COMBATTING OVERFITTING IN CLASSIFYING PNEUMONIA FROM X-RAYS WITH CONVOLUTIONAL NEURAL NETWORKS

By Megan Finley
CSCI 547
11 December 2019

Introduction:

Applications of machine learning is becoming increasingly popular and important in the world of medical imaging. Algorithms like convolutional neural networks (CNN's) can be extremely accurate in diagnosing images where disease is present. If trained properly, they can predict better than a trained professional. However, there is a problem that needs to be overcome to create accurate models: overfitting from class imbalance. Class imbalance is a common problem in medical data because it is often the case that the disease you are trying to predict doesn't occur that often. According to Justin M. Johnson and Taghi M. Khoshgoftaar in an article for Journal of Big Data, "When class imbalance exists within training data, learners will typically over-classify the majority group due to its increased prior probability" (2019). This will be a problem we will attempt to combat with a few methods ranging from fairly simple to more complex. The methods we will be implementing are dropout in the fully connect layer of the cnn, dropout coupled with L2 regularization, and finally dropout coupled with regularization and data augmentation. These methods will be used with two approaches to the data; treating the data as two class and treating the data as three classes. The intended purpose is to predict cases of pneumonia with better precision, recall, and accuracy than that of a physician.

Methods:

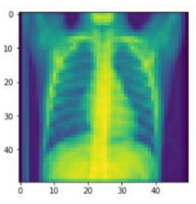
The data was obtained from Kaggle and contains x-rays of lungs, some of which are normal lungs (absence of pneumonia) and the others have pneumonia. Further, the lungs with pneumonia are classified as caused from bacteria or a virus. This distinction can be important when considering the problem of overfitting from class imbalance. In our data, there are 624 images in our testing set and 5216 images in the testing set. Within the training images, 1341 are classified as normal, 2530 as bacteria, and 1345 as virus. From here, we can decide to treat our data as having two classes, normal or pneumonia, or we can treat it is as three classes, normal, bacteria, virus. In attempt to avoid overfitting from the class imbalance, in which we would have about one-third normal images and two-thirds pneumonia images, we will first build our model with two separate classes then compare to a model if we treated the data as three classes. In addition to defining varying class models, we will take extra measures to try to combat the problem of class imbalance. Johnson and Khoshgoftaar explain that there are three ways to try to fix overfitting from class imbalance: data-level methods, algorithm-level methods, and hybrid methods (2019). The approaches to attempt to combat overfitting due to class imbalance will be dropout in the fully connected layers (algorithm-level method), dropout plus regularization(algorithm-level method), and dropout plus regularization and data augmentation(hybrid method). Data augmentation is the process of creating new data from the existing data by either rotating, shearing, decreasing or enlarging size, or a combination of all.

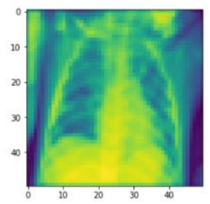
In order to prepare the data, we search the file name to determine which images are bacterial pneumonia and which are viral pneumonia. This can be done with a simple regular expression which searches the file name for a key word. Next, we need to read in our images. However, there are a few problems with the images. First, all the images are different sizes which won't work for our models since they will expect each input to have the same number of pixels, and second, some images were created as grey scale and others as color. This is a problem only encountered by the code because to the naked eye, all images look black and white. The images that are 'color' meaning they have three bands (RGB) instead of one, create problems when we try to resize the image. To bypass this problem, we convert all images to greyscale then resize them to the new desired width and height.

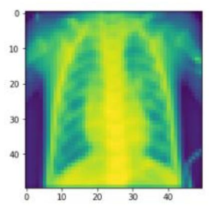
Once all problems are fixed, we can begin the process of building and training models. We will build two models for each of the three methods, one for two class data and another for three class data The first model that is implemented is a 1-dimensional convolutional neural network with dropout. The first convolutional layer takes one channel as an input, outputs 16 features, has a kernel size of two and padding size of one, which will add zeros around the edge of the image. The max pooling layer has a kernel size of two and padding size of one. The next convolutional layer takes the 16 outputs from the previous layer as inputs and outputs 64 features and has the same kernel size and padding size as the first layer. The max pooling layer is the same as well. At this point we have turned the 1-channel, 50-pixel by 50-pixel image into 40,000 features. Next, we define the fully connected layers that will yield the prediction. We take the 40,000 features we created from convolution as inputs, map to a hidden layer with 128 nodes, map those nodes to another hidden layer with 64 nodes, and finally output to two or three nodes for the prediction depending on whether we are using the two or three class data. Additionally, there are dropout layers in the fully connected network for the input layer, and the two hidden layers. The intent is to help with any overfitting by randomly selecting certain nodes to leave out during an epoch. For the feed forward part of the network, we apply the convolutions, max pooling, and dropout in the fully connected layer. In addition, every time we pass our data through a layer in the fully connected network, we apply a non-linear transformation. For this network we use the rectified linear unit activation function (ReLU). It is important to apply a nonlinear transformation because it aids the network in learning something more complex from the features. Once the network is defined, we can implement it. We create our batches, which are the sets of training data we pass through the network for each epoch. Before the data is passed through the model, we need to define a few other parameters. We need the criterion and the optimizer. The criterion is the loss function; we will use cross entropy for all models. The optimizer is how the model handles optimizing the loss function. We will use Adam for all models, with the default learning rate. The learning rate is how much the weights are updated after gradient descent. Next, we pass the data through the model and let it train. During this process we predict output, compute loss then feed the information backwards into the network to determine how much our network should change or learn. We repeat this process for as many epochs as we decide. Throughout the process we can see how the model learns by printing the loss and training and testing set accuracies. This will be repeated for every model we create.

