Convolutional Neural Network for Pneumonia detection using chest X-Ray images

Shubham Narandekar
Computer Science
University College Dublin
Student id: 20200132
shubham.narandekar@ucdconnect.ie

Abstract

The task of detecting pneumonia using the chest X-Ray images is described in this paper. The dataset used for this task was obtained from the Kaggle platform which was named as 'Chest X-Ray images. The goal of this task is to build a neural network model or fine tune a pre-trained model to classify the chest x-ray images. Detailed analysis of the data is performed and required pre-processing steps are taken. A baseline approach is used to build the proposed model and after analysing the results, the proposed model is improved using necessary techniques. Fine tuning of VGG16 pre-trained model using transfer learning is done. Finally, both the proposed model and pre-trained model are evaluated and the results are compared.

1 Introduction

This is an image classification task where the images are classified into one of the two classes that are 'Pneumonia' and 'Normal'.

https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

This project is about creating and building a neural network model for classifying the chest x-ray images in one of the two categories. The great success achieved by deep learning models in analyzing medical images have assisted Convolutional Neural Networks to gain much more attention in disease classification field. The pre-trained models like VGG16 have been trained on large datasets making the features learned by these models very much useful in image classification tasks. Deep learning allows models to learn the data that is passed through multiple layers of the neural network architecture and improve along with learning so that it could classify well on unseen data. Applications like image classification, object detection, etc comes under the category of computer vision which is one of the fields of deep learning. According to previous studies Convolutional Neural Networks have been proven to perform best for computer vision tasks.
I have build a basic convolutional neural network model and have compared it with basic VGG16 [2] model without fine tunning it. The results have been analysed to check how these models perform on the dataset which is initially not pre-processed. After analysing the results, required pre-processing steps are taken to improve the performance on my proposed model. By applying transfer learning I have also fine tuned the VGG16 model to perform best for the given dataset.

2 Related Work

My research to find how to build a convolutional neural network and to apply pre-processing on the given dataset have led me to many previous related works. There are many previous studies that are

done for image classification using convolutional neural networks. A recent similar study[1] was done for predicting Covid19 with chest X-Ray images using CNN in which they encountered issues with smaller size of training data. This issue was resolved by using ImageGenerator from keras and ten thousand augmented images were generated and also the performance of their proposed model was then compared with VGG16 and AlexNet. It was seen that their proposed model performed better than AlexNet by giving higher AUC score. The dataset that I am using has similar issues of less training data.
In a previous study[3] for pneumonia detection using CNN based feature extraction they have proven that DenseNet169 is the optimal model for feature extraction from the images. This model was then supported by Support Vector Machine for further classification of the images using the extracted features. To reduce the heavy computational cost, the images were resized from (1024*1024) into (224*224) pixels for faster processing.
Transfer learning of the pre-tained models like AlexNet and ResNet18 were carried out in[4] for detecting pneumonia using chest x-ray images. Mainly three data augmentation techniques like Rotation, Scaling and Translation were applied in the pre-processing of the data. These augmentation techniques actually helped them in increasing the classification accuracy. After comparison of the results it was found that ResNet18 gave an AUC score of 0.96 for binary classification. As their data was imbalanced, they have used ROC AUC as their evaluation metrics for comparing the pre-trained models. However, the work done in most of the literature mainly focuses on detecting the features by using CNN architectures.

3 Experimental setup

Points to be taken under consideration before starting:

- To check if GPU (cuda) is available on your machine because model training is faster on GPU than CPU.
- To check which neural network architecture is well suited for computer vision applications. As discussed earlier, Convolutional Neural Networks perform well for the computer vision tasks.
- To check the shape of an image to know the number of channels which is to be specified as number of input channels to the first convolutional layer while building your model.
- To check if the data is balanced. In the case of the proposed implementation, the dataset obtained is imbalanced. There are more number of 'Pneumonia' images than 'Normal.'
- If the data is imbalanced, which steps should be taken to handle such data? Is there a need of applying data augmentation techniques?
- To check how the entire data is divided into training, validation and testing set. In the case of the current implementation, the validation contains only 16 images.
- If using GPU, choose appropriate batch size while loading the data because GPU will result in memory issues if it could not handle the given batch size.

3.1 Dataset

The dataset used in this project is obtained from kaggle platform. There are two categories in which the given images should be classified that are 'Pneumonia' and 'Normal' which makes it a binary classification task. After plotting the graph for class count, it was seen that the dataset is highly imbalanced with 3875 images belonging to the 'Pneumonia' class and 1341 belonging to the 'Normal' class. Using such data will lead to poor prediction for minority class. There are two approaches that can be used to handle this problem which are oversampling by using Weighted Random Sampler or simply assigning weights to the classes within the loss function. The splits for training and validation set which is provided has 5216 images for train and only 16 images for validation. Due to the small size of the validation set, it won't be able to evaluate model's performance in a robust manner. This issue is resolved and the validation set is increased in the data pre-processing part.

