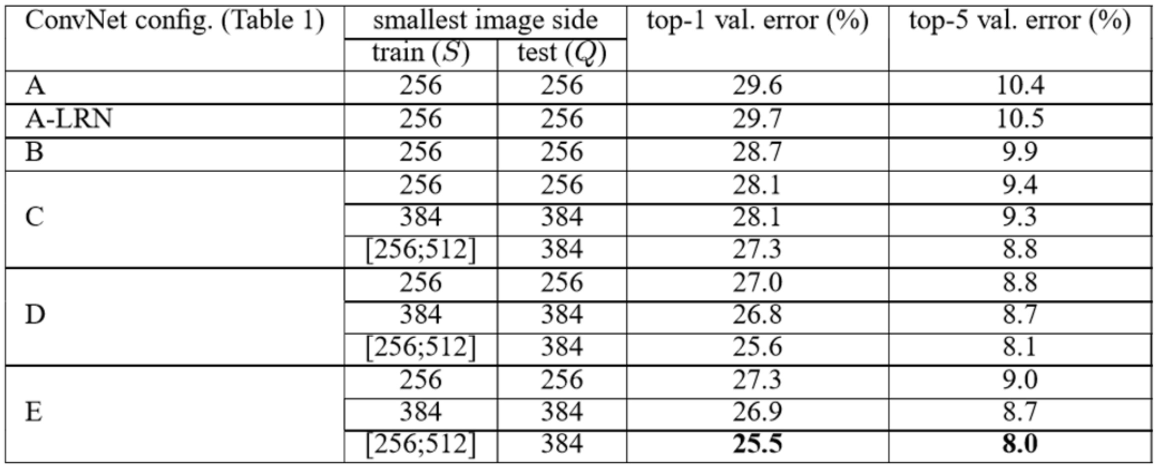


Figure 10 The function of two concatenated 3*3 convolutional layers is similar to a 5*5 convolutional Layer.

VGGNet has a small trick when training, first train a simple network of level A, and then reuse the weight of the A network to initialize the following complex models, so that the training convergence speed is faster. When predicting, VGG uses the multi-Scale method to scale the image to a size Q, and input the image into the convolutional network for calculation. Then in the last convolutional layer, a sliding window method is used for classification prediction, the classification results of different windows are averaged, and the results of different sizes Q are averaged to obtain the final result, which can improve the utilization rate of image data and improve the prediction accuracy. In training, VGGNet also uses the multi-Scale method for data enhancement, scaling the original image to a different size S, and then randomly cropping 224´224 images, which can increase a lot of data and prevent overfitting of the model. Has a very good effect. In practice, let S take a value in the interval [256,512], use multi-Scale to obtain multiple versions of data, and combine multiple versions of data for training. Obtain figure shows the results obtained when VGGNet uses multi-Scale training. It can be seen that both D and E can reach an error rate of 7.5%. The final version submitted to ILSVRC2014 is a fusion of only 6 different levels of Single-Scale networks and Multi-Scale D networks, achieving an error rate of 7.3%. However, after the game, the author found that only fusing Multi-Scale D and E can achieve better results, with an error rate of 7.0%, and then using other optimization strategies to achieve a final error rate of about 6.8%, which is very close to the champion Google InceptinNet of the same year. At the same time, the following points are summarized when comparing networks at all levels:

1. The LRN layer is not very useful (VGGNet does not use local response standardization (LRN). This standardization does not improve performance on the ILSVRC data set, but leads to More memory consumption and calculation time.);
2. The deeper the network, the better;
3. 1*1 convolution is also very effective, but it is not as good as 3*3 convolution, larger convolution The product kernel can learn larger spatial features.



ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

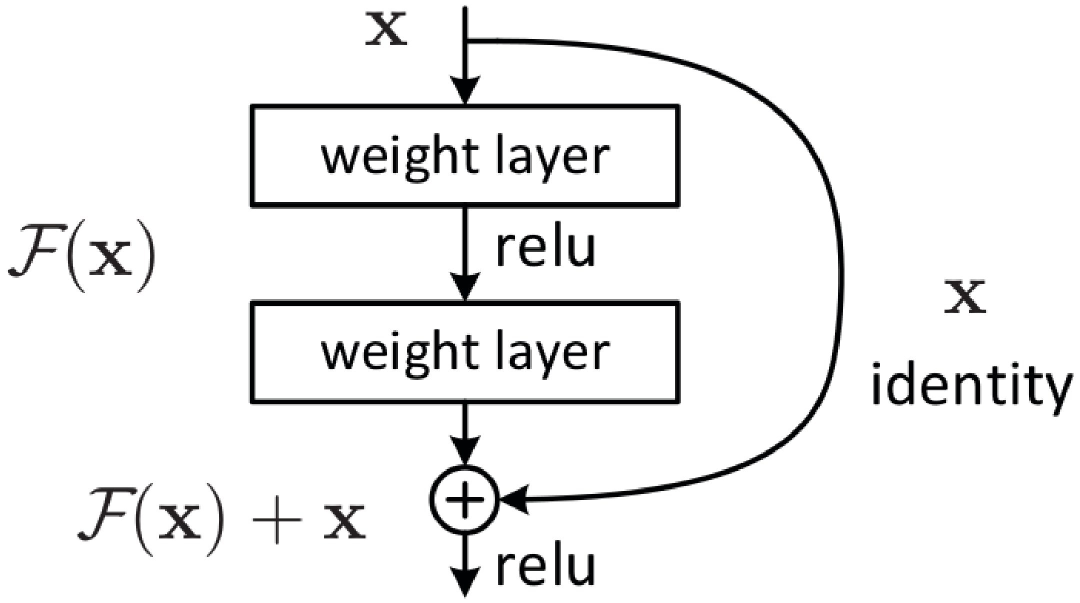
Figure 11 the top-5 error rate of each level of VGGNet when using multi-scale training

Resnet

In 2012 at the LSVRC2012 classification contest Alex Net won the first price, after that Reset was the most interesting thing that happened to the computer vision and the deep learning world.

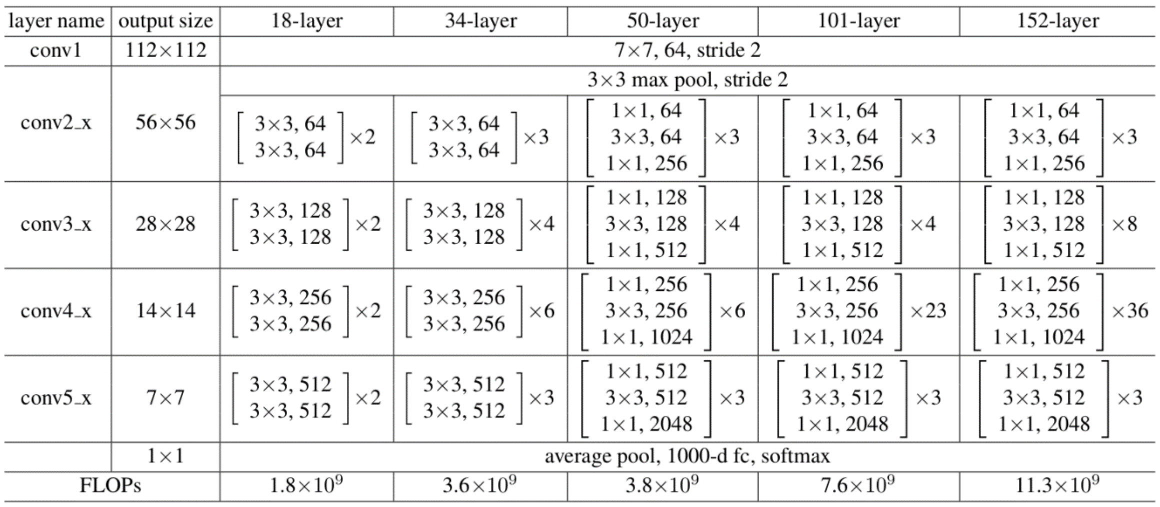
Because of the framework that ResNets presented it was made possible to train ultra-deep neural networks and by that, I mean that I network can contain hundreds or thousands of layers and still achieve great performance.

