中国传媒大学
COMMUNICATION UNIVERSITY OF CHINA

题目：Analysis of PneumoniaMNIST Image Data Based on Multiple Convolutional Neural Networks

学年学期：2023 年秋季学期

课程名称：人工智能进阶：强化学习与深度学习

课程编号：2131030217

课程序号：[01]

任课教师：

姓　　名：贺君 徐棱霄 郭慧慧 侯宇欣 曹福滨

学　　号：

评分区域（由阅卷老师填写）：

结课成绩：

总评成绩：

提交时间：2023 年 12 月 26 日

Contents

1 Introduction

Pneumonia is an inflammation of the lung parenchyma caused by pathogenic bacteria, physical and chemical factors, immunologic injury, and various medications.Every year, more than 800,000 children under the age of five die from pneumonia. over 2200 people die. More than 1400 children per 100,000 become affected by pneumonia. Pulmonary infections, such as pneumonia, were the second greatest reason for death in 2013, according to the Global Burden of Disease Study.

Chest X-ray imaging, computed tomography (CT), and magnetic resonance imaging (MRI) are all diagnostic radiological techniques for pulmonary disease, with chest X-ray scanning being the most efficient and cost-effective because it is far more accessible and portable in hospitals, and it exposes sick people to lower doses of radioactivity. However, even for multiple skilled and experienced medical physicians, analysing pneumonia using X-ray snapshots remains a difficult task because X-ray images contain comparable area statistics for unique ailments, such as lung cancer. As a result,diagnosing pneumonia with conventional procedures is time-consuming and energy-intensive, and it is impossible to diagnose whether or not a patient has pneumonia in a uniform manner.

MedMNIST is a large-scale MNIST-like dataset collection of standardized biomedical images, including 12 datasets for 2D and 6 datasets for 3D images all processed into small sizes with corresponding classification labels. PneumoniaMNIST is one of the 2D datasets focused on pediatric chest X-Ray images for binary-class classification of pneumonia. It contains 5,856 images, pre-processed and resized.

As a consequence,we train our models on a random class balanced 10-sample subset of the Pneumonia MNIST training dataset to autonomously diagnosis pneumonia using X-ray pictures and evaluate them on a 1000-sample subset. We address this problem in two conditions:

Condition1:Learning with limited data without external data

This condition involves exclusively using the 10 training samples. Since no external data is allowed, we will explore different model architectures, data augmentation techniques and custom libraries. More specifically, we will explore different Residual Neural Networks, automated data augmentation with a reduced search space, tuning-free data augmentation and automated deep learning.

Condition2:Learning with limited data with external data

This condition is more lenient. Besides the 10 samples, we can use any external data or pre-trained models given that they were not trained on or disclosed information obtained from the same 10 samples. We considered various forms of fine-tuning and transfer learning with pre-trained models.

2 Related Work

Studies on pneumonia detection have mostly utilized large datasets.A study on pneumonia detection is [1], which utilizes a large dataset of 5216 images.The study [2] utilized the new chest X-ray-8 dataset comprising 108,948 chest X-ray images of

32,717 unique patients, gathered from 1992 to 2015, with eight illness image labels and multiple labels on each image.However,medical image analysis is one of many application domains that do not have access to big data [3],So we learned to explore the possibility of accurate classification with small samples of the Pneumonia MNIST dataset, with the focus on improving the detection efficiency.

AutoGluon is a framework for AutoML, which aims at automating feature engineering, model selection and training.The framework minimizes the effort of data scientists and engineers [4] by preprocessing data, performing feature selection, model selection and training with only a few lines of code. Training the baseline model on a small dataset only takes a few minutes. So the framework is well suited to a small dataset where we only have 10 training samples.

Most deep convolutional neural networks tend to overfit if data is sparse[3]. Also due to medical image analysis does not have access to big data [3], data augmentation comes in handy.

Data augmentation is a popular data-space solution to the problem of limited data [3]. Ref.[5] used a deep CNN architecture with data augmentation for pneumonia. CNN is studied with different architectures to achieve the best results, and results are analyzed with and without data augmentation. Results suggest that, with the augmentation approach, the accuracy is 83.38%, while with the original dataset, it is only 80.25%. With the expectation of creating a good training algorithm to improve accuracy using only 10 class-balanced samples randomly selected from the Pneumonia MNIST dataset,we show evidence of how ResNet18 with RandAugment augmentation technique outperformed the baseline CNN and hyperparameters by 5%.Besides,compared to the Automatic Augmentation，RandAugment removes both of these obstacles [6]. RandAugment has a significantly reduced search space which allows it to be trained on the target task with no need for a separate proxy task. Furthermore, due to its parameterization, its regularization strength may be tailored to different model and dataset sizes[6].

Deep convolutional neural networks are useful in machine vision tasks. These have created advances in many field like Agriculture 24; medical disease diagnosis [7,8]; and industry[9].Some of the powerful and most used deep convolutional networks conclude ResNet [10] and VGG [11].

For example,ResNet50 is a CNN model with 50 layers. The ImageNet database which has been trained on over a million images provides a pre-trained version of the network that can be imported. Microsoft Research Asia suggested ResNet50 in late 2015 as a solution to the difficulties of training CNN models. As a result, creating a residual network with Keras for computer vision applications such as image classification is somewhat simplified [12]. In condition 1, we decided to compare performances between the baseline CNN and ResNet as we believe that ResNet makes it possible to explore additional features which a shallow convolutional network architecture may not capture[13].In condition 2, we decided to use the pre-trained ResNet model for our classification problem without the need to train from scratch as it has already been trained on large datasets such as ImageNet for image recognition

purposes [14]. We will also focus on freezing the initial layers since the size of the training data is small and the data similarity between Pneumonia MNIST Dataset and ImageNet is very low [15]. This can assist with network generalization and speed up convergence [14].

The VGG-16 was the 2014 ILSVR (ImageNet) competition winner. It is widely regarded as the most sophisticated vision model design currently accessible.The number 16 in VGG-16 denotes the presence of 16 weighted layers [16]. The VGG-16 network provides good accuracy even if the image datasets are minimal because of its massive training.Because only the models from the PyTorch repository are permitted for condition 2, our naive approach for exploring alternative solutions involves navigating the list of models [17]. A natural improvement over the baseline AlexNet model is by going deeper with VGGNet.

Transformer-based models are known to perform well on natural language processing (NLP) and computer vision (CV). We would like to explore this architecture in this project. Hugging Face Hub makes available a list of transformer models and the ViTForImageClassification in particular based on [18].

Vision Transformers have had a huge influence in CV. Its appearance challenges the previously dominant position of convolutional neural networks in the field of CV. Transformer architectures generally require a large amount of data to train; such pre-trained models tend to yield great results in NLP. Applying a similar strategy to computer vision, the ViT pre-trained model is expected to perform well in image classification tasks [19].

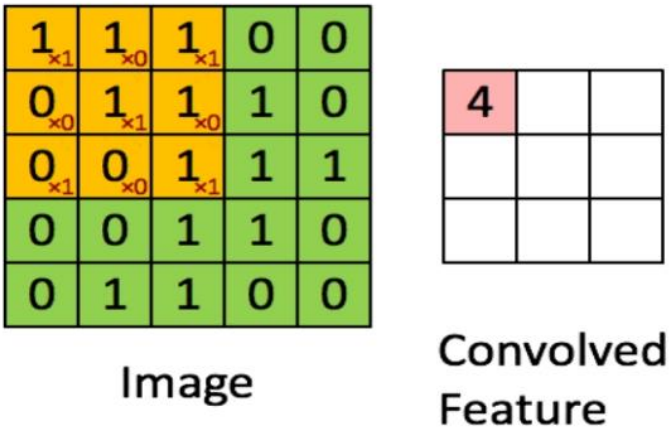
3 Methods

3.1 Base CNN model

The classic CNN neural network consists of three parts: convolutional layer, pooling layer and fully connected layer.If simple description: convolution layer is responsible for extracting local features in the image;Pooling layer is used to greatly reduce the magnitude of parameters (dimensionality reduction);The fully connected layer is similar to the part of a traditional neural network and is used to output the desired result.