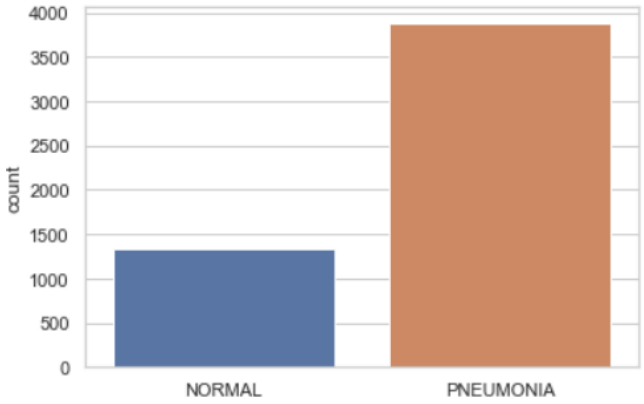

Figure 1: Imbalanced data

3.2 Data pre-processing

Initially, basic transformer is used to resize the images to have (224,224) pixels and then convert the images to tensor in the data augmentation part. Once the transformation is done. The number of samples in the training and validation set is checked before loading the data using the DataLoader. The number of samples provided in validation set are just 16 so the probability distribution of the data is not adequately represented, which will not allow to evaluate the model's performance in a robust manner as discussed earlier. To re-sample the training and validation set a function is defined that combines the training and validation samples and return the combined set. Using "traintestsplit", the combined set is then divided again into training and validation set with validation size of 0.1. Now the validation set have 524 images and the train set have 4708 images. While loading the images using the DataLoader a batch size of 8 is used for training loader and a batch size of 4 for validation loader. A bath size of 64 and 128 were also tried but every time the GPU faced memory issues as these batch sizes were costly and could have allowed the model to train faster. After following the trial and error approach, 8 is decided as an optimal value for the batch size as per the GPU's capacity. After trying different values for batch size, it was leanred that smaller batch size offers a regularization effect and also helps to lower the generalization error. A study conducted in [9] concluded that lowering the learning rate and having lower batch side will allow the model to train better.

3.3 Baseline Approach

Following is the architecture for the baseline approach of the proposed model:

cnn(
(conv1): Conv2d(3, 32, kernelsize=(3, 3), stride=(1, 1), padding=(1, 1))
(relu1): ReLU()
(pool1): MaxPool2d(kernelsize=2, stride=2, padding=0, dilation=1, ceilmode=False)
(conv2): Conv2d(32, 64, kernelsize=(3, 3), stride=(1, 1), padding=(1, 1))
(relu2): ReLU()
(pool2): MaxPool2d(kernelsize=2, stride=2, padding=0, dilation=1, ceilmode=False)
(conv3): Conv2d(64, 128, kernelsize=(3, 3), stride=(1, 1), padding=(1, 1))
(relu3): ReLU()
(fc): Linear(infeatures=401408, outfeatures=2, bias=True)
)

Initially the image passed to the first convolutional layer is of size (224*224) with three channels. As there are two Max Pooling layers applied, the flattened vector passed to the fully connected linear layer is of (56*56*128) input features. The non linear activation function is not specified after the last linear layer because while training CrossEntropyLoss is used as a loss function provided by pytorch which internally combines LogSoftmax and NLLLoss in a single class so automatically the output of the final fully connected layer will be given to the LogSoftmax activation function. The reason for using LogSoftmax is it distributes the probabilities throughout each output

class and it is just an generalization for Sigmoid function. Another way is that Sigmoid function can be used after the final fully connected layer and 'BCELoss' or 'BCEWithLogitsLoss' can be used be used as a loss function.
Along with the above model, transfer learning was applied on a pre-trained VGG16 model [2] in which only a single fully connected layers was applied. Once these models were trained using 10 epochs, their performance on unseen test data was evaluated. As the data is imbalanced, 'Receiver Operating Characteristic (ROC) Area Under Curve (AUC)' is the best choice as an evaluation metrics to evaluate the performances of these models. After evaluating these models following AUC scores were achieved:

- Proposed Model (Basic)
  Training AUC score: 0.96 Testing AUC score: 0.53

- VGG16 Model (Basic)
  Training AUC score: 0.94 Testing AUC score: 0.75

3.4 Improved Approach

Following are the techniques that were used to resolve the overfitting problem of both the models:

1. Data Augmentaion: Due to the limitation of data in the training set the model may tend to overfit. Various data augmentation techniques have been applied like flipping the images horizontally, rotating them with an angle of 30 which creates additional augmented data from the existing training data which is also experimented in [1]. The images are also normalized.