The ResNets were initially applied to the image recognition task but as it is mentioned in the paper that the framework can also be used for non-computer vision tasks also to achieve better accuracy.


x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

They explicitly let the layers fit a residual mapping and denoted that as H(x) and they let the nonlinear layers fit another mapping F(x): =H(x)−x so the original mapping becomes H(x): =F(x)+x as can be seen in figure 12.

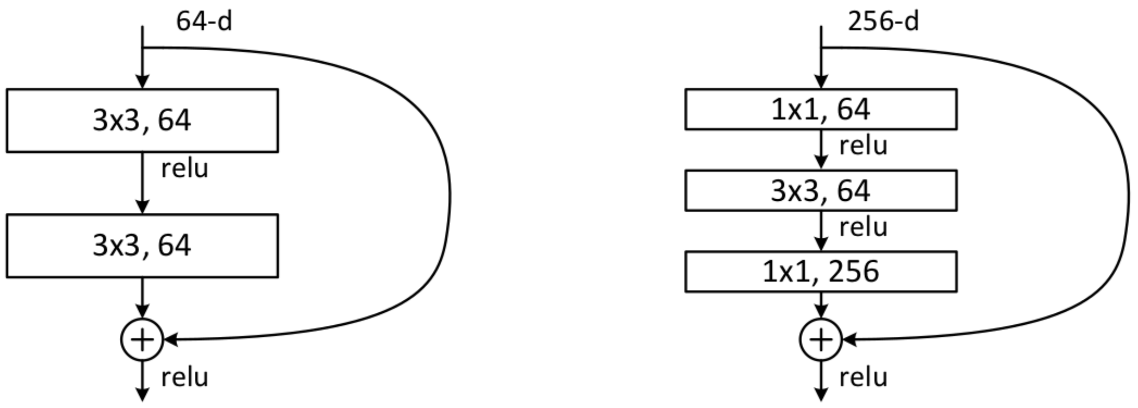
And the benefit of these shortcut identity mapping was that there were no additional parameters added to the model and also the computational time was kept in check.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 13 ResNet50 Architecture

There was a small change that was made for the ResNet 50 and above that before this the shortcut connections skipped two layers but now they skip three layers and also there was 1 * 1 convolution layers added that we are going to see in detail with the ResNet 50 Architecture.



So, as we can see in the table 1 the resnet 50 architecture contains the following element:

- A convolution with a kernel size of 7 * 7 and 64 different kernels all with a stride of size 2 giving us 1 layer.
- Next, we see max pooling with also a stride size of 2.
- In the next convolution there is a 1 * 1,64 kernel following this a 3 * 3,64 kernel and at last a 1 * 1,256 kernel, these three layers are repeated in total 3 time so giving us 9 layers in this step.
- Next, we see kernel of 1 * 1,128 after that a kernel of 3 * 3,128 and at last a kernel of 1 * 1,512 this step was repeated 4 time so giving us 12 layers in this step.
- After that there is a kernel of 1 * 1,256 and two more kernels with 3 * 3,256 and 1 * 1,1024 and this is repeated 6 time giving us a total of 18 layers.

- And then again, a 1 * 1,512 kernel with two more of 3 * 3,512 and 1 * 1,2048 and this was repeated 3 times giving us a total of 9 layers.
- After that we do a average pool and end it with a fully connected layer containing 1000 nodes and at the end a softmax function so this gives us 1 layer.

We don't actually count the activation functions and the max/ average pooling layers. so, totaling this it gives us a 1 + 9 + 12 + 18 + 9 + 1 = 50 layers Deep Convolutional network.

Model Comparison

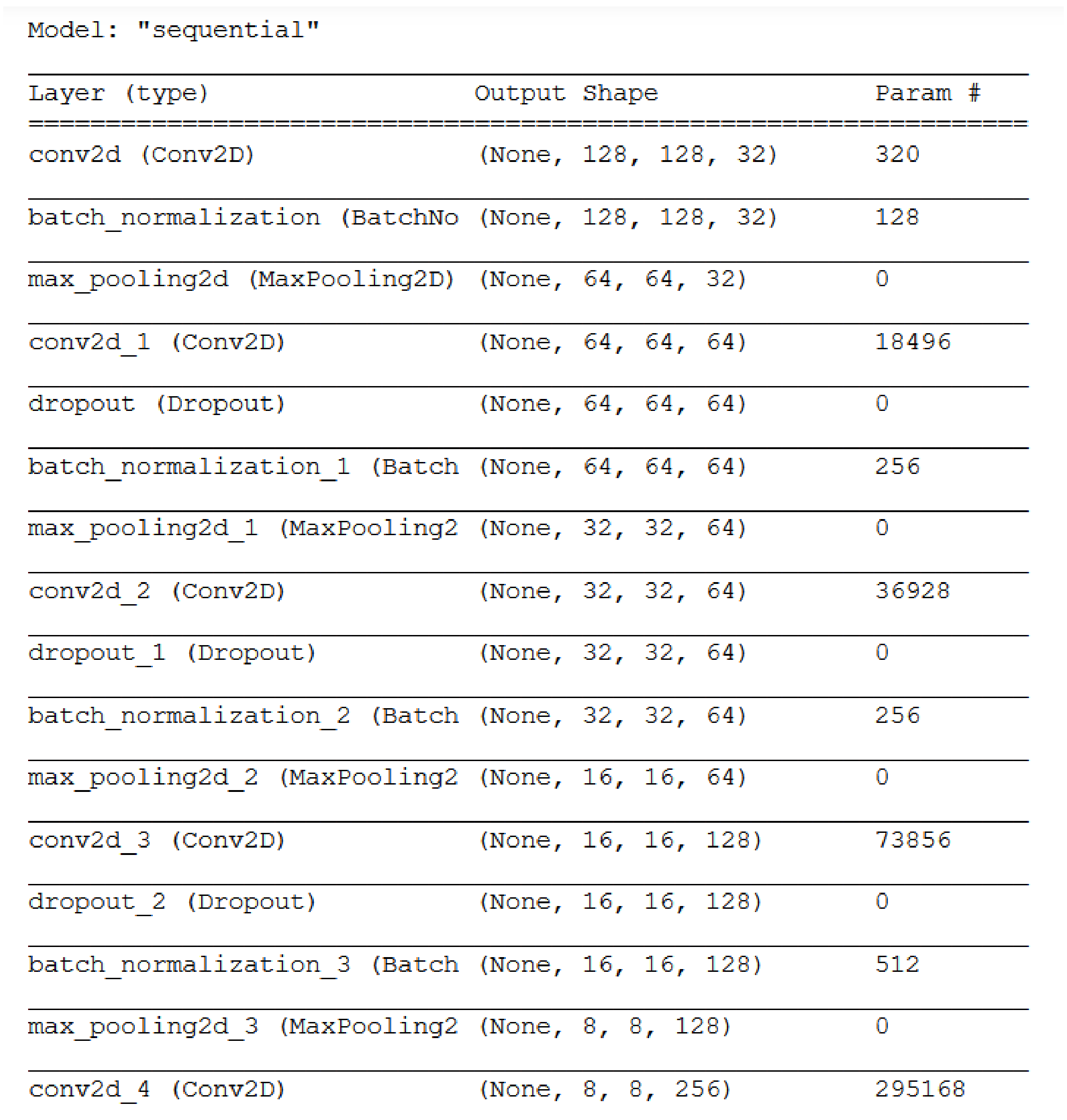
VGGNet: The architecture comes from Oxford VGG Group. It improves on AlexNet by replacing the large kernel size filters (11 and 5 in the first and second convolutional layers, respectively) with multiple 3X3 kernel-sized filters one after another. In the case of a given receiving field (the output depends on the effective area size of the input image), multiple stacked smaller-sized kernels are better than larger-sized kernels, because multiple non-linear layers will increase the depth of the network, so that it can understand more complex functions, and the price is lower. For example, three 3X3 filters overlap each other with a stride of 1, and a receiving size of 7, but compared with the 49C^2 parameter of a kernel with a size of 7, the number of parameters involved is 3*(9C^2). Here, suppose the number of input and output channels of the layer is C. In addition, the 3X3 kernel helps preserve the finer-level attributes of the image.