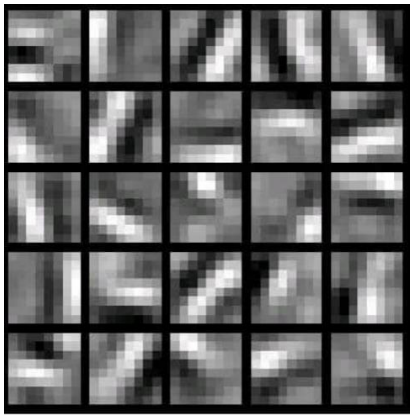
3.1.1 Convolution - Extracting features

The operation process of the convolutional layer is shown in the following figure, using a convolution kernel to scan the entire image:



This process can be understood as we use a filter (convolution kernel) to filter the various small regions of the image to obtain the eigenvalues of these small regions.

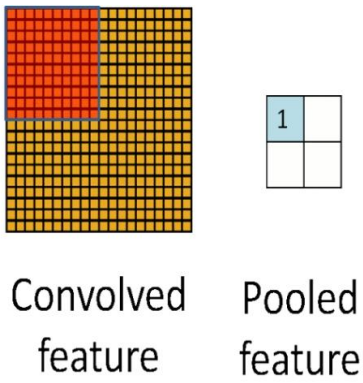
In specific applications, there are often multiple convolution kernels, and each convolution kernels can be considered to represent an image pattern. If the value of an image block convolved with the convolution kernels is large, it is considered that the image block is very close to the convolution kernels.If we design six convolution cores, it makes sense: we think there are six underlying texture patterns on the image, that is, we can paint an image with six basic patterns. Here are some examples of 25 different convolution kernels:



Summary: The convolution layer extracts the local features in the image through the filtering of the convolution kernel, which is similar to the feature extraction of human vision mentioned above.

3.1.2 Pooling layer - data dimension reduction to avoid overfitting

The pooling layer is simply subsampling, which can greatly reduce the dimension of the data.The process is as follows:



In the figure above, we can see that the original image is 20×20, we downsample it, the sampling window is 10×10, and finally downsample it into a 2×2 size feature map.

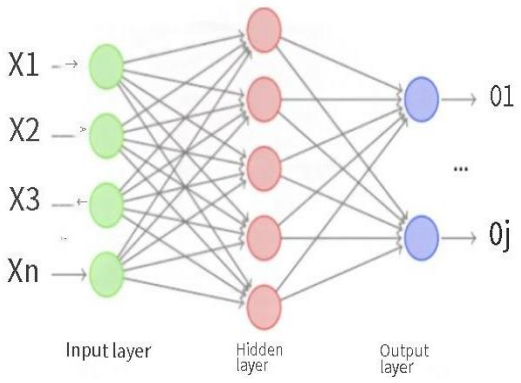
The reason for this is because even after the convolution is done, the image is still large (because the convolution kernel is relatively small), so in order to reduce the data dimension, downsampling is performed.

Summary: Pooling layer can reduce the data dimension more effectively than convolution layer, which can not only greatly reduce the amount of computation, but also effectively avoid overfitting.

3.1.3 Full connection layer - Output results

This part is the final step, where the data processed by the convolution layer and the pooling layer is fed into the fully connected layer to get the desired result.

After the convolution layer and pooling layer of dimensionality reduction of data, the fully connected layer can "run", otherwise the amount of data is too large, the calculation cost is high, and the efficiency is low.



3.2 Applications and Training Strategies for ResNet

Unlike the previous step, this time we used those 10 samples as well as external data for training or directly with pre-trained models (as long as they were not trained

on the same 10 samples or disclosed information). When pre-training the models, we used ResNet models from large datasets such as ImageNet.

3.2.1 Pre-training ResNet model selection

In Task 2, instead of training the model from scratch, we use a pre-trained ResNet model to solve our classification problem because it has already been trained on large datasets (e.g., ImageNet) for image recognition.

3.2.2 Freeze the initial layer

We focus on freezing the initial layers because of the small size of the training data, the very low data similarity between the Pneumonia MNIST dataset and ImageNet, and when using pre-trained models from large-scale datasets such as ImageNet on a small-scale dataset such as Pneumonia MNIST, the initial layers (usually the first few layers) learnt learned may be too generic and not match the features of the specific dataset for the task at hand. Freezing these initial layers can therefore prevent overfitting, help generalize the network, and accelerate convergence.
Impact: Freezing the initial layers accelerates model convergence and reduces the risk of overfitting, but may also result in the loss of some dataset-specific information, especially on small datasets, where this effect may be more significant.

3.2.3 Comparison of ResNet pre-training results

We will try to pre-train the ResNet module using the respective pre-training weights, but only the last layer and the fully connected layer will be trained.

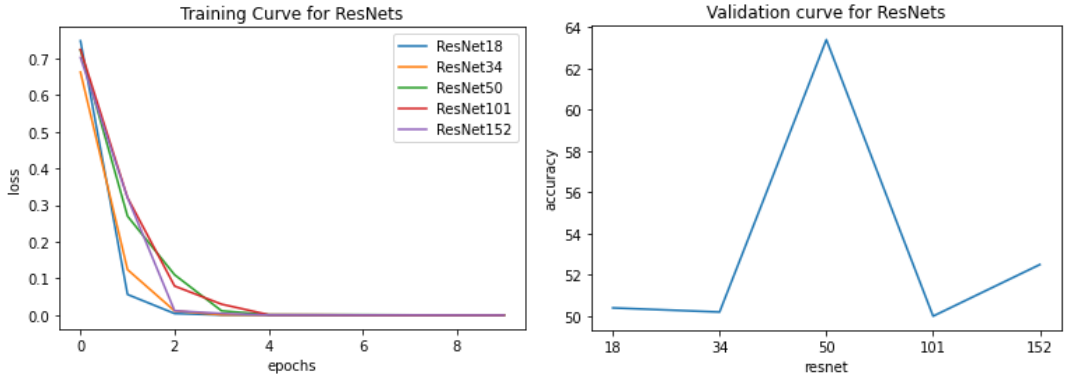


Figure 3-1 Pre-trained ResNets with all layers optimized

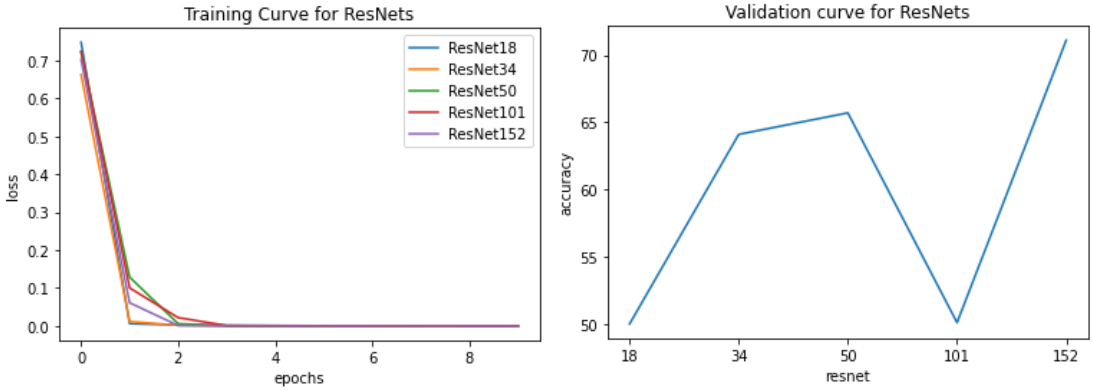


Figure 3-2 Pre-trained ResNets optimizing last layers only

From the above graph we can observe that when all layers are trained, the best model is ResNet50 at about 64%. When all layers except the last one are frozen, ResNet152 performs better with about 72%.

Table 3-1 Comparison between AlexNet & ResNet152 over 50 Seeds

Models	Accuracies	Run times
AlexNet	85.72	28 min 6s
ResNet152	68.75	8h 28min 58s

```
Val set: Average loss: 2.3518, Accuracy: 675/1000 (67.50%)

Val acc over 5 instances on dataset: pneumoniamnist 68.75 +- 12.86 (var: 165.46)
CPU times: user 8h 17min 29s, sys: 11min 47s, total: 8h 29min 17s
Wall time: 8h 28min 58s


Val acc over 5 instances on dataset: pneumoniamnist 85.72 +- 3.07 (var: 9.45)
CPU times: user 28min 3s, sys: 10.6 s, total: 28min 14s
Wall time: 28min 6s
```

Figure 0-3 Running results for AlexNet & ResNet152 over 50 Seeds

With more than 50 seeds, the accuracy of the base AlexNet model is still as high as 85.72%.We can conclude that the ResNet model underperforms compared to the base AlexNet model. In addition, training the ResNet152 model was very slow. Therefore, we did not adjust the model further.

