Using Machine Learning Models to Classify Pneumonia from X-Ray Images

Oscar O'Rahilly

1. Introduction

Pneumonia is the fourth leading cause of death globally, and in 2016 alone nearly 3 million people lost their lives due to pneumonia complications. Additionally, COVID-19 induced pneumonia has been one of the main reasons the virus is so deadly. When we combine this with the fact that a country such as Kenya, with over 43 million citizens, has only 200 radiologists to serve the needs of the entire population, it becomes clear that something needs to be done to combat the inequality in access to healthcare professionals we see globally.

As a result, in this paper we explore different Machine Learning implementations that can help classify pneumonia from X-ray scan images. We will be looking at a Naive Bayes classifier, SVM classifier, a Convolutional Neural Net (CNN) classifier and finally a Transfer Learning based classifier. Our input is in the form of X-ray scan images, and we output a classification of the presence of pneumonia.

2. Related Works

2.1. Image Feature Extraction:

In his 2016 paper titled "Image Classification Using Naive Bayes Classifier" Park Discusses various approaches for image feature extraction, which can then be fed as inputs to Naive Bayes for an image classification task. In his paper, Park compares the discrete cosine transform (DCT), local binary pattern (LBP), covariance descriptor, and wavelet transform as feature extraction methods for a 4 class classification task. Park identifies DCT as the best feature extraction method since it even beats many of the nueral network classification models he tried. Thus we decided to similarly use the discrete cosine transform to extract features for our Naive Bayes classifier

2.2. Data Augmentation:

Perez and Wang (2017) discusses the usefulness of performing data augmentation on image data sets. One use for data augmentation is in the medical field where finding large data sets can be challenging, particularly for use in diagnosis and classification. As a result, we have decided to perform geometric augmentation on our images (transformations that involve mapping pixels to other locations) as they are known for their simplicity in design, ease of use, and quick runtime.

2.3. Convolutional Neural Networks

The Krizhevsky et al. (2012) architecture won the ImageNet competition in 2012. While it is large-scale and computationally expensive, we took certain features of its architecture in designing our CNN. Their use of max-pooling layers was useful in reducing overfitting in their model, and as a result we implemented them after every activation function in our own model.

2.4. Optimization:

In class, we discussed the basic stochastic gradient ascent rule, $\theta := \theta + \alpha \nabla_\theta \ell(\theta)$. Other solutions to the loss minimization problem, such as RMSprop and Adam (Kingma and Ba (2014)) offer faster and more efficient calculations for first-order gradient descent, based on approximation of lower order moments. We experimented with both in our models.

2.5. Transfer Learning

Razavian et al (2014) provide a strong argument for the use of Transfer Learning in our image classification problem. By using an existing trained high-performance model, and using it on a separate classification problem, we can still observe very strong performance. This is a result of generic descriptor on a pre-trained CNN. Whereas Razavian et al decided to use *OverFeat* for image object classification, we have decided to use a similar approach with the CNN model *DenseNet161* for its balance of high performance and reduction of memory usage. *DenseNet161* uses half the number of parameters and FLOPs as competing state of the art models like *ResNet*.

3. Data set

Using X-ray scan datasets from Kaggle, we split data into a training, validation and testing set (roughly 70%, 15%, 15% split, respectively). We had **5856** images, resulting in a respective **4099**, **878** and **879** split.

Data Augmentation: In order to prevent the issue with overfitting, we experimented with data augmentation. Specifically, we transformed images in the data set with random inverting, rotation, translating and grayscaling. We experimented with different resizing techniques depending on the model, which will be discussed in the results section.