ResNet: Although Resnet has achieved amazing accuracy on the ImageNet data set, due to the huge computational requirements in terms of memory and time, it is a problem even if it is deployed on the most moderately sized GPU. Due to the large width of the convolutional layer, the efficiency is low.

Parameters Setting

Number of Conv Layer	Number of Pooling Layer	The size of Conv Layer	The type of activation func	Padding	Dropout
3,3	2,2	32	relu	same	0.2
3,3	2,2	64	relu	same	0.2
3,3	2,2	128	relu	same	0.2
3,3	2,2	256	relu	same	0.2

In model summary, I created sequential class model and model layers are created and added to it. Here for the first layer the number of conv layer (3,3), number of pooling layer (2,2), the size of conv layer is 32, the activation function is relu, padding is same and dropout is 0.2. For the second layer the size of conv layer 64, conv layer (3,3) pooling layer (2,2), type of activation is relu, dropout 0.2 and the padding is same. I set another two parameters where only the size of Convolution layer is different (128 and 256) but other parameter is same. Also add Flatten model where dropout (0.3) and dense model where activation function relu and for send dense activation function is sigmoid.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 32)      320
_________________________________________________________________
batch_normalization (BatchNo (None, 128, 128, 32)      128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
dropout (Dropout)            (None, 64, 64, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 64, 64, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 32, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 128)       73856
_________________________________________________________________
dropout_2 (Dropout)          (None, 16, 16, 128)       0
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 256)         295168
```

```
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 256)         295168
_________________________________________________________________
dropout_3 (Dropout)          (None, 8, 8, 256)         0
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 256)         1024
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dropout_4 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 512)               2097664
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 2,525,121
Trainable params: 2,524,033
Non-trainable params: 1,088
```

In model summary we can see for Conv2D type- for conv2d output shape is (None, 128,128,32) and parameter is 320. For conv2d_1 output shape is (None, 64,64,64) and parameter is 18496. For conv2d_2 output shape is (None, 32,32,64) and parameter is 36928. For conv2d_3 output shape is (None, 16,16,128) and parameter is 73856. For conv2d_4 output shape is (None, 8,8,256) and parameter is 295168.

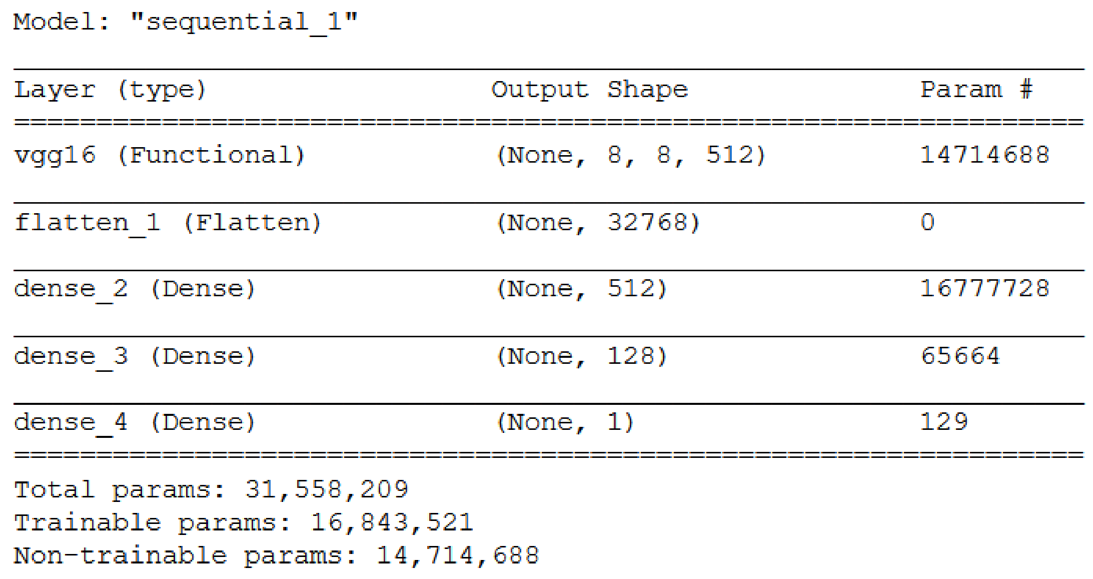
For Batch normalization- batch_normalization output shape (None,128,128,32) and param is 128. For batch_normalization_1 output shape (None,64,64,256) and param is 256. batch_normalization_2 output shape (None,32,32,64) and param is 256. batch_normalization_3 output shape (None,16,16,128) and param is 512. batch_normalization_4 output shape (None,8,8,256) and param is 1024.

For the type of MaxPooling2D for max_pooling2d layer output shape is (None, 64,64,32) and the parameter is 0. for max_pooling2d_1 layer output shape is (None, 32,32,64) and the parameter is 0. for max_pooling2d_2layer output shape is (None, 16,16,64) and the parameter is 0. for max_pooling2d_3layer output shape is (None, 8,8,128) and the parameter is 0. for max_pooling2d_4layer output shape is (None, 4,4,56) and the parameter is 0.

For the type of Dropout- for dropout layer output shape is (None,64,64,64) and param is 0. for dropout_1 layer output shape is (None,32,32,64) and param is 0. for dropout_2 layer output shape is (None,16,16,128) and param is 0. for dropout_3 layer output shape is (None,8,8,256) and param is 0. for dropout_4 layer output shape is (None,4096) and param is 0.

For type of Flatten for flatten layer output shape is (None, 4096) and param is 0 and for the dense layer output shape is (None, 512) and parameter is 2097664. For dense-1 layer output shape is (None, 1) and param is 513.

In model summary total Params is 2525121, among them trainable params is 2524003 and non-trainable params is 1088.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 8, 8, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 32768)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               16777728
_________________________________________________________________
dense_3 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 129
=================================================================
Total params: 31,558,209
Trainable params: 16,843,521
Non-trainable params: 14,714,688
```