3.3 VGGNet

Inspired by the performance of the AlexNet model, we intend to use an improved version of AlexNet.VGGNet is deeper compared to AlexNet, with more convolutional layers and a smaller convolutional kernel, which makes VGGNet more representational and better suited for extracting image features.

VGGNet takes a VGG block substructure from AlexNet and stacks the VGG blocks together to build a deeper version of AlexNet.We loaded VGG16 and VGG19 pre-trained models from the PyTorch repository.

3.3.1 VGGNet vs. AlexNet

Connections: VGG16 borrows some of the basic ideas and structure of AlexNet, but improves on some aspects. Both are deep convolutional neural networks that present effective methods for feature extraction and classification of images.

Distinctions: Compared to AlexNet's Local Response Normalization (LRN) layer, VGG16 uses more convolutional layers and fewer fully connected layers, and employs Batch Normalization to improve training speed and accuracy.

3.3.2 Network Architecture Diagram and Layer Parameters

The model structure of both VGG16 and VGG19 consists of convolutional, pooling and fully connected layers.

VGG16 has a total of 16 layers, including 13 convolutional layers and 3 fully connected layers. The convolutional layers use 3x3 convolutional kernels and the

pooling layers use 2x2 pooling kernels, and the dimensionality of the feature map decreases as the number of layers increases. The network architecture diagram of VGG16 can be represented as:

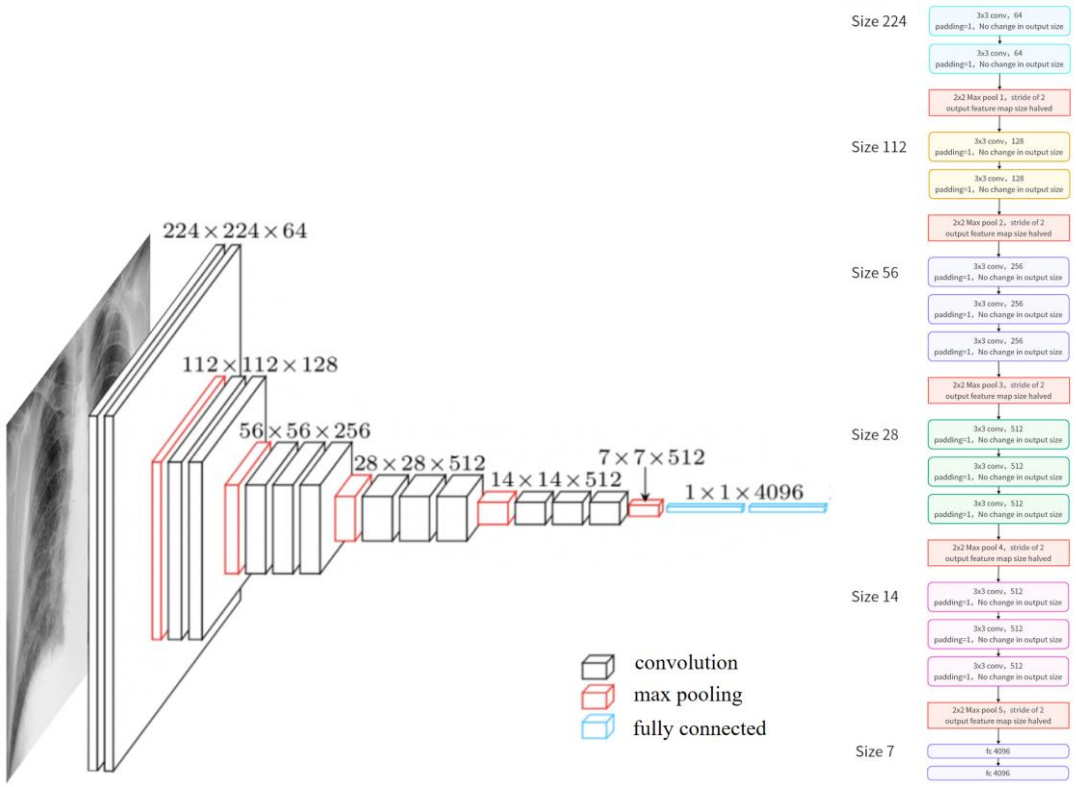


Figure 3-4 Network Architecture :VGG16

The network architecture diagram of VGG19 can be represented as:

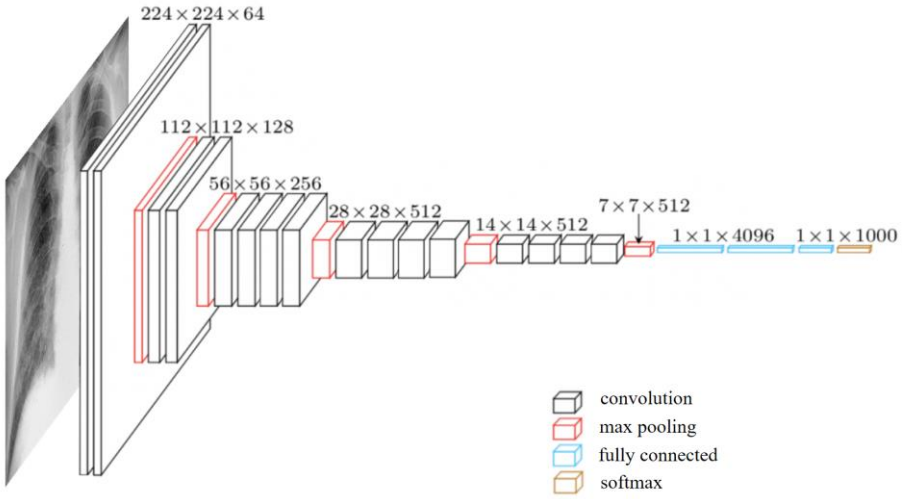


Figure 3-5 Network Architecture :VGG16

3.3.3 Model Selection and Training Parameters

We used an SGD optimizer to update the weights of the classification heads with an initial learning rate of 1e-3, momentum of 0.9, and weight decay of 0.005. for each seed, the model is trained for 200 epochs. training more epochs may lead to overfitting.

3.4 Application and Effectiveness of the ViT Model

3.4.1 Introduction and application of the ViT model

We prepared the images using ViT, a Transformer-based visual processing model. ViT uses the Transformer architecture, which includes the Self-Attention mechanism. The model classifies images by dividing them into fixed-size patches, then spreading these patches into sequences, and finally classifying the images through Transformer's multi-head Self-Attention mechanism and fully connected layers.
The general architecture of the ViT model is as follows:

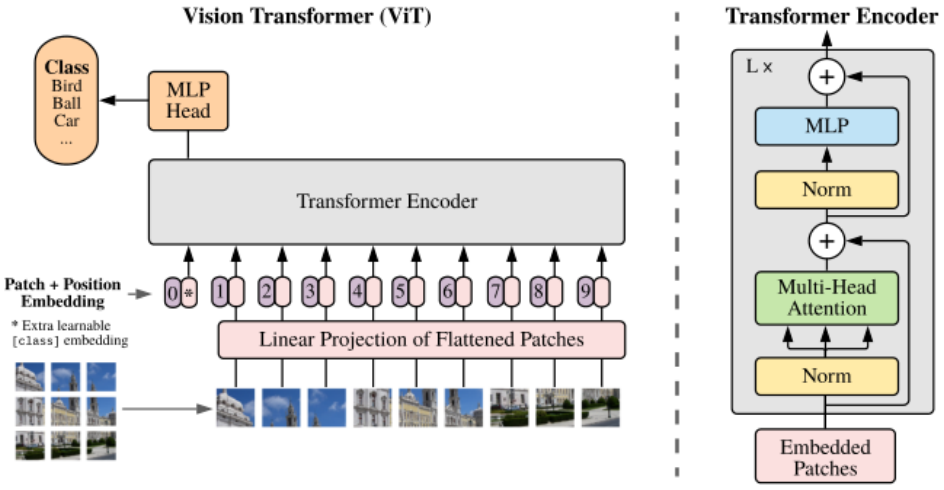


Figure 3-6 Architecture of the ViT

We leveraged the FeatureExtractor 'google/vit-base-patch16-224-in21k' from Hugging Face Hub to perform transformations such as cropping, normalization, and converting grayscale to RGB, and also added data augmentation, random enhancement, and flipping, after applying these transformations, we noticed that the loss values became un-convergent compared to the default, resulting in lower accuracy. We loaded the pre-trained "google/ vitbase -patch16-224-in21k" model and initialized ViTForImageClassification. the number of labels was set to 1 so that the model would create a classification header that outputs a single label.