The next model we defined was the cnn with dropout and L2 regularization. The cnn framework is the same as the previous one, our only difference is the L2 regularization term that will penalize the loss function. The purpose of the regularization term is to penalize large weights because large weights may indicate overfitting. This penalty makes the loss function larger, and thus forces the model to learn and adjust accordingly. The penalty we add to the loss function is the sum of the weights squared, times some constant gamma. The choice for gamma is less than one but greater than zero. There really isn't a method for choosing gamma but it is small. For our models, gamma is chosen to be 2e3. To implement L2 regularization, we proceed with running our model as usual, but after the loss is computed, we loop through all parameters/weights, compute the penalty, then add this to the loss before back propagation begins.

Below are a few images plotted with imshow. Since they have been rescaled, the images are a bit fuzzy compared to their original quality. However, this should work just fine for the models. We could increase the quality, but that would make the algorithms much slower.


Normal


Bacteria


Virus

Results:

The model trained with dropout for the overfitting technique was able to achieve 100% training accuracy by about the 25th epoch when trained on the two-class data. As the model trained for 50 epochs the training accuracy remained at 100% accuracy, but the testing accuracy was fairly volatile and did not achieve an accuracy rate of much more than 80%. Below demonstrates the accuracy as the model trained.

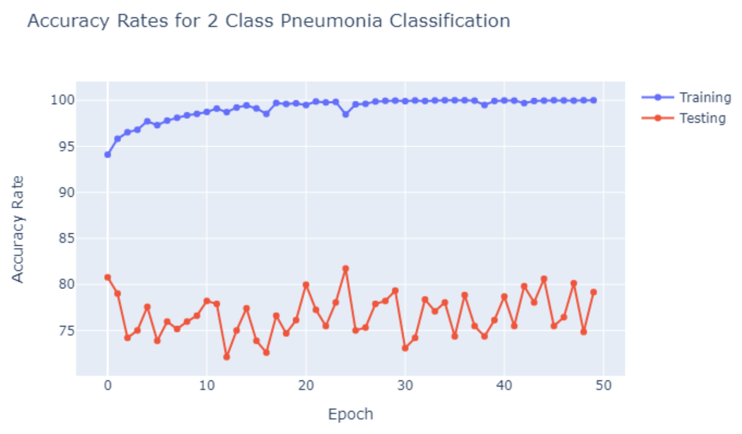


Figure 1.1

The model for the three-class data achieved a training accuracy rate of just under 100% but was less accurate for the testing data. This was a problem encountered by all the models. I believe this issue came from confusing the types of pneumonia. Below we have the results of training the three-class model.

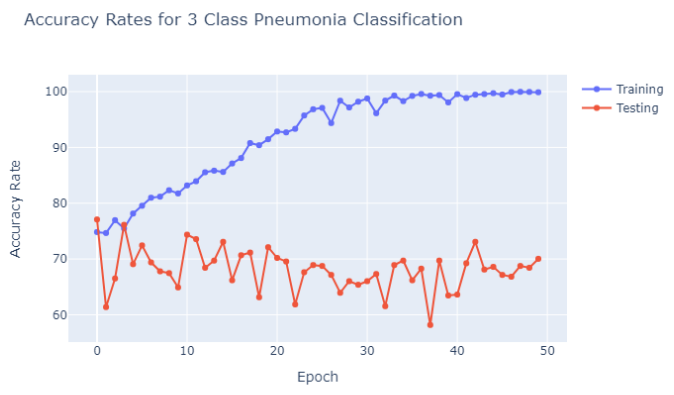


Figure 1.2

Since we don't have 100% accuracy when we test the model, we can compute a confusion matrix to see where the model is getting "confused." However, while the accuracy is of interest, primarily to see if there is overfitting and if the model has trained long enough, the metrics we are most interested in are precision and recall. Precision will tell us the proportion of true positives to true positives plus false positives and recall will tell us the proportion of true positives to true positives plus false negatives. Recall is important in medical diagnostics because if the recall is low when we are trying to predict pneumonia that means there are cases of pneumonia going undiagnosed which can be very harmful.