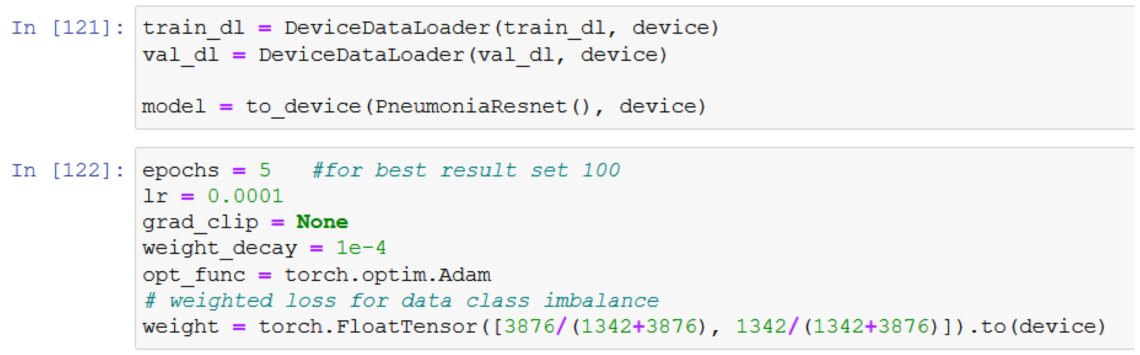
For VGG-16 sequential_1 model, for vgg layer output shape is (None, 8, 512) and the param is 14714688. For the flatten_1 layer output shape (None, 32768) and param is 0. For the dense_2 layer the output shape is (None, 512) and the output shape is 16777728. For the dense_3 the output shape is (None, 128) and param is 65664. For the dense_4 the output shape is (none, 1) and the param is 129.

Here for VGG model total param is 31558209, trainable params is 16843521 and non-trainable param is 14714688.

Learning Scheme

The training uses the mini batch gradient descent method with momentum, the batch size is 32, the parameter multiplied by the training regular term is set to 5*10-4, the dropout rate is set to 0.2, and the learning rate is set to le-4. The experiment was (28 epochs) early stopping, steps per epoch train_X.shape[0]//32, verbose set to 2, validation step to test_X.shape[0]//32. First train the A network with random initialization parameters, and then initialize the parameters of the trained A network. The first 4 layers of the deep network and the fully connected layer are used. The remaining layers are positive with a mean value of 0 and a variance of 0.000001. The state distribution is initialized randomly, and biases is initialized to 0. There may be methods that directly initialize parameters without pre-training parameters. In order to input image of 224x224, the original image is selected to be scaled isotopically and then randomly cropped. In order to expand the data set, the cropped image is randomly horizontally flipped and random RGB color conversion

The training uses the adam optimizer, the batch size is 32, the dropout rate is set to 0.2, and the learning rate is set to le-4. The experiment was (27 epochs) early stopping, steps per epoch train_X. shape[0]//32, verbose set to 2, validation step to test_X.shape[0]//32. First train the A network with random initialization parameters, and then initialize the parameters of the trained A network. The first 4 layers of the deep network and the fully connected layer are used. The remaining layers are positive with a mean value of 0 and a variance of 0.000001. The state distribution is initialized randomly, and biases is initialized to 0. There may be methods that directly initialize parameters without pre-training parameters. In order to input image of 224x224, the original image is selected to be scaled isotopically and then randomly cropped. In order to expand the data set, the cropped image is randomly horizontally flipped and random RGB color conversion



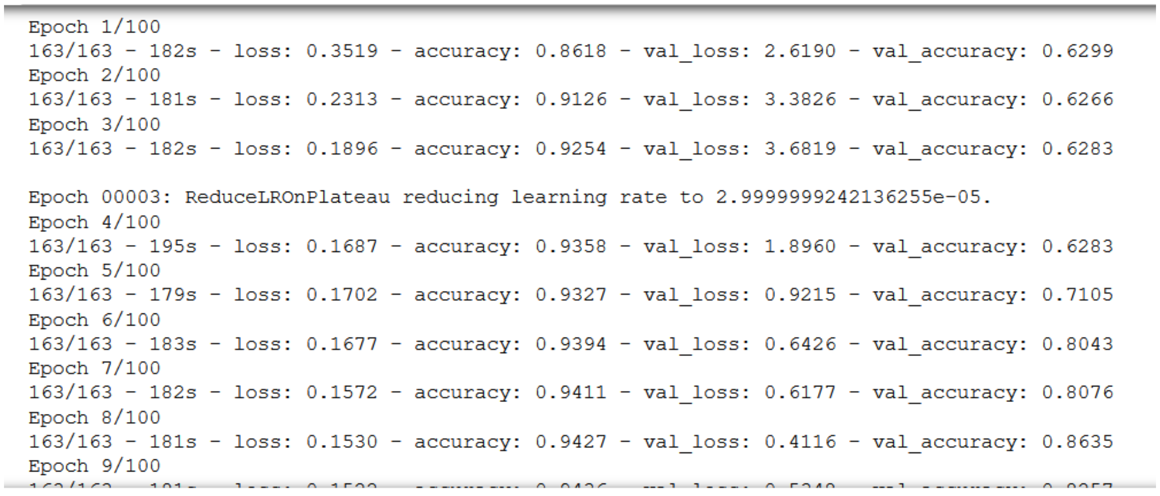
```
In [121]: train_dl = DeviceDataLoader(train_dl, device)
          val_dl = DeviceDataLoader(val_dl, device)

          model = to_device(PneumoniaResnet(), device)

In [122]: epochs = 5   #for best result set 100
          lr = 0.0001
          grad_clip = None
          weight_decay = 1e-4
          opt_func = torch.optim.Adam
          # weighted loss for data class imbalance
          weight = torch.FloatTensor([3876/(1342+3876), 1342/(1342+3876)]).to(device)
```

Here ensure that the data and the models parameters(weight and biases) are on the same device (in my case GPU). We can reuse the to_device function to move the model’s parameters to the right device. Configuration batch size, learning rate etc. epochs set 5. Learning rate is 0.001, epochs set to 5 and best training loss is 1.

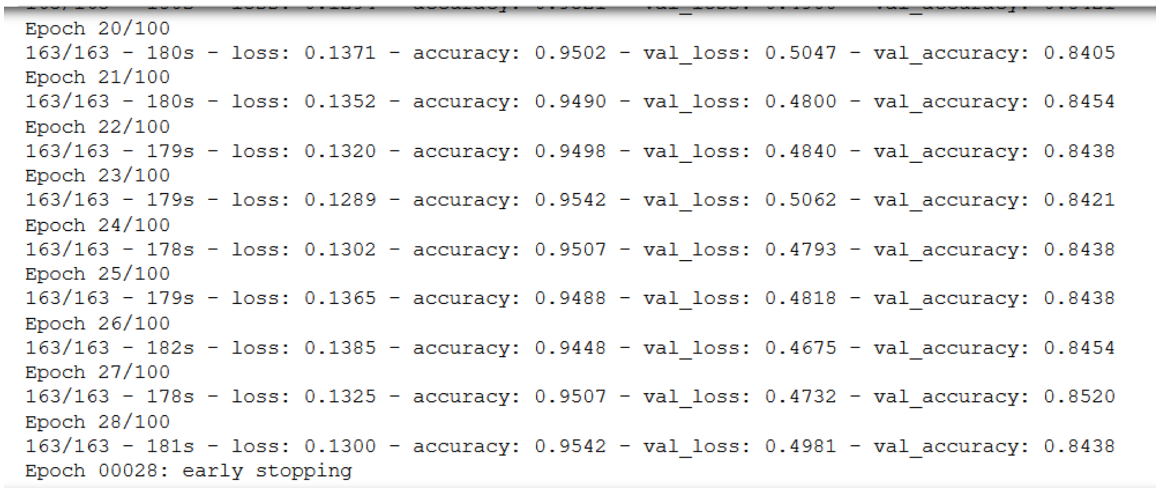
Experimental Result

Initial experimental results:

In the initial experiment, a part of the dataset and verification set were loaded and the number of trainings was 28 times. The experimental results are shown in the below figure:



```
Epoch 1/100
163/163 - 182s - loss: 0.3519 - accuracy: 0.8618 - val_loss: 2.6190 - val_accuracy: 0.6299
Epoch 2/100
163/163 - 181s - loss: 0.2313 - accuracy: 0.9126 - val_loss: 3.3826 - val_accuracy: 0.6266
Epoch 3/100
163/163 - 182s - loss: 0.1896 - accuracy: 0.9254 - val_loss: 3.6819 - val_accuracy: 0.6283

Epoch 00003: ReduceLROnPlateau reducing learning rate to 2.9999999242136255e-05.
Epoch 4/100
163/163 - 195s - loss: 0.1687 - accuracy: 0.9358 - val_loss: 1.8960 - val_accuracy: 0.6283
Epoch 5/100
163/163 - 179s - loss: 0.1702 - accuracy: 0.9327 - val_loss: 0.9215 - val_accuracy: 0.7105
Epoch 6/100
163/163 - 183s - loss: 0.1677 - accuracy: 0.9394 - val_loss: 0.6426 - val_accuracy: 0.8043
Epoch 7/100
163/163 - 182s - loss: 0.1572 - accuracy: 0.9411 - val_loss: 0.6177 - val_accuracy: 0.8076
Epoch 8/100
163/163 - 181s - loss: 0.1530 - accuracy: 0.9427 - val_loss: 0.4116 - val_accuracy: 0.8635
Epoch 9/100
```



```
Epoch 20/100
163/163 - 180s - loss: 0.1371 - accuracy: 0.9502 - val_loss: 0.5047 - val_accuracy: 0.8405
Epoch 21/100
163/163 - 180s - loss: 0.1352 - accuracy: 0.9490 - val_loss: 0.4800 - val_accuracy: 0.8454
Epoch 22/100
163/163 - 179s - loss: 0.1320 - accuracy: 0.9498 - val_loss: 0.4840 - val_accuracy: 0.8438
Epoch 23/100
163/163 - 179s - loss: 0.1289 - accuracy: 0.9542 - val_loss: 0.5062 - val_accuracy: 0.8421
Epoch 24/100
163/163 - 178s - loss: 0.1302 - accuracy: 0.9507 - val_loss: 0.4793 - val_accuracy: 0.8438
Epoch 25/100
163/163 - 179s - loss: 0.1365 - accuracy: 0.9488 - val_loss: 0.4818 - val_accuracy: 0.8438
Epoch 26/100
163/163 - 182s - loss: 0.1385 - accuracy: 0.9448 - val_loss: 0.4675 - val_accuracy: 0.8454
Epoch 27/100
163/163 - 178s - loss: 0.1325 - accuracy: 0.9507 - val_loss: 0.4732 - val_accuracy: 0.8520
Epoch 28/100
163/163 - 181s - loss: 0.1300 - accuracy: 0.9542 - val_loss: 0.4981 - val_accuracy: 0.8438
Epoch 00028: early stopping
```

Figure: Epoch is 28 training results

After 28 iterations, it is found that the accuracy rate on the verification set is 95.42%, loss 13.00%, validation loss 49.41% and validation accuracy is 84.38%. Through the experimental results, we can find that the results are actually very unstable. We found that the verification set has only 28 pictures and the verification set is relatively small.

The we visualization the data and the final results was 84.13%. But result of this experiment are not too bad but not good.