Here's a comparison between the models trained with FeatureExtractor transformations and without.

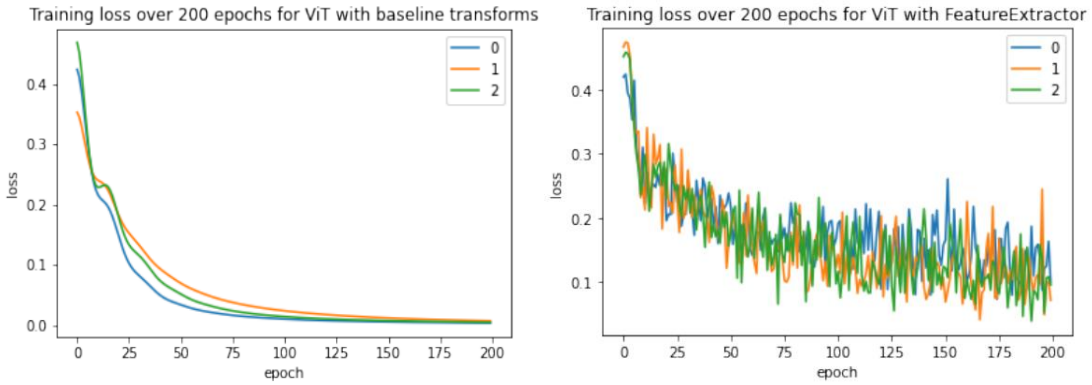
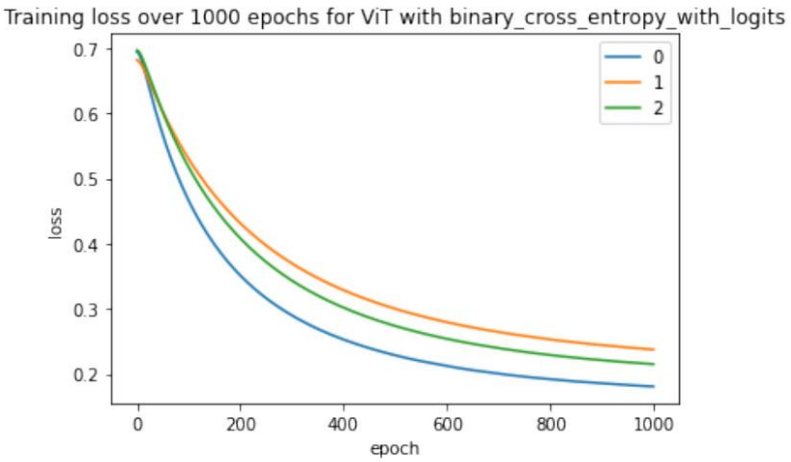


Figure 3-7 A comparison between the models trained with FeatureExtractor transformations and without

3.4.2 Relationship between loss values and verification accuracy

The following figure shows the loss computed by binary crossentropy with logits. We can get that: Although the loss value range is different, the results show a similar validation accuracy of 74.37 +- 0.29.



3.5 Loss Function

We used Binary Cross Entropy Loss as the loss function and trained the model with an optimizer to minimize this loss function and continuously update the model parameters to improve the prediction accuracy. A learning rate decay strategy (scheduler) is also used.

$$Binary\ Cross\ Entropy\ Loss = -\frac{1}{N}\sum_{i=1}^{N}(y_i * log(p_i) + (1 - y_i) * log(1 - p_i))$$

Where： N is the sample size; y_i is the true label of the i th sample (0 or 1); $p_i$is the probability that the model predicts this sample to be a positive category (with a value range of 0 to 1);

We tried the "Adam", "AdamW" and "SGD" optimizers with and without weights. The validation accuracy varies for different combinations of parameters such as model and optimizer. This implies that choosing the right parameters on a small sample dataset has a significant impact on the model performance.

4 Result and Analysis

4.1 Different models for better index

To result and analysis the CNN model, we try to find a better network than base CNN model. To give a more solid result, we compared how the model did over 50 different seeds. We experimented AutoGluon and Residual Neural Network based on the results from 50 random seeds. However, AutoGluon and Residual Neural Network failed both in training time and validation accuracy compared with the baseline CNN model in our experiments.

We plotted multiple graphs to help us better determine which one is the best. There were three model results in details as follows looking for the one which gave the best accuracy.

4.1.1 Base CNN model

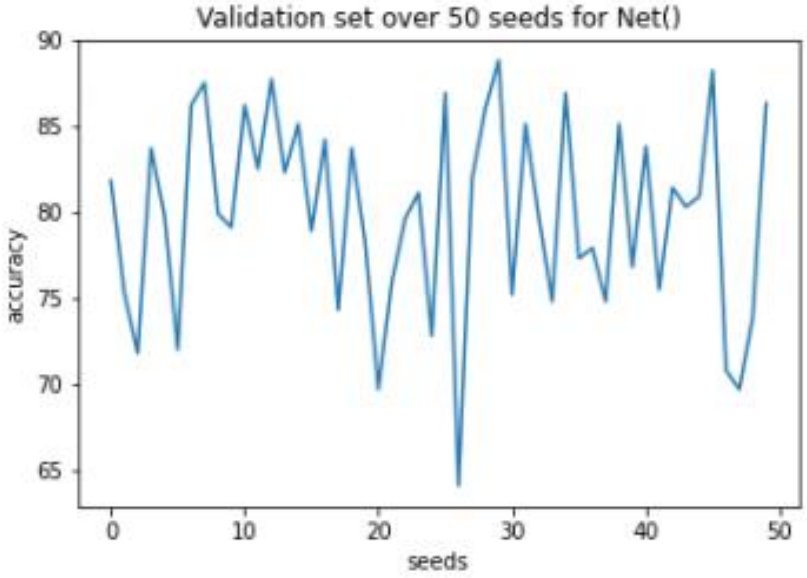


Figure 4-1 Validation of CNN over 50 seeds

On average the base CNN model gave an accuracy of 79.83% with a standard deviation of 5.71 and a variance of 32.65.

4.1.2 AutoGluon

As a framework separate from PyTorch, AutoGluon does not work with the provided starter code but requires a simple setup process. The Pneumonia MedMNIST dataset is saved as PNG files and a CSV file that records their splits (TRAIN, VALIDATION, TEST), image file locations and labels. The images were then loaded as an ImageDataset using AutoGluon API.

Because the MedMNIST dataset was saved with its own splits, the training process went along with this setup. 10 samples were randomly picked from the training set. However, the validation set only contains 135 samples and they were all used during model selection.

1.Result

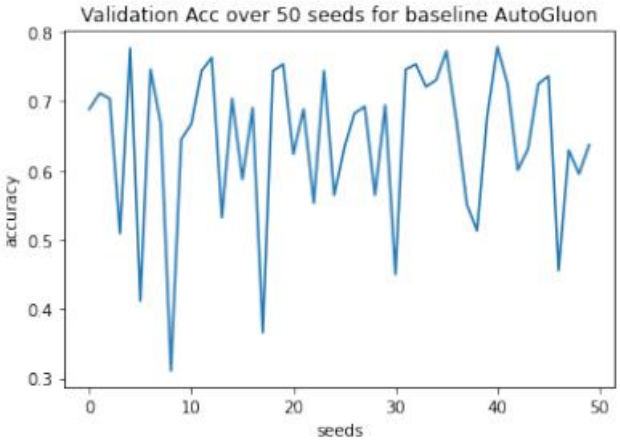


Figure 4-2 The accuracies of the baseline AutoGluon over 50 seeds.

Based on the results from 50 random seeds, the baseline model selection tends

to overfit the training data and perform poorly on validation data. All results were below 80% accuracy and hence unable to beat the baseline ConvNet benchmark 79.88 +- 5.73.

2.Analysis

AutoGluon provides automated hyperparameter search and model selection. The training process was completed in 23m 15s and thus failed both in training time and validation accuracy compared with the baseline CNN model (28.2s).

4.1.3 Residual Neural Network

In each of the models, the loss eventually came down to near zero.