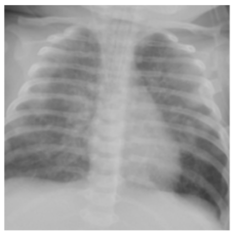


Figure 1. Standard Image from Data Set

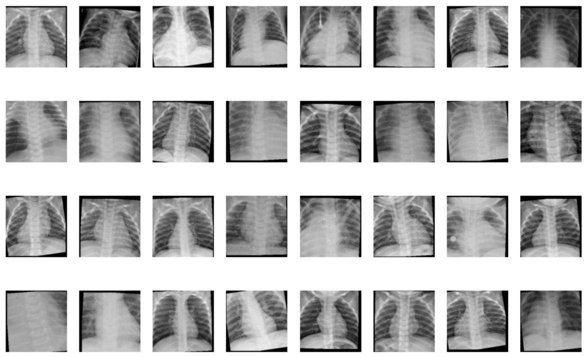


Figure 2. Transformed Images used in CNN and Transfer Learning

Our data set was more heavily weighted towards X-Rays with positive classifications for pneumonia. Specifically, there were **73%** pneumonia labeled images, and **27%** normal labeled images. As a result, simple accuracy metrics are not descriptive enough in discussing the effectiveness of our models. In our results section we will elaborate further on the metrics we used to assess our model accuracy.

4. Methods

4.1. Naive Bayes

The Model: While Naive Bayes is one of the simplest algorithms for machine learning, we thought it would be a good place to start when building our pneumonia detection model since it would allow us to learn more about the dataset and give us a baseline accuracy that we could improve on with more complex models. We tried several feature extraction techniques as described below to generate features to feed to our model during training and testing.

How it works: The Naive Bayes model works by simplifying a classification task using the Naive Bayes assumption. The assumption is that the probability of each feature occurring is conditionally independent of all other features given the class. While this is clearly not correct, it allows for a massive simplification that actually works well in practice. The simplification allows a computation for the probability of some input X belong to come class C to be $P(class = C|features = X) = \prod_i P(class = C|features_i = X_i)$. This expression can be computed much more easily by counting the number of times feature i occurs in class C vs the total number of times feature i occurs at all.

Image pre-processing: In order to prepare our dataset for training on the Naive Bayes classification model, we first had to process our images and give them a standard format. To do this we converted all the images to grayscale, and then resized them to be 100 x 100 pixels to keep the dimensions constant across images.

Feature extraction: Since the Naive Bayes model is relatively simple, it is not a powerful enough model to classify images based solely on pixel data as is the case with more complex deep learning models. Thus a key step in using Naive Bayes for image classification is choosing an effective feature extraction technique to derive trainable features from the pixel data of the images. Two techniques that we tried for this task are discrete cosine transform (DCT) and local binary pattern (LBP). DCT is commonly used in image compression where images are represented as a weighted sum of cosine waves, and the original image can then be recovered by calculating this sum. DCT is performed on an image by looking at a sliding square of pixels (usually 8 x 8), and performing the local DCT calculation. Thus the 100 x 100 pixels has a grid of 100 x 100 DCT values after the calculation. We experimented with windows of size 4, 8, and 16.

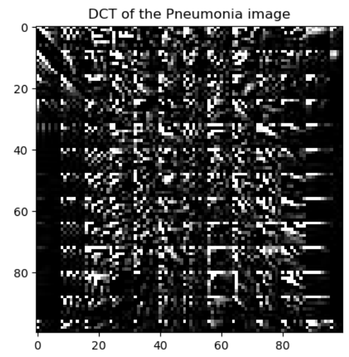


Figure 3. Discrete Cosine Transform example

4.2. Support Vector Machines

The Model: In addition to Naive Bayes, the SVM model is commonly used as an early benchmark given its relative simplicity, and success in a wide variety of tasks. We experimented with three different kernels: linear, degree 3 polynomial, and radial basis function (rbf).

How it works: SVM models for classification tasks generally work by trying to find a line or hyperplane that separates the data points of both classes. While the data does not need to be perfectly separable, SVMs define the best separating line as the one with the widest margin to the nearest data points of each class. SVMs can also be used in non-linear cases by defining a kernel function that creates new relationships between features to better separate the data. Similarly to the Naive Bayes model, we used DCT as a feature extraction technique to feed to our model.

4.3. Convolutional Neural Network (CNN)

A Convolutional Neural Net is the gold standard neural network for computer vision. Rather than passing the input through linear layers followed by non-linearities, a convolutional neural net uses filters to convolve the training image before passing the results through non-linearities. The advantage of this is that we can preserve the spatial structure of the inputs as we are not flattening them like we do in a purely affine case.