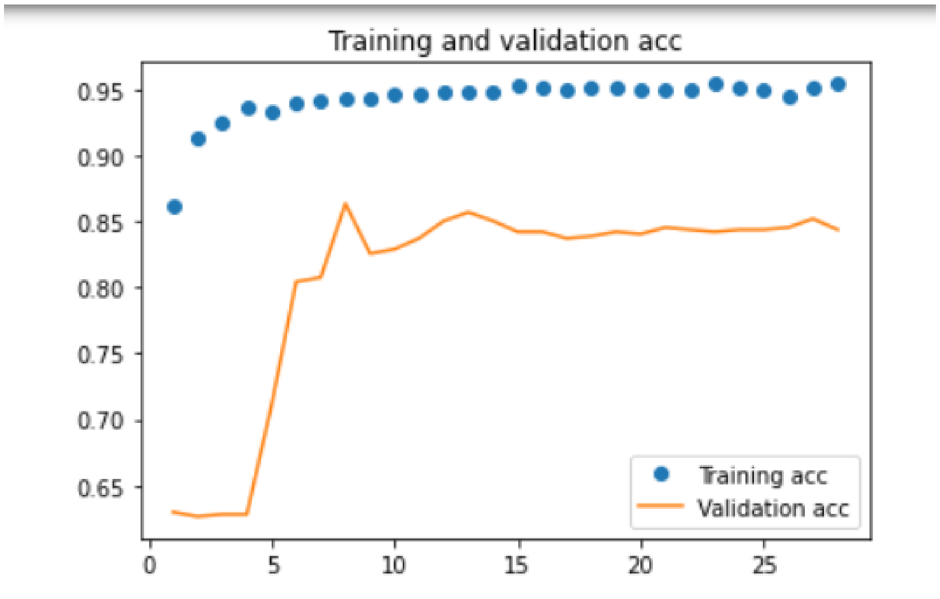


Figure 15 Training and validation Accuracy

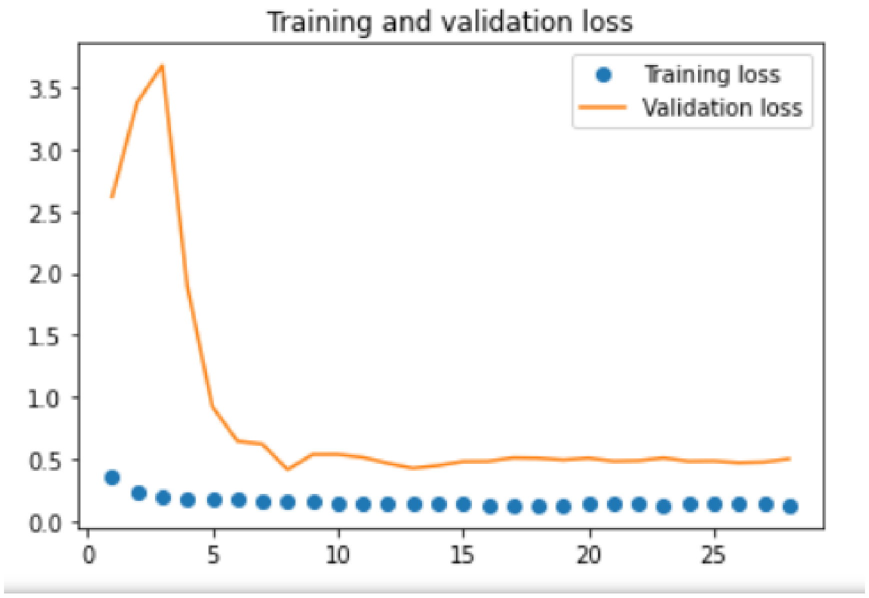
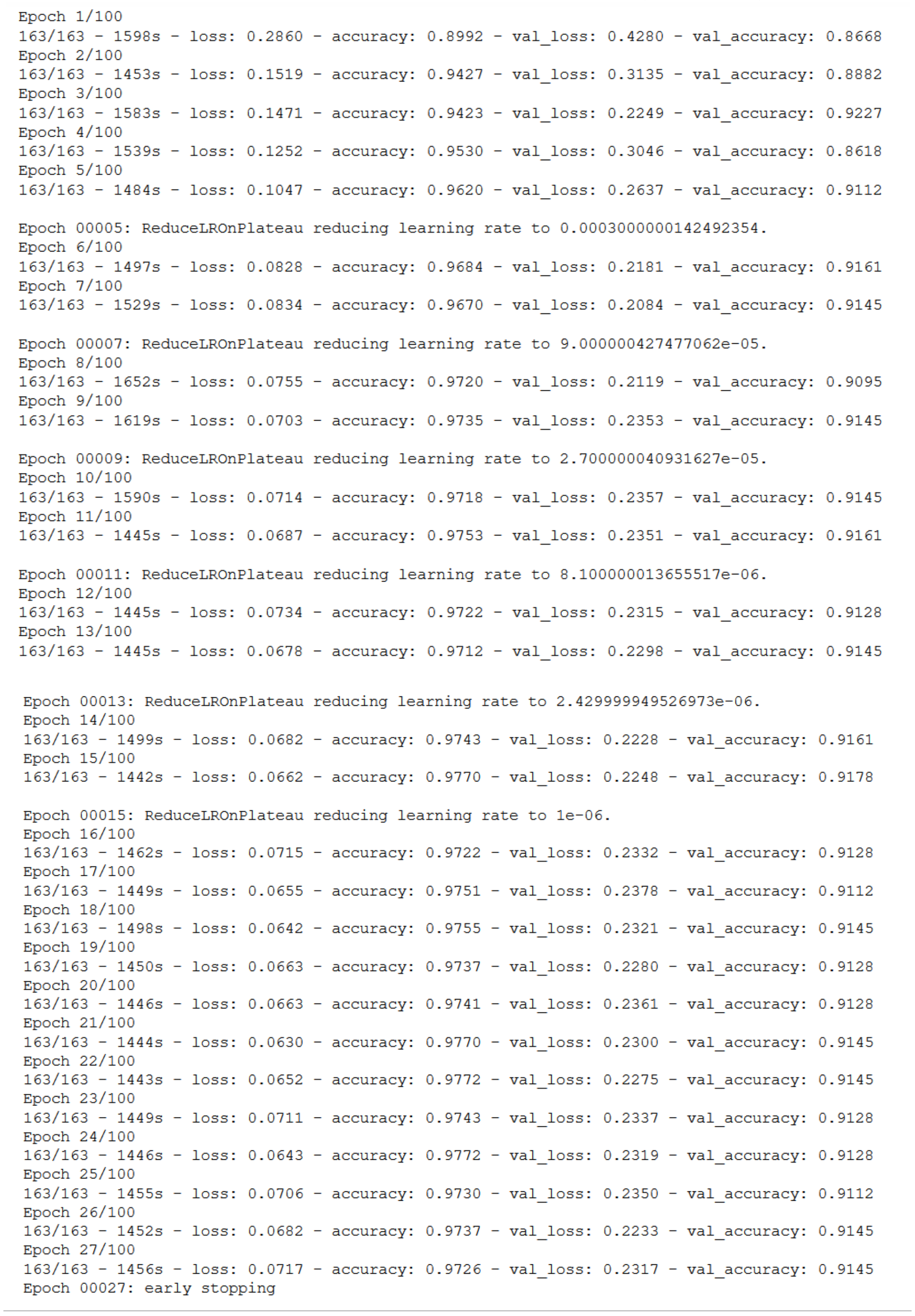