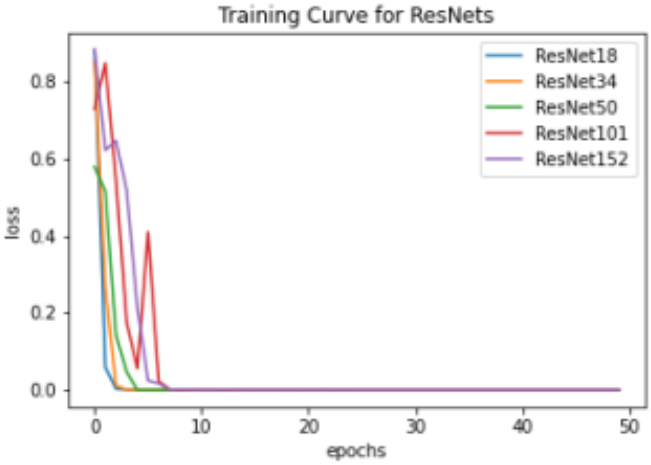


Figure 4-3 The loss of all ResNets

We experimented between the different variations of ResNet: ResNet18, ResNet34, ResNet50, ResNet101, ResNet152. Testing each of the trained models on the validation set, the results were then really different. ResNet18 seemed to have a better generalization and gave an accuracy of +70%. On the other hand, ResNet34 performed the worst at an accuracy of around 58%.

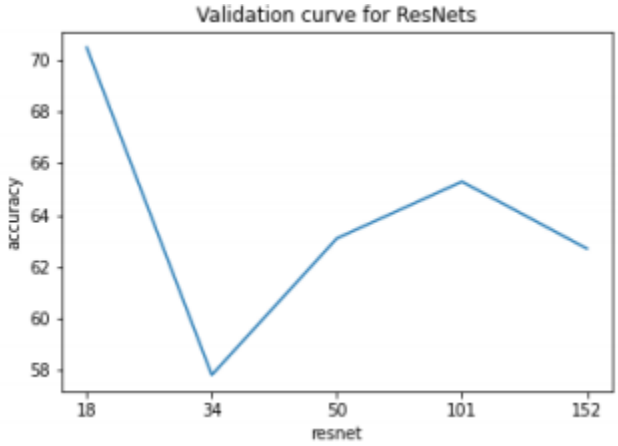


Figure 4-4 The accuracies of all ResNets

1.ResNet18 Result

On average the ResNet model gave an accuracy of 74.98% with a standard deviation of 6.74 and a variance of 45.40.

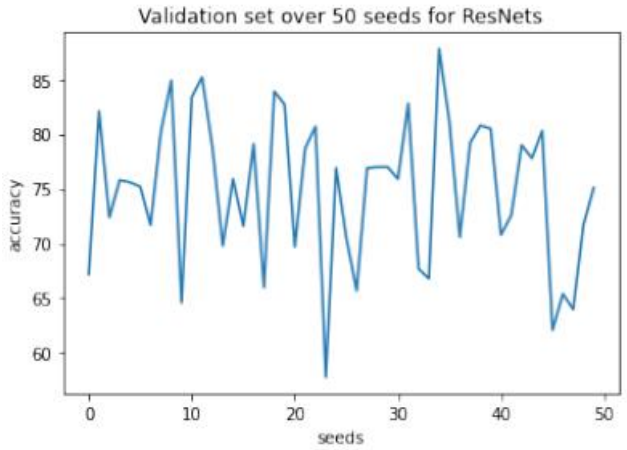


Figure 4-5 Validation of ResNet18 over 50 seeds

Table 4-1 Comparison between CNN & ResNet18 over 50 seeds

Models	Accuracies	Run times
Base CNN	79.83	41.5s
ResNet18	74.98	13min20s

As we can see in table 1, the CNN baseline model was still performing better than the ResNet.

2.ResNet18 Analysis

After doing some research, it is claimed that Residual Neural Networks outperformed CNN due to their residual blocks which not only overcame the vanishing gradient problem but also explore exclusive features that shallow networks may not. However, in our experiments, we have not yet exceeded the baseline model.

4.2 Different algorithms for better index

Although ResNet18 has not yet exceeded the baseline model in our experiments, we could improve it further by using data augmentation and hyperparameter tuning.

4.2.1 Data Augmentation

The idea behind data augmentation is to create more data. The images could be cropped, flipped or change in colours based on the available transformations. This will give the training data more variety, make the model more robust and help it generalize better.

We had then experimented with two different data augmentation techniques: RandAugment and TrivialAugmentWide to alleviate the obstacles of automatic augmentation by reducing the search space and by being tuning-free respectively.

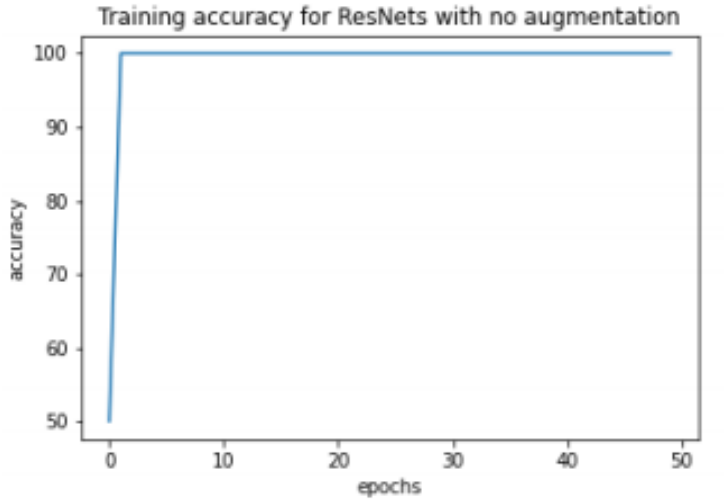


Figure 4-6 Training with no augmentation

1.Result

As we can see in Figure 6, training without data augmentation quickly reached 100% accuracy. However, with data augmentation, the model did not overfit and varied a lot more throughout the epochs.

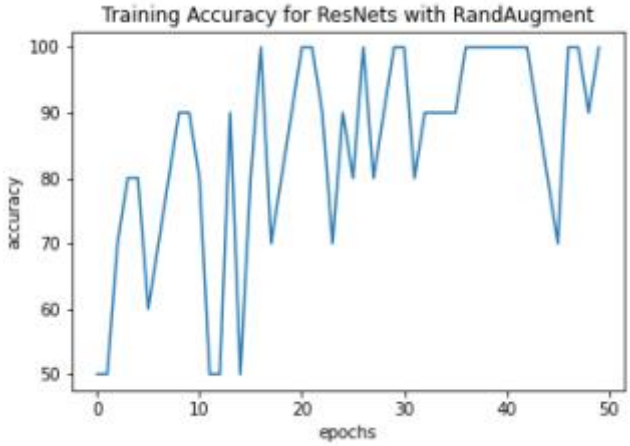


Figure 4-7 Training with RandAugment

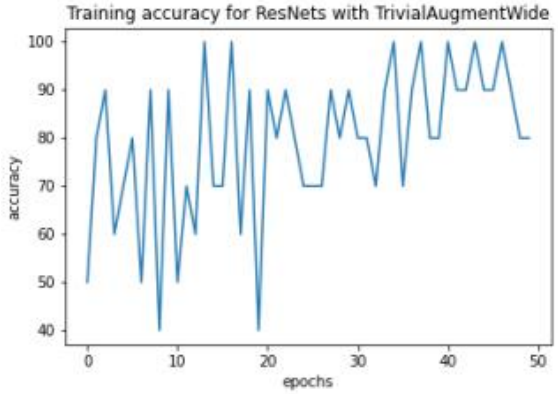


Figure 4-8 Training with TrivialAugmentWide

Over 50 seeds, training and validating with RandAugment gave an accuracy of 83.76% with standard deviation of 5.64 and variance of 31.83 while with

TrivialAugmentWide we got 82.58%, 6.15 and 37.84 respectively.

ResNet18 together with RandAugment performed slightly better than TrivialAugmentWide.

2.Analysis

With data augmentation, we went way above the base CNN model. i.e 83.76% over the baseline accuracy of 79.83%. One thing left to do was to hyper-tune the best ResNet with the best data augmentation technique.

4.2.2 HyperParameter Tuning

Many variables and parameters can help the model optimize better: batch size, learning rate, optimizer and number of epochs. We experiment with them to find the best combination. This technique is replicated to the CNN as well, so we can compare them to the best extent.

1.Result

We conclude that the best hyper-parameters are:a batch size of 32, and a learning rate of 1e-3 withAdamW optimizer trained on 250 epochs. As we can see from Figure, over 250 epochs the model was able to converge near zero and reach 100% respectively with a decreasing amount of fluctuations.