Below we can the confusion matrices for the two types of data. As I suspected, the model confuses the different types of pneumonia in the three-class data. However, there are quite a few cases we are still misdiagnosing.

Confusion Matrix for Two-Class Data

	Predicted		
Actual		Normal	Pneumonia
	Normal	107	3
	Pneumonia	127	387

Table 1.1

Confusion matrix for three-class data

	Predicted			
Actual		Normal	Bacteria	Virus
	Normal	103	3	0
	Bacteria	32	224	38
	Virus	99	15	110

Table 1.2

The table below summarizes the precision and recall of the models if we treat each class as the "positive."

Two-Class Data

	Normal	Pneumonia
Precision	.9727	.7529
Recall	.4573	.9923

Table 1.3

Three-Class Data

	Normal	Bacteria	Virus
Precision	.9717	.7619	.4911
Recall	.4402	.9256	.7432

Table 1.4

The model did an okay job at prediction, but we would like to see better results. We want to try to reduce the number of images that are classified as normal when they are pneumonia or a type of pneumonia. This error is costly not just for the model but also in a real-world setting.

The next model implements L2 regularization as well as dropout in hopes to fit a better model. Below are the plots for the accuracy rates as the models trained. We can see we achieve 100% accuracy for the training set but overfitting is still a problem as the testing accuracy decreases for the two-class data.

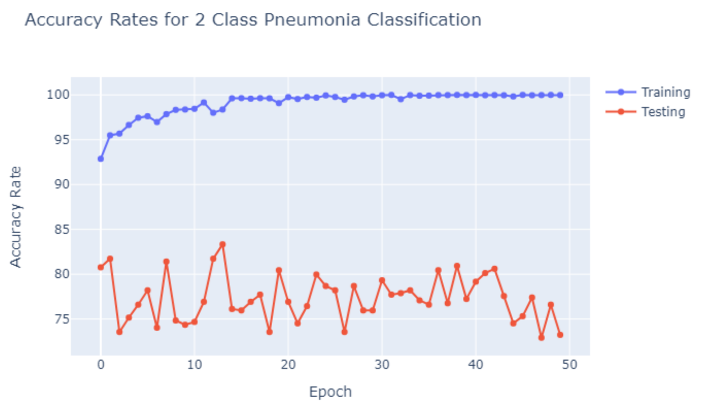


Figure 2.1

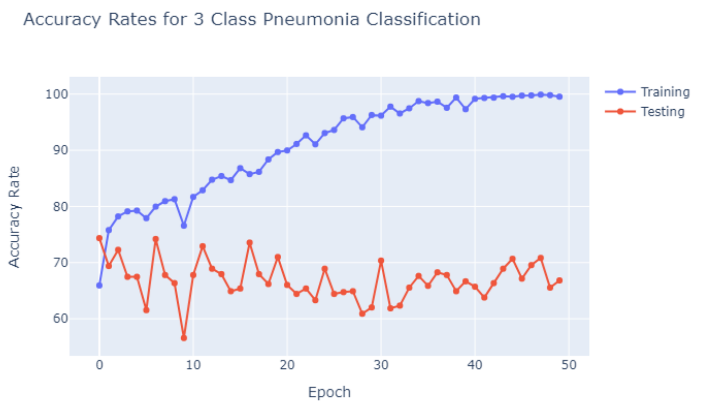


Figure 2.2

Let's take a look at where the models have problems. Below are the confusion matrices and precision and recall tables.

Confusion Matrix for Two-Class Data

		Predicted	
Actual		Normal	Pneumonia
	Normal	68	1
	Pneumonia	166	389

Table 2.1

Confusion matrix for three-class data

		Predicted		
Actual		Normal	Bacteria	Virus
	Normal	105	2	3
	Bacteria	51	234	67
	Virus	78	6	78

Table 2.2

The table below summarizes the precision and recall of the models if we treat each class as the "positive."

Two-Class Data

	Normal	Pneumonia
Precision	.9855	.7009
Recall	.2905	.9974

Table 2.3

Three-Class Data

	Normal	Bacteria	Virus
Precision	.9545	.6648	.4815
Recall	.4487	.9669	.5270

Table 2.4

We can see the model getting better with each addition to combat overfitting. From the plots, it appears accuracy isn't getting much better, but the number of images classified as normal when they aren't is decreasing.

Our final model implements the dropout, regularization, and data augmentation. Below we see the accuracy rates for the two and three-class data.