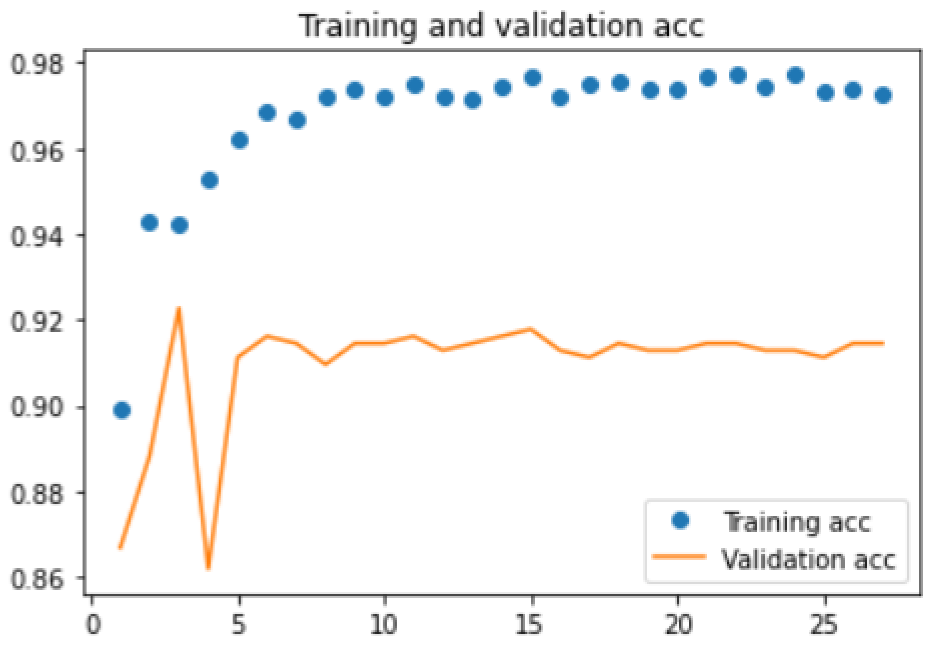
Figure 17 Training and validation Loss

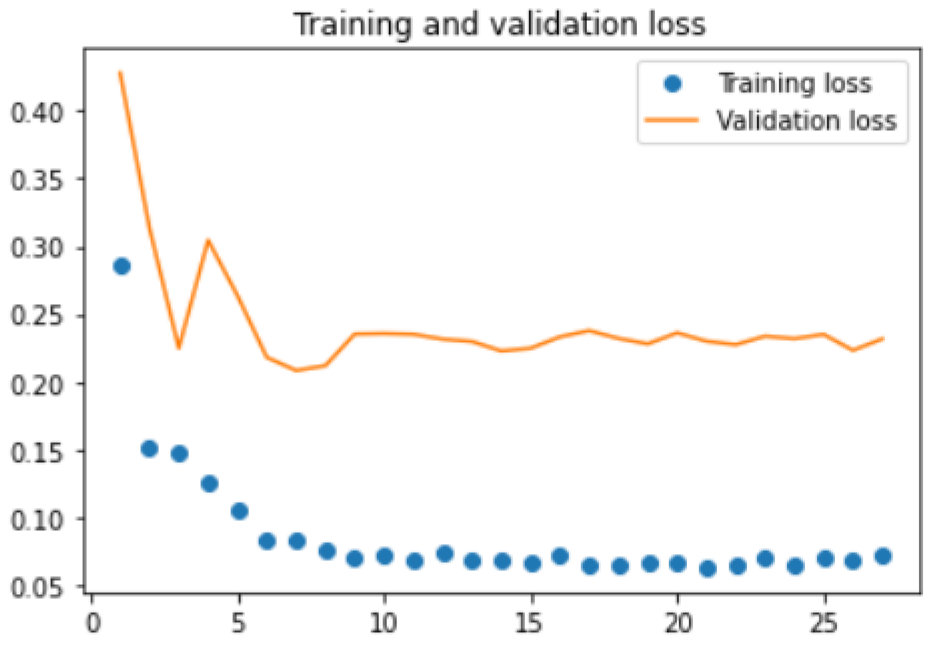
VGG-16 Result

After carefully adjusted the model, re-adjust the parameters, image size and increase the batch size the result was 91.34%. this result is good enough.



```
Epoch 1/100
163/163 - 1598s - loss: 0.2860 - accuracy: 0.8992 - val_loss: 0.4280 - val_accuracy: 0.8668
Epoch 2/100
163/163 - 1453s - loss: 0.1519 - accuracy: 0.9427 - val_loss: 0.3135 - val_accuracy: 0.8882
Epoch 3/100
163/163 - 1583s - loss: 0.1471 - accuracy: 0.9423 - val_loss: 0.2249 - val_accuracy: 0.9227
Epoch 4/100
163/163 - 1539s - loss: 0.1252 - accuracy: 0.9530 - val_loss: 0.3046 - val_accuracy: 0.8618
Epoch 5/100
163/163 - 1484s - loss: 0.1047 - accuracy: 0.9620 - val_loss: 0.2637 - val_accuracy: 0.9112

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
Epoch 6/100
163/163 - 1497s - loss: 0.0828 - accuracy: 0.9684 - val_loss: 0.2181 - val_accuracy: 0.9161
Epoch 7/100
163/163 - 1529s - loss: 0.0834 - accuracy: 0.9670 - val_loss: 0.2084 - val_accuracy: 0.9145

Epoch 00007: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
Epoch 8/100
163/163 - 1652s - loss: 0.0755 - accuracy: 0.9720 - val_loss: 0.2119 - val_accuracy: 0.9095
Epoch 9/100
163/163 - 1619s - loss: 0.0703 - accuracy: 0.9735 - val_loss: 0.2353 - val_accuracy: 0.9145

Epoch 00009: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.
Epoch 10/100
163/163 - 1590s - loss: 0.0714 - accuracy: 0.9718 - val_loss: 0.2357 - val_accuracy: 0.9145
Epoch 11/100
163/163 - 1445s - loss: 0.0687 - accuracy: 0.9753 - val_loss: 0.2351 - val_accuracy: 0.9161

Epoch 00011: ReduceLROnPlateau reducing learning rate to 8.100000013655517e-06.
Epoch 12/100
163/163 - 1445s - loss: 0.0734 - accuracy: 0.9722 - val_loss: 0.2315 - val_accuracy: 0.9128
Epoch 13/100
163/163 - 1445s - loss: 0.0678 - accuracy: 0.9712 - val_loss: 0.2298 - val_accuracy: 0.9145


Epoch 00013: ReduceLROnPlateau reducing learning rate to 2.429999949526973e-06.
Epoch 14/100
163/163 - 1499s - loss: 0.0682 - accuracy: 0.9743 - val_loss: 0.2228 - val_accuracy: 0.9161
Epoch 15/100
163/163 - 1442s - loss: 0.0662 - accuracy: 0.9770 - val_loss: 0.2248 - val_accuracy: 0.9178

Epoch 00015: ReduceLROnPlateau reducing learning rate to 1e-06.
Epoch 16/100
163/163 - 1462s - loss: 0.0715 - accuracy: 0.9722 - val_loss: 0.2332 - val_accuracy: 0.9128
Epoch 17/100
163/163 - 1449s - loss: 0.0655 - accuracy: 0.9751 - val_loss: 0.2378 - val_accuracy: 0.9112
Epoch 18/100
163/163 - 1498s - loss: 0.0642 - accuracy: 0.9755 - val_loss: 0.2321 - val_accuracy: 0.9145
Epoch 19/100
163/163 - 1450s - loss: 0.0663 - accuracy: 0.9737 - val_loss: 0.2280 - val_accuracy: 0.9128
Epoch 20/100
163/163 - 1446s - loss: 0.0663 - accuracy: 0.9741 - val_loss: 0.2361 - val_accuracy: 0.9128
Epoch 21/100
163/163 - 1444s - loss: 0.0630 - accuracy: 0.9770 - val_loss: 0.2300 - val_accuracy: 0.9145
Epoch 22/100
163/163 - 1443s - loss: 0.0652 - accuracy: 0.9772 - val_loss: 0.2275 - val_accuracy: 0.9145
Epoch 23/100
163/163 - 1449s - loss: 0.0711 - accuracy: 0.9743 - val_loss: 0.2337 - val_accuracy: 0.9128
Epoch 24/100
163/163 - 1446s - loss: 0.0643 - accuracy: 0.9772 - val_loss: 0.2319 - val_accuracy: 0.9128
Epoch 25/100
163/163 - 1455s - loss: 0.0706 - accuracy: 0.9730 - val_loss: 0.2350 - val_accuracy: 0.9112
Epoch 26/100
163/163 - 1452s - loss: 0.0682 - accuracy: 0.9737 - val_loss: 0.2233 - val_accuracy: 0.9145
Epoch 27/100
163/163 - 1456s - loss: 0.0717 - accuracy: 0.9726 - val_loss: 0.2317 - val_accuracy: 0.9145
Epoch 00027: early stopping
```

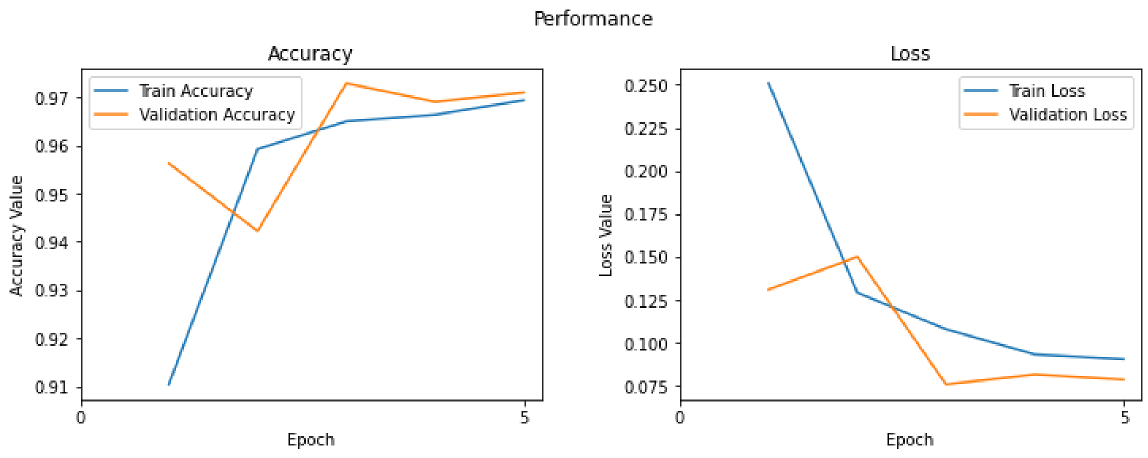




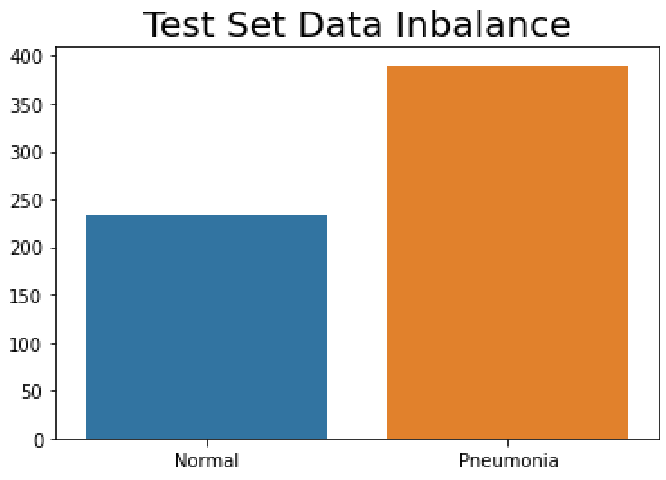
With batch size 32 and epoch 100 the program was early stop at epoch 27. Accuracy was 97.26%, loss 7.17% validation loss 23.17% and validation accuracy 91.45%.