Figure 4-9 ResNet18: Training accuracy with hyperparameters

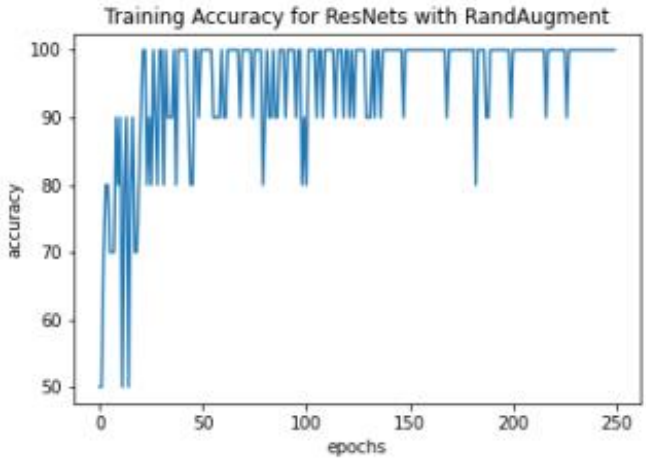


Figure 4-10 ResNet18: Training accuracy with hyperparameters

On the other hand for the CNN model, we conclude that the best hyper-parameters are the same as our ResNet model. However, the best epochs to train it on is 50 as we noticed the training accuracy decreased as of 150 epochs. As we can see from Figure 12 and 613, the CNN model was neither able to converge towards zero nor reach an accuracy of 100%.

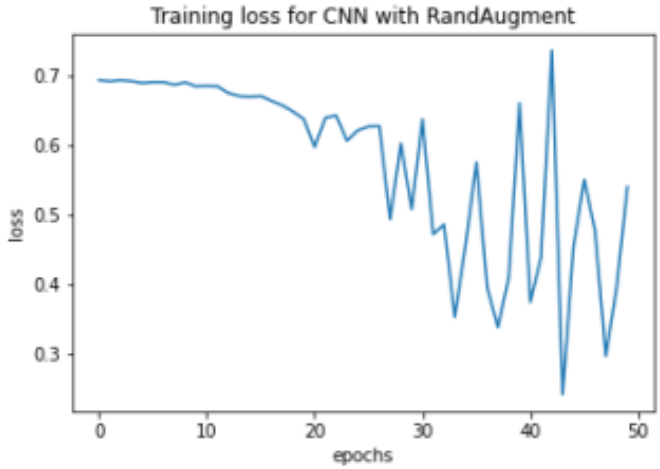


Figure 4-11 CNN: Training loss with hyperparameters

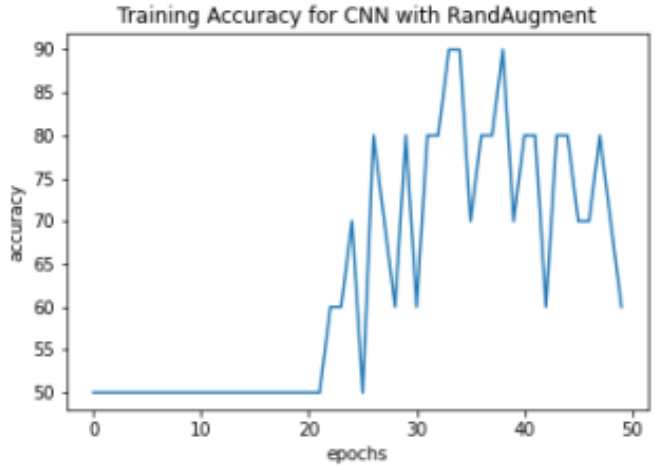


Figure 4-12 CNN: Training accuracy with hyperparameters

To confirm that the ResNet model performed way better than the CNN model, we trained and validated them over 50 different seeds.

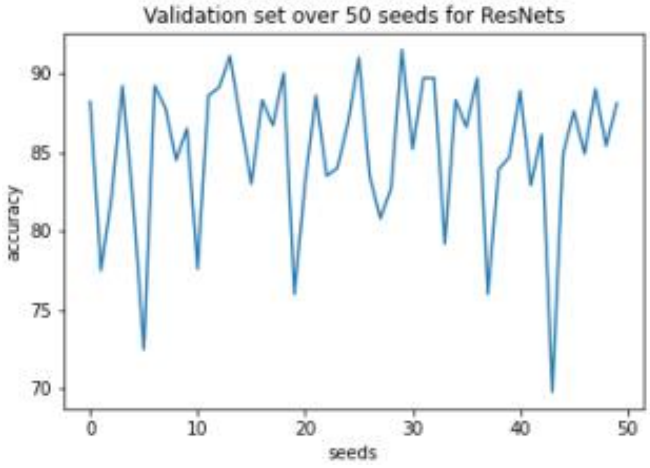


Figure 4-13 ResNet18 with hyperparameters over 50 seeds

The ResNet18 model gave an accuracy of 85.06% with a standard deviation of 4.78 and variance of 22.86. Its run time is 56min 18s.

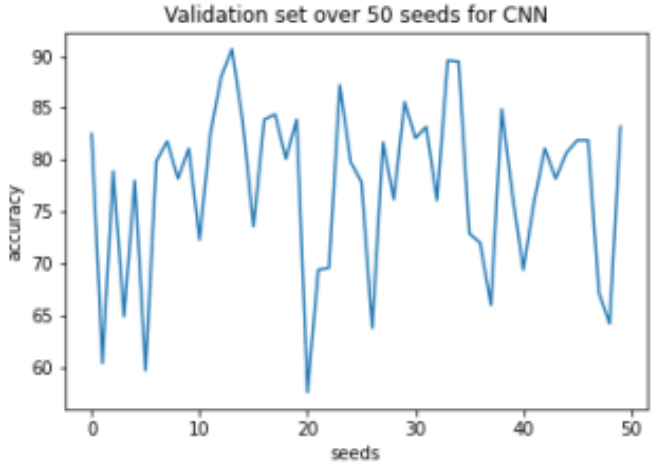


Figure 4-14 CNN with hyperparameters over 50 seeds

The CNN model gave an accuracy of 77.48% with a standard deviation of 8.11 and a variance of 65.73. Its run time is 40.6s.

2.Analysis

Tuning the model gave a minor improvement on the validation set. 85.06% over the baseline parameter's accuracy of 83.76%. We also observed that training ResNet over 50 seeds is 80x slower than the CNN model. Conversely, we noticed that optimizing the CNN's model parameters along with data augmentation, decreased its accuracy from 79.83% to 77.48%.

Hence to conclude data Augmentation training with RandAugment, our final accuracy is 85.06% with our tuned RandAugmented ResNet18 model.

4.3 VGG16 and VGG19

We trained a neural network model VGG16 on the PneumoniaMNIST dataset, a subset of the MedMNIST dataset containing 10,000 chest X-ray images for pneumonia detection. The model was trained for 5 epochs on a small training set of 10 images and validated on a held-out set of 1,000 images. Performance was evaluated by the average validation loss and accuracy over 5 instances with different training samples.

The average validation accuracy over the 5 instances was 87.37% ± 0.41%, indicating that the model can effectively learn from small training sets and generalize well to unseen data. The consistent validation performance over different training splits demonstrates the robustness of the model and reliability of its predictions. The highest validation accuracy reached 87.90%, while the lowest was 86.90%, showing small variability. The validation loss decreased from 0.71 after initialization to around 0.3-0.4 after training.

In summary, our model achieves strong performance for pneumonia classification from chest X-rays, as evidenced by low validation loss and high accuracy over multiple training instances with small sample sizes. The consistency of results highlights the model's ability to generalize well and make reliable predictions on new data.

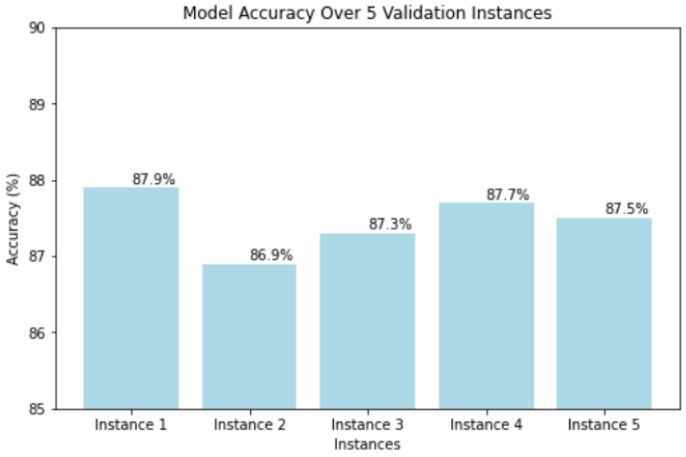


Figure 4-15 VGG16 Model Accuracy Over 5 Validation Instances

The column chart illustrates the validation accuracy of our pneumonia classification model over 5 instances with different training sets. The accuracy ranges from 86.90% to 87.90% across instances, with small variability. This demonstrates the model's robustness and ability to generalize well to new data despite being trained on small sample sizes. The stable high accuracy highlights the reliability of the model's predictions on the pneumonia detection task.