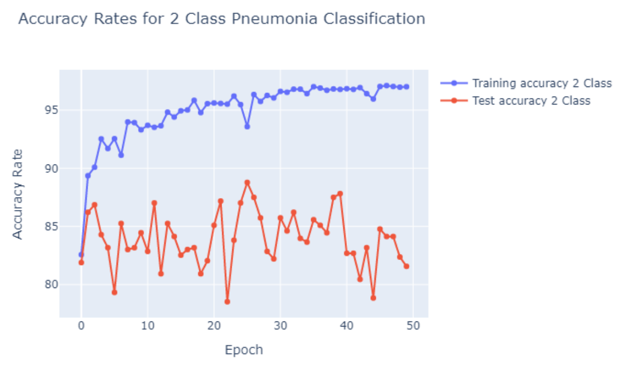


Figure 3.1

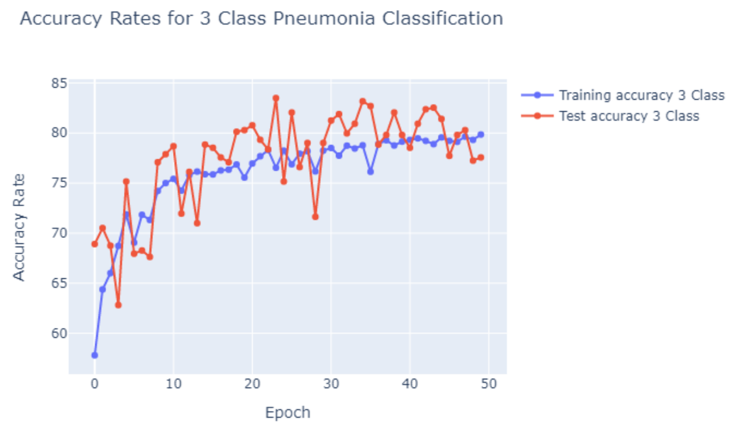


Figure 3.2

We can see in the plots that our models have not trained completely. If we were to increase the number of epochs, we might see better results. However, this model is computationally expensive and already takes a long time to run. In the tables below we can see that the models do well at predicting pneumonia.

Confusion Matrix for Two-Class Data

		Predicted	
Actual		Normal	Pneumonia
	Normal	121	2
	Pneumonia	113	388

Table 3.1

Confusion matrix for three-class data

		Predicted		
Actual		Normal	Bacteria	Virus
	Normal	140	2	1
	Bacteria	39	237	40
	Virus	55	3	107

Table 3.2

The table below summarizes the precision and recall of the models if we treat each class as the "positive."

Two-Class Data

	Normal	Pneumonia
Precision	.9837	.7745
Recall	.5171	.9947

Table 3.3

Three-Class Data

	Normal	Bacteria	Virus
Precision	.9790	.75	.6485
Recall	.5983	.9793	.7230

Table 3.4

Conclusion:

I think we can be satisfied with our results overall. According to one study, by Wipf JE et al., clinical diagnosis, those where no technology is used in to determine the diagnosis, doctors diagnosed pneumonia with sensitivity of 47% to 69% and specificity of 58% to 75% (1999). While these diagnoses were confirmed with x-rays, our model solely relies on images without the access to physical condition. In another study by Wooten and Feldman, diagnoses had a sensitivity (recall) of 95% when diagnosing pneumonia by using chest radiographs and symptoms (2014). In that study, they were just looking for the presence of a respiratory infection/inflammation and were not considering various causes of the pneumonia. We achieved a 99.47% recall with the two-class data, and 97.93% recall for bacteria and 72.3% recall for virus. Our model was able to achieve better results than a trained professional. In addition to producing acceptable metrics, we were able to overcome overfitting from class imbalance. Further directions may include increasing the quality of the image and training for a longer amount of time.

Works Cited

Johnson, Justin M., and Taghi M. Khoshgoftaar. “Survey on Deep Learning with Class Imbalance.” Journal of Big Data, vol. 6, no. 1, 2019, doi:10.1186/s40537-019-0192-5.

Kermany, Daniel; Zhang, Kang; Goldbaum, Michael (2018), “Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification”, Mendeley Data, v2, doi:10.17632/rscbjbr9sj.2.

Wipf, Joyce E., et al. “Diagnosing Pneumonia by Physical Examination.” Archives of Internal Medicine, vol. 159, no. 10, 1999, p. 1082., doi:10.1001/archinte.159.10.1082.

Wootton, Dan, and Charles Feldman. “The Diagnosis of Pneumonia Requires a Chest Radiograph (x-Ray)–Yes, No or Sometimes?” Pneumonia, vol. 5, no. S1, 2014, pp. 1–7., doi:10.15172/pneu.2014.5/464.

*All code was created or taken from class examples and homework.