Resnet-50 Result

```
Epoch [1], train_loss: 0.2508, train_acc: 0.9105, val_loss: 0.1309, val_a
cc: 0.9563
Epoch [2], train_loss: 0.1291, train_acc: 0.9592, val_loss: 0.1500, val_a
cc: 0.9422
Epoch [3], train_loss: 0.1079, train_acc: 0.9650, val_loss: 0.0758, val_a
cc: 0.9728
Epoch [4], train_loss: 0.0932, train_acc: 0.9663, val_loss: 0.0816, val_a
cc: 0.9690
Epoch [5], train_loss: 0.0905, train_acc: 0.9693, val_loss: 0.0786, val_a
cc: 0.9709
```



Plot of the accuracy and loss for the training and validation data gives us and idea of how our model is performing(underfitting, overfitting)

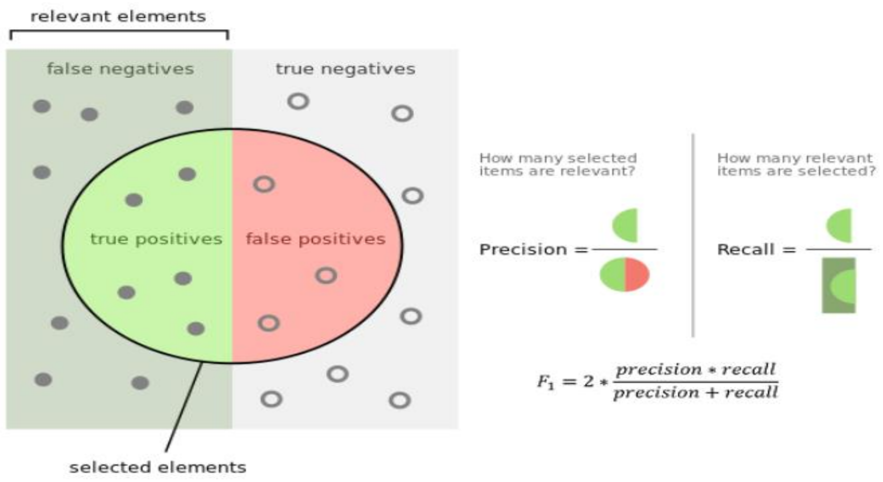


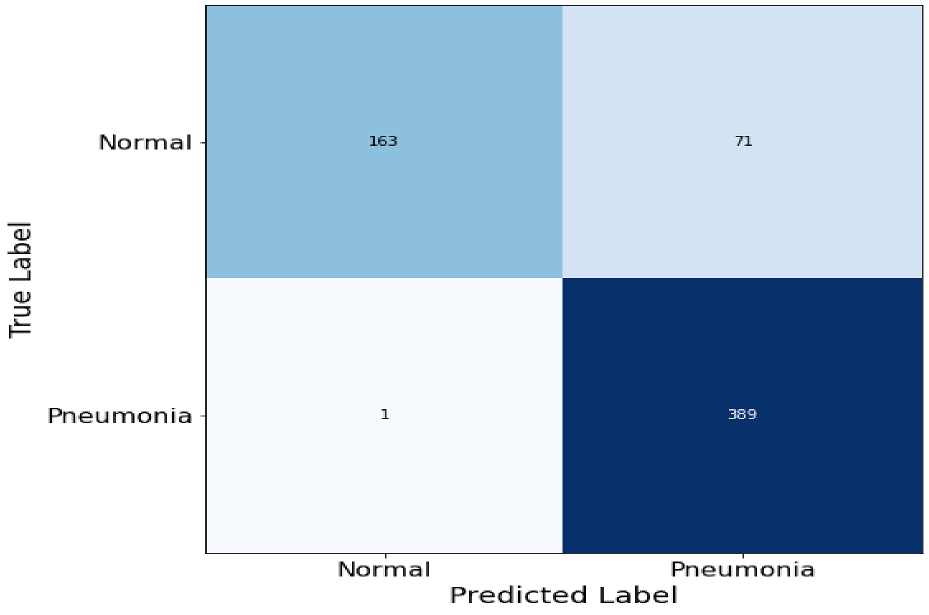
This figure test our model on the test set and it shows how well it performs on data that it has not been seen before. This is important as we want a model that can generalize to othr datasets other than what it is tained on. A model that only does well on the training and validation dataset but not on a testing dataset is not a useful one.

In the next figure we get a measure of how well our model is performing by evaluating several metrics of the predictions against the actual target labes.

Here accuracy is not a good evaluation metric when there is huge data class imbalance, imagine that if we have 100 samples: 99 pneumonia and 1 normal, then a model that predicts everything as pneumonia will get an accuracy of 99%. In this case, its better to look at precision and recall, and their harmonic mean, the F1 score.

This can be visualized using a confusion matrix as well.


relevant elements
false negatives
true negatives
true positives
false positives
selected elements
How many selected items are relevant?
How many relevant items are selected?
Precision =
Recall =
$F_1 = 2 * \frac{precision * recall}{precision + recall}$



Precision is lower than recall, but this is expected due to the data class imbalance. The weightings can perhaps we tuned to improve the precision. However, recall and precision is balance and one has to be sacrificed for the other.

Conclusion

In conclusion, our model performed reasonably well, as it manages to predict most of the pneumonia images. This is important as in healthcare diagnosis, accurate prediction of diseases saves lives. However, false positives can also increase the cost of healthcare, as more people with diseases are diagnosed with diseases. It can also lead to panic and affect people's physical and mental well-being. Better performance can be achieved with better tuning of the model hyperparameters. This is an iterative process, with lots of trial and error.

The results show that the recognition accuracy of the model on the validation set reaches 95.42%. The accuracy of the VGG-16 was 91.35%; while the RestNet-50 network model trained and got an accuracy of 88.00%. Comparing the result of this test, it is not difficult to find that VGG-16 is better effect in the recognition of CT images of pneumonia.

Below are some of the difficulties that I faced for this project:

- Data class imbalance: This is quite prevalent in the real world and as data scientists, we should all learn to embrace it and find ways to get around this problem. Usually, one way is to simply collect more data for the under sampled class. However, this is not possible for this dataset and especially for healthcare data, which is very difficult to collect and share. In this kernel, I used weighted loss to give more weights to the loss of the normal images. This weighting can be tuned as well to achieve the optimal performance. Stratified sampling is also important in this case, but pytorch does not yet have any built-in functions.
- Overfitting: This dataset is prone to overfitting as seen by the train and validation plots. Therefore, I only selected the model weights that achieved the best/lowest validation loss. However, sometimes the best validation loss is achieved in the early epochs, and it performs very badly on the test set, probably due to class imbalance when batching or sampling. For that reason, I have chosen to select the best validation loss only after certain epochs. I am not sure if this is a valid method. Please let me know if anyone has any comments on it!
- Hardware problem: this dataset was quite big for 8GB memory so that whenever I set more batch size it was difficult to train and for epochs training it was time consuming.