The Model: We experimented with a variety of different architectures for our CNN. Our first iteration of the CNN was as follows,

(5x5) Convolution $\rightarrow$ ReLU $\rightarrow$ (3x3) Convolution $\rightarrow$ ReLU $\rightarrow$ Affine $\rightarrow$ Softmax

Whilst this provided respectable results (total accuracy of 85%) we noticed that once we applied transformations to our training set, the accuracy began to dip. In order to rectify this, we altered our CNN to add a Maxpool layer after every activation function in our CNN. Maxpool layers make the model more robust when it comes to translational invariances within the training images, producing a model better at generalization. Thus, it made a lot of sense to try inserting these more within our CNN. Our new model was thus as follows.

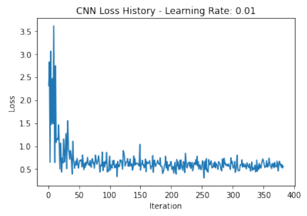
(5x5) Convolution $\rightarrow$ ReLU $\rightarrow$ MaxPool $\rightarrow$ (3x3) Convolution $\rightarrow$ ReLU $\rightarrow$ MaxPool $\rightarrow$ Affine $\rightarrow$ Softmax

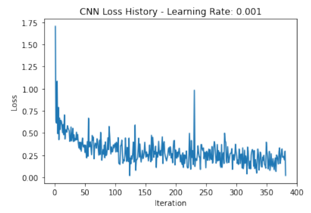
Optimizer: Choosing the right optimizer to perform gradient descent is very important in any deep learning task as they can vastly improve the rate at which your network is able to arrive at an optimal solution. Due to its impressively fast convergence capabilities, we originally began using the Adam optimiser, as discussed earlier. However, we found that Adam's optimization was too aggressive for our model, often overshooting and causing the loss to jump around rather than steadily converge. Because of this, we decided to switch to a slightly less aggressive optimiser, RMSprop. As a result of switching to RMSprop, we saw much more steady declines in our loss and an overall improved accuracy rate.

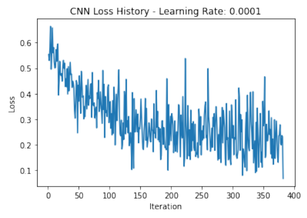
Furthermore, we also increased the size of the images passed in from our dataset from 100x100 to 258x258. This helped boost our accuracy in the deep learning approach as the convolutional model was able to extract more expressive features.

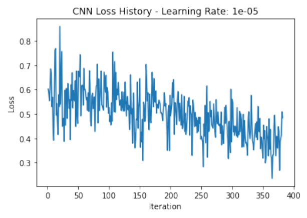
Hyperparameter Tuning: The main choices regarding hyperparameters we wanted to investigate were the learning rate, number of epochs, and hidden layer dimensions.

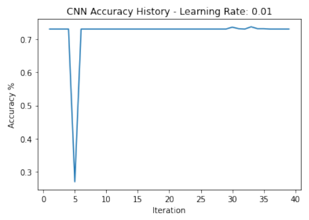
In order to determine the optimal learning rate, we ran our model on a variety of different learning rates, ranging from 0.01 to 0.00001. After rigorous testing, we found that a learning rate of 0.001 provided both the highest accuracies with some of the fastest convergence times. An analysis of the accuracies over time for the rates 0.01, 0.001 and 0.0001 are shown below. Note that for the accuracy plots, each iteration represents 10 iterations, so iteration 20 on the graph is actually 200. As can be seen from the accuracy curves below, learning rates of 0.01 and 0.00001 are not very good hyperparameters for our model, with 0.01 producing no improved accuracy from the first iteration. For 0.001 and 0.0001 we see that at around 325 iterations the accuracies begin to level out. As each epoch for us is 125 iteartions we thus decided that we should use just 3 epochs in training our CNN as more than this actually resulted in increased overfitting and a drop off in validation accuracy.

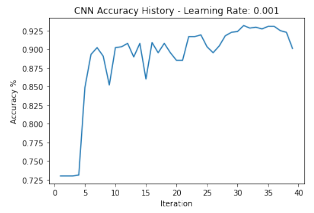


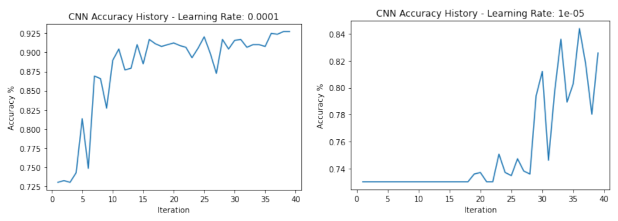












The hidden layer size was also determined in a similar way. We eventually settled upon just two hidden layers with channel sizes 24 and 12 respectively. We found this combination provided the highest accuracy whilst also minimizing overfitting.