2. Regularization: Dropout which is a regularization technique have been applied before both of the fully connected layers of my proposed model. The reason for using dropout is it helps to reduce overfitting by randomly dropping neurons from the neural networks during each epoch. Since each time we are dropping neurons, the model does not favour particular neurons every time, thus reducing overfitting.

The above models were trained using the imbalanced dataset but, later on, the issue of class imbalance was dealt by using class weights in the loss function. By doing this we can give higher priority to the minority class while computing the loss for that class. The weights assigned to 'Normal' which is the minority class is 0.80 and to the 'Pneumonia' class is 0.20.

```
cnn2(
(conv1): Conv2d(3, 32, kernelsize=(3, 3), stride=(1, 1), padding=(1, 1))
(relu1): ReLU()
(pool0): MaxPool2d(kernelsize=2, stride=2, padding=0, dilation=1, ceilmode=False)
(conv2): Conv2d(32, 64, kernelsize=(3, 3), stride=(1, 1), padding=(1, 1))
(relu2): ReLU()
(pool1): MaxPool2d(kernelsize=2, stride=2, padding=0, dilation=1, ceilmode=False)
(conv3): Conv2d(64, 128, kernelsize=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(batchnorm1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, trackrunningstats=True)
(relu3): ReLU()
(pool2): MaxPool2d(kernelsize=2, stride=2, padding=0, dilation=1, ceilmode=False)
(conv4): Conv2d(128, 256, kernelsize=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
(batchnorm2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, trackrunningstats=True)
(relu4): ReLU()
(pool3): MaxPool2d(kernelsize=2, stride=2, padding=0, dilation=1, ceilmode=False)
(dropout1): Dropout(p=0.5, inplace=False)
(fc1): Linear(infeatures=50176, outfeatures=256, bias=True)
(relu5): ReLU()
(dropout2): Dropout(p=0.2, inplace=False)
(fc2): Linear(infeatures=256, outfeatures=2, bias=True)
)
```

The above architecture depicts the proposed model after the improvements were done. As there are four Max Pooling layer applied in the improved model, the flattened vector passed to the first fully connected layer is of (14*14*256) input features. A dropout of 0.5 is applied before the first fully connected layer and of 0.2 before the second fully connected layer to add a regularization effect in the model. Along with the proposed model, fine tuning on the basic VGG16 model is performed by adding dropout and batch normalization as the batch normalization was not introduced when the VGG16 was originally developed. Adding batch normalization helped in stabilizing the learning process. Once the improvements were done and the overfitting problem was resolved both the models were evaluated on unseen test data and following AUC scores were achieved:

- Proposed Model (Improved)
  Training AUC score: 0.94 Testing AUC score: 0.83
- VGG16 Model (Improved)
  Training AUC score: 0.93 Testing AUC score: 0.87

3.4.1 Hyperparameter Tuning and optimization function

Trial and error approach was followed to try out couple of hyperparameters listed below.

- Batch-size: [**8**, 64 ,128]
  Among the above specified batch sizes, 8 worked best for me because smaller batch size lowered the generalization error and also showed faster convergence into good solutions.
- Learning rate: [0.1, **0.001**, 0.0001]
  After trying the above learning rates, 0.001 worked best for me.
- Optimizer used: **Adam optimizer**
  Adam optimizer is a combination of Adaptive Gradient Algorithm and Root Mean Square Propogation and unlike Stochastic Gradient Descent the learning rate of each network is maintained and adapted as the learning goes on. Also Adaptive Gradient Algorithm has been proven to improve performance on computer vision problems. Another reason for considering this optimizer is it requires little memory as compared to other optimizers and it is also computationally efficient.

If high computation power is available then we can even apply GridSearchCV and Randomized-SearchCV to try multiple hyperparatmeters on the model and get the best parameters out of those which are well suited for the model and gives best performance.