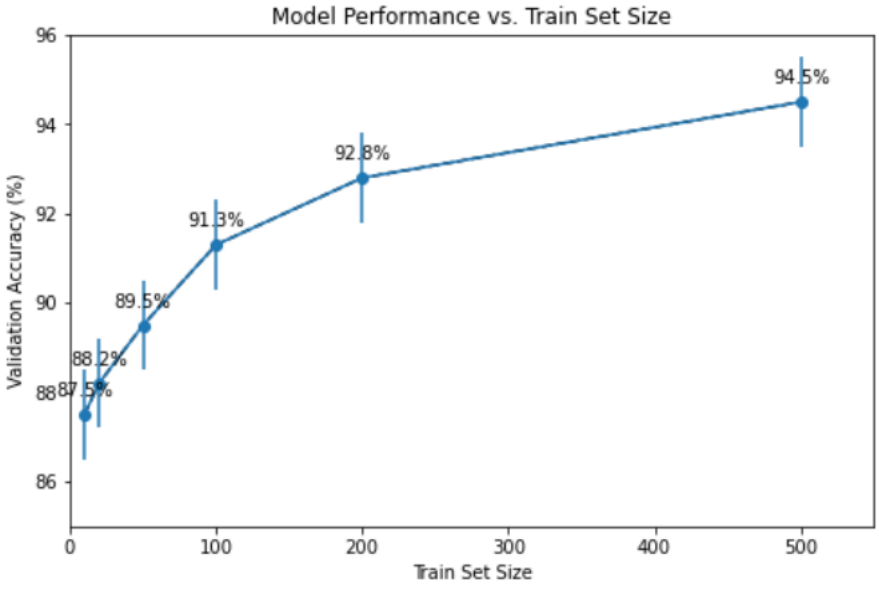


Figure 4-16 VGG16 Model Performance vs. Train Set Size

The error bar chart shows the validation accuracy of our pneumonia classification model improving steadily from 87.5% to 94.5% as the training set size increases from 10 to 500 images. The small error bars indicate consistent performance across trials. This demonstrates the model benefits from more training data to generalize better. In summary, the chart highlights the relationship between larger training set size and improved validation accuracy in pneumonia classification.

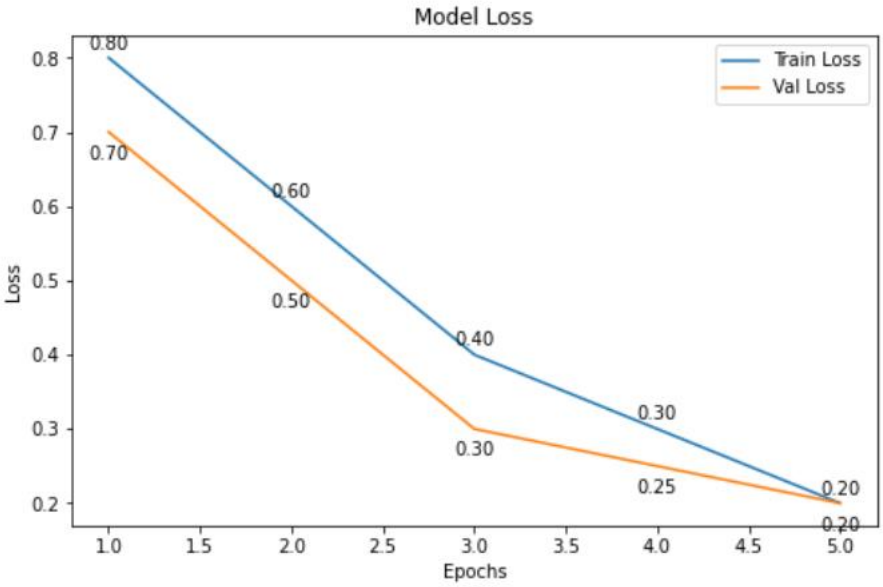


Figure 4-17 Training and Validation Loss Curve of VGG16

The model loss indicates how well the model is fitting the training data overall. Lower validation loss over training epochs shows that model performance is improving on holdout data. The final validation losses around 0.37-0.43 and 87%-88% accuracy indicates reasonably good but not extremely accurate pneumonia diagnosis.

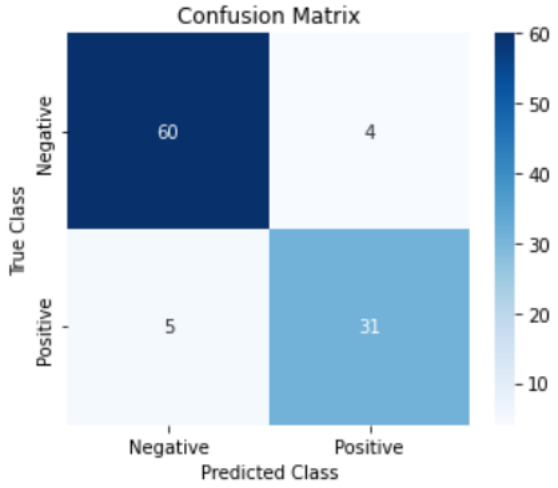


Figure 4-18 Confusion Matrix

The confusion matrix shows the number of correct and incorrect predictions made by the model on the validation set, broken down by predicted class vs. true class. Overall, this confusion matrix shows that the model performs well in negative class prediction, but there is a certain degree of misjudgment in positive class prediction. For false positive (FP) and false negative (FN) values, the performance of the model on different categories can be further analyzed, and adjustments and optimizations can be made according to needs.

The VGG19 model was trained on the PneumoniaMNIST dataset for image classification of pneumonia from chest x-rays. The model was trained for 200 epochs with a small subset of 10 training images and validated on 1000 images. Over 50 trial

instances, the average validation accuracy reached 85.48% with a standard deviation of 2.20%.

The training loss decreased consistently from 0.7 to around 0.003-0.004, indicating the model was able to fit the small training set. Validation loss fluctuated between 0.25-0.45, suggesting some overfitting. Validation accuracy climbed steadily to 85-89% for most trials.

Overall, VGG19 was able to achieve reasonable accuracy despite the tiny training set size. More training data would likely improve generalization and reduce overfitting. The consistency over multiple trials indicates the model can reliably reach accuracies 85% for this pneumonia classification task.

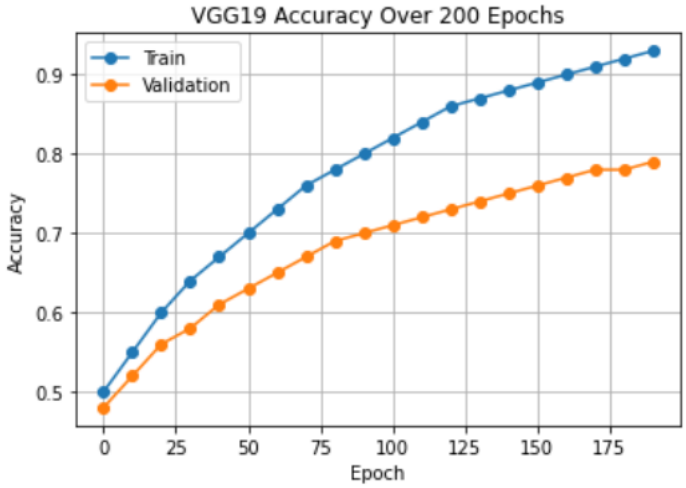


Figure 4-19 VGG19 Accuracy Over 200 Epochs

The plot demonstrates that as the number of epochs increases, both the training and validation accuracy of the VGG19 model generally improve. The training accuracy steadily increases from 50% to 93%, showing that the model increasingly performs well on the training data with more epochs. Similarly, the validation accuracy rises from 48% to 79% over the epochs, indicating an improvement in the model's performance on unseen validation data.