4.4. Transfer Learning

Transfer learning is an exciting learning technique that leverages the use of large pretrained models to aid with a variety of different learning tasks. When it comes to computer vision classificiation, we can use state of the art image classification models, like ResNet and AlexNet, to aid with image classification tasks. What makes transfer training so powerful and widely used within the computer vision community is that we can use these pretrained models to classify images that they weren't even trained, so in our case X-rays.

When deciding how to integrate a pretrained model into our classification task, we had a few things to consider. Firstly, we had to decide which pretrained model to use. Secondly, we needed to determine what sort of architecture to append to the pretrained model in order for it to work with our specific classification task.

Model: We settled on the DenseNet161 model, notably due to its exceptional performance to parameter ratio. As mentioned earlier, DenseNet161 uses only around half the number of parameters as other state of the art models like ResNet whilst simultaneously providing impressive classification accuracy. The result of using DenseNet161 over other models is improved computational efficiency and increased training speed.

Classification Layer: As our dataset is not very large, we decided to freeze the whole of the pretrained model and simply add a linear classifier at the end that takes the ouput of densenet161 as its input and produces outputs two classes (Normal or Pneumonia). In this way, we are essentially using the pretrained model as an extremely powerful feature extractor and then leveraging this information to make our classificaton. The architecture is shown below.

Densenet161 $\rightarrow$ Affine $\rightarrow$ Softmax

Note that as we have freezed the layers of our pretrained model, we are only modifying the weights in our linear layer in our backwards pass.

5. Results

5.1. Naive Bayes

The success of our Naive Bayes model was varied based on our feature extraction technique. The discrete cosine transform was the most successful, with the 4 x 4 window transformation resulting in an overall accuracy of **85.1%**, and the 8 x 8 window achieving an accuracy of 83.9%. On our test set the Naive Bayes model did not seem to massively exploit any biases in our dataset as can be seen in the confusion matrix in figure 5.

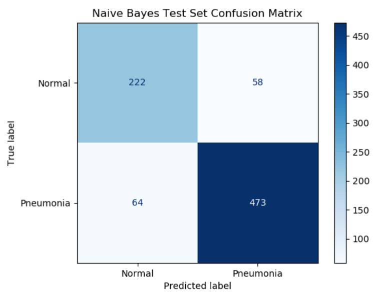


Figure 4. Naive Bayes Test Set Confusion Matrix

5.2. Support Vector Machines

When using the same feature extraction techniques as our Naive Bayes model, the SVM approach was not as effective. The linear and RBF kernel SVM models both scored an accuracy of 75.0% on the test set, while the RBF kernel model performed slightly better with an accuracy of 80.1%. When we account for Pneumonia to Normal class size discrepancy in our dataset, we realize that our SVM model was barely improving on a model that always predicts pneumonia. The confusion matrix is below.

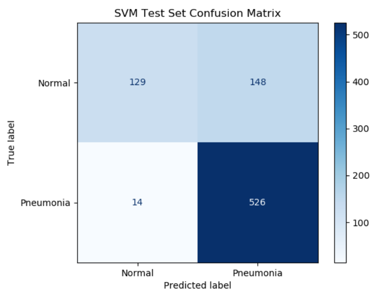


Figure 5. SVM Test Set Confusion Matrix

5.3. Convolutional Neural Network

As we expected, our CNN model significantly outperformed the more simple Naive Bayes and SVM implementations.

Furthermore, through data augmentation and Maxpooling, we were able to significantly reduce the problems we previously had with overfitting. Our final test accuracy for our CNN was **92.6%** and our train accuracy was just slightly higher at 94.3%. The CNN managed to successfully classify pneumonia much better than any other model we trained; however, it's classification accuracy for Normal was much worse than the transfer learning model. The confusion matrix can be seen in figure 9.