4 Results

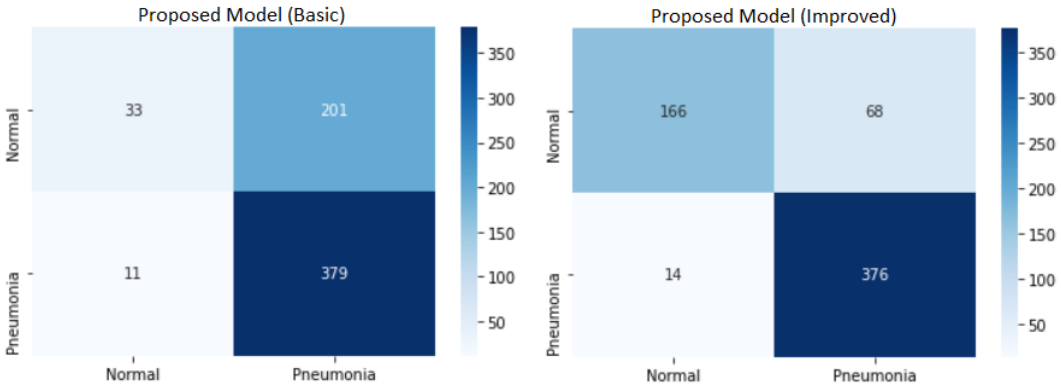

Figure 2: Proposed Model (Before and After)

The above figure depicts the confusion matrix of the proposed model after evaluating on unseen test data. We can see on the left side that before improvement the model was overfitting and was actually

not able correctly classify images from minority class and it was biased towards the 'Pneumonia' class. But once the class imbalance was resolved by using class weights in the loss function and more training data was generated using data augmentation the model performed better than before in predicting the minority class.

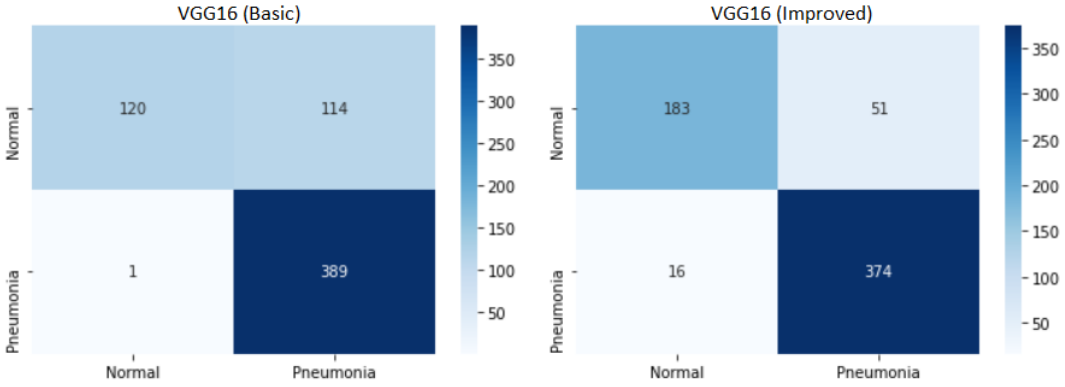

Figure 3: VGG16 Model (Before and After)

Earlier the VGG16 model was not predicting well on the unseen test data as there was only one fully connected layer and no regularization techniques were used. After applying dropout, adding another fully connected layer and giving class weights in the loss function, the model performed very well than all other models.

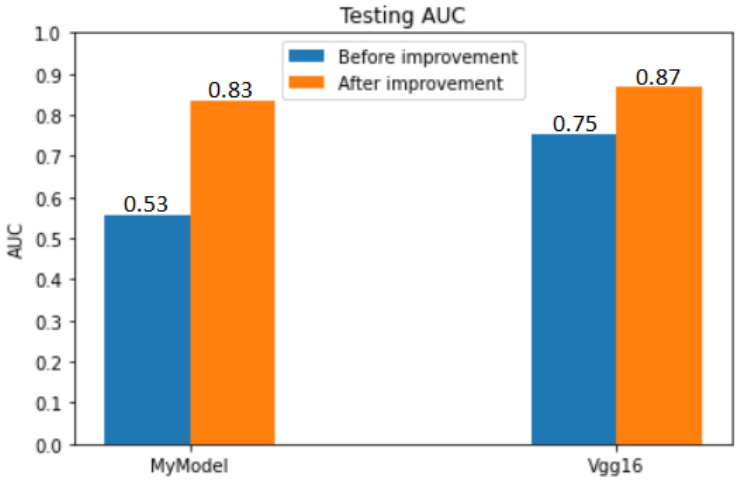

Figure 4: Testing ROC AUC

From the above figure we can see that both the models were overfitting before but after proper data augmentation and regularization techniques I was able to achieve higher AUC score for both the models. Also specifying class weights in the loss function played an important role as the data was majorly imbalanced. Following is a table with multiple evaluation metrics for all the models:

	Proposed Model (Basic)	VGG16 (Basic)	Proposed Model (Improved)	VGG16 (Improved)
ROC AUC	0.53	0.75	0.83	**0.87**
PR AUC	0.82	0.88	0.91	**0.93**
Test Accuracy	66.02 %	81.57 %	86.85 %	**89.26 %**
Sensitivity	97.17 %	**99.74 %**	96.41 %	95.89 %
Specificity	14.10 %	51.28 %	70.94 %	**78.20 %**

Figure 5: Evaluation metrics

AUC: Area Under Curve.
ROC: Receiver Operating Characteristic curve.
PR: Precision-Recall curve.
Sensitivity: It is the ability of the model to correctly identify patients with Pneumonia.
Specificity: It is the ability of the model to correctly identify patients that are Normal.