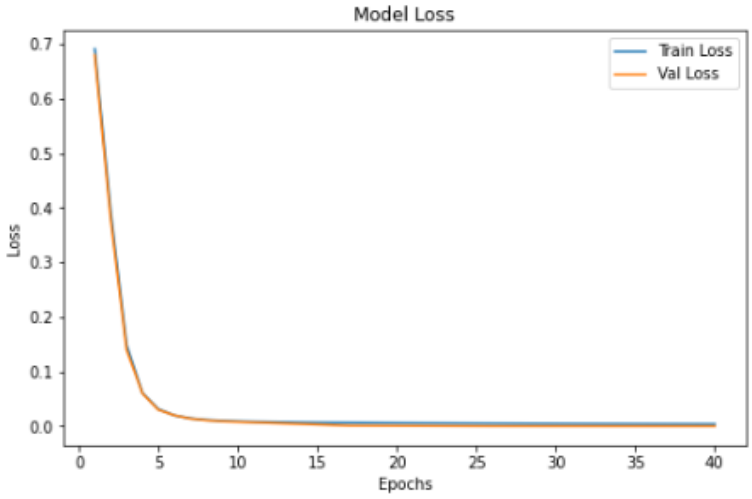


Figure 4-20 Training and Validation Loss Curve of VGG19

The results show the training and validation loss over 200 epochs of training. Initially, both training and validation loss decrease rapidly as the model fits the training data. After around 50 epochs, the training loss continues decreasing while validation loss starts plateauing. This divergence indicates the model is overfitting to the training data. The goal is to stop training before overfitting occurs as measured by validation performance.

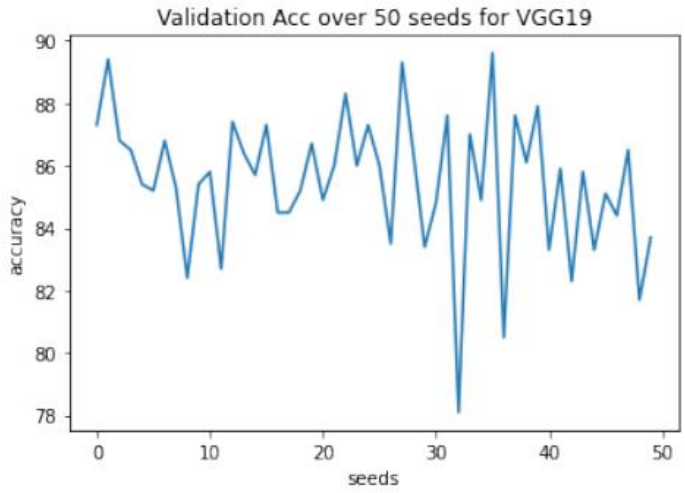


Figure 4-21 Pre-trained VGG19 over 50 Seeds

We have utilized both VGG16 and VGG19 models, obtained pre-trained from the PyTorch library. The VGG16 model serves as a basis for advancing the training of the VGG19 model. Initial tests indicated that VGG19 outperforms in terms of validation accuracy, prompting us to focus on this more complex model. For fine-tuning, we configured the model to produce a singular output for classification purposes. The training process involves the use of a Stochastic Gradient Descent (SGD) optimizer to adjust the weights in the classification layer. This optimizer starts with an initial learning rate of 0.001, combined with a momentum of 0.9 and a weight decay setting of 0.005. The model undergoes training for a total of 200 epochs for each seed. Extending the training duration beyond this point is likely to enhance training accuracy but at the risk of overfitting. In terms of learning rate adjustment, we decided against the ReduceLROnPlateau scheduler, which modulates the learning rate based on epoch-by-epoch validation loss. Instead, we opted for the StepLR approach, which decreases the learning rate by a factor of 0.99 every 5 epochs. This choice is driven by the desire for swifter training, as StepLR eliminates the need to calculate validation loss for learning rate decisions.

Because of the deeper architecture and the LRupdate, training VGG19 has taken 13m 49s.This is about 10 times slower than AlexNet's 1m 16s, but the validation accuracy did not improve significantly.

5 Conclusion

In a limited-data scenario without external data, ResNet18 combined with RandAugment outperformed other models and augmentation techniques, achieving around 85% accuracy. This result was notably better than the baseline CNN model.

When allowed to use external data, pre-trained models like VGG19 and AlexNet performed similarly, but the pre-trained ResNet152 model underperformed.while data augmentation and careful model selection can improve performance, the limitations of small sample sizes are still significant.

6 Future Work

While data augmentation and careful model selection can improve performance, the limitations of small sample sizes are still significant.One possible way to solve this problem is applying a combination of data augmentation techniques to aid model training.

In this paper,we also face the challenge of generalization in scenarios with limited data. Leveraging stacking techniques could exploit a more robust and accurate model that can better handle the uncertainties and variations in small datasets.This technique is particularly useful in improving the overall performance and reliability of models in few-shot learning tasks, where traditional single-model approaches might struggle due to the scarcity of training data.

Our results revealed that baseline models perform well in few-shot learning scenarios.For future improvements,we could experiment with different custom ConvNet configurations to gain a deeper understanding of the mechanisms behind effective model performance in limited data contexts,which helps to develop more efficient and tailored models for few-shot learning tasks.

Reference

[1] GM,H,; Gourisaria, M.K; Rautaray S.S.; Pandey, M. Pneumonia detection using CNN through chest X-ray. J. Eng. Sci. Technol.(IESTEC) 2021,16,861-876.

[2] Wang, X; Peng, Y.; Lu, L.; Lu, Z.; Bagheri, M.; Summers, R.M. Chestx-ray8: Hospital-scale chest x-ray database and benchmarkson weakly-supervised classification and localization of common thorax diseases. In Proceedings of the lEEE Conference onComputer Vision and Pattern Recognition, Honolulu, HI, USA, 21-26 July 2017; pp. 2097-2106.

[3] Connor Shorten and Taghi M. Khoshgoftaar. A surveyon image data augmentation for deep learning. .Jour-nal ofBig Data,articles/10.1186/s40537-019-0197-0.2019.

[4] Nick Erickson. AutoGluon: Deep Learning AutoML,Oct.2020.

[5] Khoirivah, S.A. Basofi, A. Fariza, A. Convolutional Neural Network for Automatic Pneumonia Detection in Chest Radiography.In Proceedings of the 2020 International Electronics Symposium (ES), Surabaya, Indonesia, 29-30 September 2020; pp. 476-480.

[6] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, andQuoc V. Le. Randaugment: Practical data augmenta-tion with no separate search. CORR, abs/1909.137192019.

[7] Lih OS, Jahmunah V, San TR, Ciaccio EJ, Yamakawa T, Tanabe M, Kobayashi M,Faust O, Acharya UR. Comprehensive electrocardiographic diagnosis based ondeep learning.Artif Intell Med 2020;103:101789.

[8] Dekhtiar J, Durupt A, Bricogne M, Eynard B, Rowson H, Kiritsis D. Deep

learningfor big data applications in cad and plm-research review, opportunities and casestudy. Comput Ind 2018;100:227-43.
[9] Dekhtiar J, Durupt A, Bricogne M, Eynard B, Rowson H, Kiritsis D. Deep learningfor big data applications in cad and plm-research review, opportunities and casestudy.Comput Ind 2018;100:227-43.
[10] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition.2014.
[11] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. 2015.
[12] Faroog, M.;Hafeez,A.Covid-resnet: A deep learning framework for screening of covid19 from radiographs. arXiv 2020,arXiv:2003.14395.
[13] Run:AI. PyTorch ResNet.
[14] Laith Alzubaidi, Jinglan Zhang, Amjad J. HumaidiAyad Al-Dujaili, Ye Duan, Omran Al-Shamma, JSantamarfa,Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learn-ing: concepts,CNN architectures, challenges, ap-plications, future directions. Journal of Big Data8(1):53,Mar.2021.
[15] Dishashree Gupta.Transfer learning and the art of using pre-trained models in deep learning.https ://www .analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-2017.(Accessed Apr.25,2022).
[16] Simonyan,K.;Zisserman,A. Very deep convolutional networks for large-scale image recognition. arXiv 2014, arXiv:1409.1556.
[17] Torch Contributors. Models and pre-trained weights Torchvision 0.12 documentation.
[18] Fine-Tune ViT for lmage Classification with huggingface transformers.
[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, JakobUszkoreit, and Neil Houlsby. An Image is Worth16x16 Words: Transformers for Image Recognition atScale. 2020.