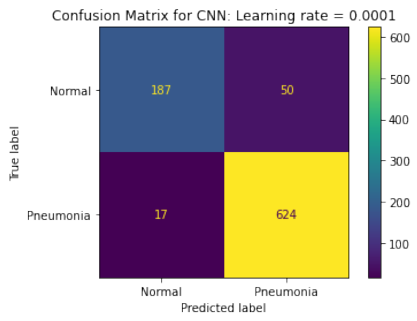


Figure 6. CNN Confusion Matrix with learning rate = 1e-4

5.4. Transfer Learning Model

The transfer model consistently provided the strong performance, with an overall accuracy of **93.3%** across our test set. It also performed notably well in classifying Normal, a problem which all of the other models ran into due to the imbalance of data in our training set. One concern is the time it takes to train this model. As DenseNet161 is such a large pretrained model, training it on our dataset took around 4 hours. The confusion matrix is shown below (figure 7) and highlights the strength of the Transfer model in dealing with Normal classification.

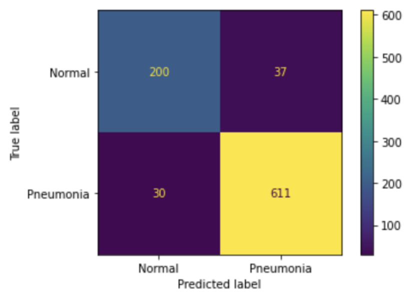


Figure 7. Transfer Learning Model Confusion Matrix

Saliency Map: Saliency maps provide us with a way of visualizing how a model made a classification decision. By looking at the strength of the gradient flow for each pixel in an example image, we can see which pixels were the most important in the model's classification decision. These pixels are shown in red, with greater intensity corresponding to greater importance in the classification. The results of the saliency map makes sense. The Transfer Learning model appears to be looking at the chest cavity the most (see figure 11), ignoring the spine, the heart and the area below the chest cavity. This is consistent to where a human doctor would look, which is incredible considering DenseNet161 wasn't even trained to classify pneumonia.

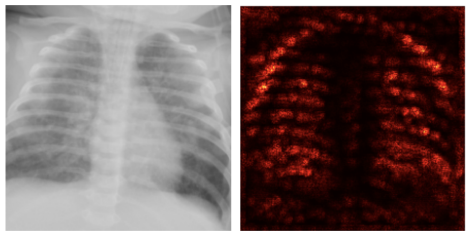


Figure 8. Example Image and Corresponding Saliency Map

6. Conclusion

To summarize our findings, we found Transfer Learning to be an incredibly powerful model and the most accurate. It was an improvement over our CNN model (second-best), and in terms of accuracy offered less false positive classifications of pneumonia (4.2% vs 5.7%) in the context of our data set, where pneumonia examples are more prevalent. More naïve models offered computationally inexpensive solutions with a large accuracy tradeoff. The role of machine learning in medicine is an exciting frontier because of the scalability and deployability of AI models. Medical AI also has some key limitations such as the sparse availability of data, susceptibility to ethnic bias, and the need for high accuracy. Through our experiments we have tried to tackle these issues by experimenting with different types of machine learning models, augmenting our data to improve generalization, and tweaking hyper parameters to continually drive up our accuracy metrics. We are incredibly happy with where we were able to bring our models, and would be excited about the opportunity to learn more about cutting edge strategies in radio graphical image classification so that we can continue to improve. Some extensions of our project that we would be interested to investigate are how well our models generalize to data sets from other ethnicities, as well as how our models could be tweaked to classify other lung conditions beyond pneumonia using a multiclass output.

7. Contributions

Work done amongst the team members happened almost entirely in the presence of all three members, with the occasional absence of one member. In other words, we always worked either all together or in pairs, so there was no clear work division.

8. References

1. Kingma, Diederik P, and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” Proceedings of the 3rd International Conference on Learning Representations (ICLR), December 24, 2014.

2. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. ”Imagenet classification with deep convolutional neural networks.” Advances in neural information processing systems 25 (2012): 1097-1105.

3. Park, Dong-Chul. “Image Classification Using Naïve Bayes Classifier.” International Journal of Computer Science and Electronics Engineering (IJCSEE) 4, no. 3 (2016).

4. Perez, Luis, and Jason Wang. ”The effectiveness of data augmentation in image classification using deep learning.” arXiv preprint arXiv:1712.04621 (2017)

5. Razavian, Ali Sharif, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition.” 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014. https://doi.org/10.1109/cvprw.2014.131.