5 Future Work

For future work, more data can collected and added in the training data to deal with the overfitting problems of the model. The number of minority class images should be collected and added in the training data set to make the data set balanced. If high computation power is available then instead of resizing or cropping the images we can use original high definition images. All the images that are incorrectly classified can be used and trained again to improve the predictions. As proposed in[6] we can use ensemble techniques on neural networks which will give us a subset of most error-independent neural networks that are more effective. Previous studies[7] have proven that Super Learner have performed best among all other ensemble methods. By using more computation power, complex neural networks can be used and trained for image classification. Image data enhancement methods like InnerMove[8] can be used to enhance the existing training data which will help to get more discriminating features from the images. GridSearchCV or RandomizedSearchCV can be used to try multiple combinations of hyperparamters if GPU of higher specification is available for faster training.

6 Conclusion

This study presents the use and creation of convolutional neural networks to classify between patients that are diagnosed with 'Pneumonia' and the patients that are 'Normal' using the chest x-ray images. The proposed model have performed very well on the unseen data as it holds very high sensitivity score which is important in health care domain. This states that the model was 96.41 percent accurate in predicting patients that are diagnosed with 'Pneumonia'. As proven, regularization techniques like dropout has improved the model and have resolved the overfitting issues. Batch normalization techniques was not introduced earlier when VGG16 model was developed. By using transfer learning I have added batch normalization in the pre-trained VGG16 model which resulted in very good performance among all the other models that I have used. It gave very good sensitivity and specificity score on the unseen test data. With the help of ensemble methods these models can be improved more in future.

References

[1] Umer, M. et al. (2021) "COVINet: a convolutional neural network approach for predicting COVID-19 from chest X-ray images", *Journal of Ambient Intelligence and Humanized Computing*.

[2] Simonyan, K. and Zisserman, A. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv.org. Available at: https://arxiv.org/abs/1409.1556 (Accessed: 11 April 2021).

[3] D. Varshni, K. Thakral, L. Agarwal, R. Nijhawan and A. Mittal, "Pneumonia Detection Using CNN based Feature Extraction," 2019 *IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, India, 2019, pp. 1-7

[4] Rahman, Tawsifur Chowdhury, Muhammad Khandakar, Amith Islam, Khandaker and Islam, Khandaker Mahbub, Zaid Kadir, Muhammad Kashem, Saad. (2020). Transfer Learning with Deep Convolutional Neural Network (CNN) for Pneumonia Detection using Chest X-ray.

[5] Ibrahim, A.U., Ozsoz, M., Serte, S. et al. Pneumonia Classification Using Deep Learning from Chest X-ray Images During COVID-19. Cogn Comput (2021). https://doi.org/10.1007/s12559-020-09787-5

[6] Giacinto, G. and Roli, F. (2001) "Design of effective neural network ensembles for image classification purposes", Image and Vision Computing, 19(9-10), pp. 699-707.

[7] Cheng Ju, Aurélien Bibaut and Mark van der Laan (2018) The relative performance of ensemble methods with deep convolutional neural networks for image classification, Journal of Applied Statistics, 45:15, 2800-2818.

[8] Tang, C. et al. (2020) "PLANET: Improved Convolutional Neural Networks with Image Enhancement for Image Classification", Mathematical Problems in Engineering, 2020, pp. 1-10.

[9] Kandel, I. and Castelli, M. (2020) "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset", ICT Express, 6(4), pp. 312-315.

[10] `https://www.kaggle.com/parthdhameliya77/pneumonia-classfication-tutorial-pytorch`

[11] `https://www.youtube.com/watch?v=9OHlgDjaE2I&list=PL1oOe7xElp02NM7vMylmLuxGmb4-8cIfQ&index=2t=982s`

[12] `https://www.youtube.com/watch?v=p3CcfIjycBA&list=PL1oOe7xElp02NM7vMylmLuxGmb4-8cIfQ&index=6`

[13] `https://www.youtube.com/watch?v=4JFVhJyTZ44&list=PL1oOe7xElp02NM7vMylmLuxGmb4-8cIfQ&index=5`