Classification of Bacterial and Viral Pneumonia of Children's Upper Torso X-rays

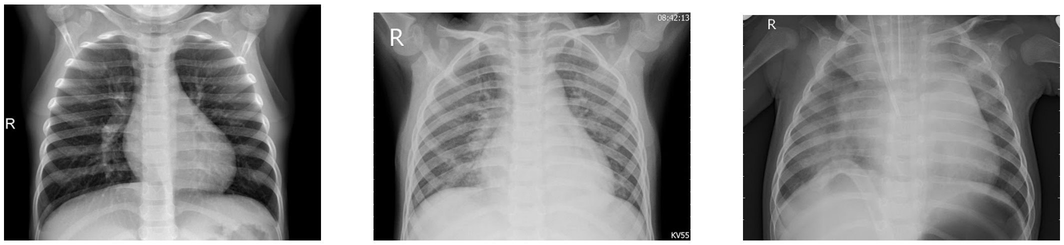
CSCI4155 Project Report
Ryan

1 Introduction

For this project, we looked at convolution neural networks in the context of classifying different pneumonia infections in children's upper torso x-ray images. The team is impressed with how well machine learning can be adapted to medical science and medical imaging problems in particular. We figured that this pneumonia problem would provide an interesting area of exploration for this project. The X-ray imaging approach can be a really good alternative to blood test for detecting pneumonia. It's not just limited to binary detection but goes out a step further and even help classify whether its a viral or bacterial infection so that patient can be prescribed medication and treatment accordingly to ensure a speedy recovery.

2 Our Data Set

Our data-set was retrieved off Kaggle and contains X-ray chest images separated into folders 'normal' and 'pneumonia'. The original objective of this data-set was to differentiate between healthy lungs and infected lungs. However, we noticed that the 'pneumonia" folder had two kinds of infections: viral and bacterial. We thought it would be interesting to try and also differentiate between the different kinds of pneumonia, seeing as there are noticeable differences between the two kinds of infections.

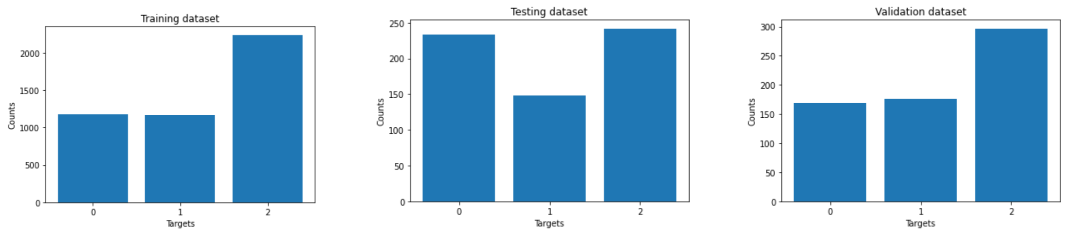


(a) Healthy lungs (b) Viral Pneumonia (c) Bacterial Pneumonia

Figure 1: The three classes

In a healthy X-ray (Figure 1a), there is no chest congestion and the borders of the heart are fairly well defined. In case of viral pneumonia (Figure 1b), x-ray image looks blurry as a result of inflamed tissues, and in bacterial pneumonia (Figure 1c), cloudiness increases in the x-ray image as inflammation spreads out further. These clearly visible differences led us to believe that it might be beneficial for the model to separate the infected classes. The images we picked are the most clearly distinguishable for their classes, but there is some ambiguity in some of the images where for our untrained eye, it was not easy to know which class the image belonged to. If that were not the case, this would have made for a very easy project.

3 Training, Validation and Testing Proportions



(a) Training classes (b) Testing classes (c) Validation Classes

Figure 2: Class Proportions

We noticed the disproportion in classes as is shown in Figure 2 plotted for image counts against targets where 0 refers to Healthy, 1 refers to viral pneumonia, and 2 refers to bacterial pneumonia. Upon digging deeper in our data set, we observed that in the pneumonia folder, there were sometimes multiple x-rays for the same infection. We therefore had to manually transfer over some images from our training set to the validation set as to prevent our validation data to have images that are very similar to training images and our validation set was very small. To help further mitigate the effects of class imbalance, we utilized class weights.

4 Data Prepossessing

4.1 Image Cropping and Resizing

Each image in our data-set has different dimensions. Most of the images were horizontally rectangular, but some of them were vertically rectangular. We started by cropping the images on their longest dimension to give them square proportions. Since the sides of the images were mostly black and not actually part of the child's chest, cropping did not remove much information. Then, we converted to grey-scale (lossless as the images are already gray-scale), and finally, we resized them to a size of 224x224. We chose this size as the model we will be trying transfer learning on uses these dimensions.

4.2 Sample-wise Min-Max Normalization

While examining our data-set, we noticed that some images were not as bright as others. This didn't seem to be have anything to do with the chest being healthy or infected, so we tried to normalize the image brightness by doing min-max normalization on each sample.

5 Baseline

As a baseline, we used the same preprocessing as on our neural networks, and we ran them through an SGDClassifier from scikit-learn. Our accuracy with this classifier was 62%. Figure 3 shows the confusion matrix.

6 Convolutional Model

6.1 Architecture

The convolutional neural network in the above figure receives the cropped image shape of 224x224x1. The six convolutional layers are composed of three pairs where the filter sizes are 64, 128 and 256 respectively. Each convolutional layer uses a kernel size of 3x3 and the rectified linear unit activation function. The max pool layers use a pool size of 2x2 and padding that retains the input size for the output. Given that this model distinguishes between three different classes, the final dense layer has an output shape of three in conjunction with the soft-max activation function. The model uses the Adam optimizer and categorical cross entropy for measuring loss.

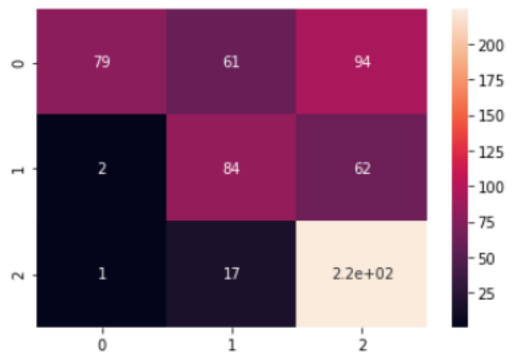


Figure 3: Baseline Confusion Matrix

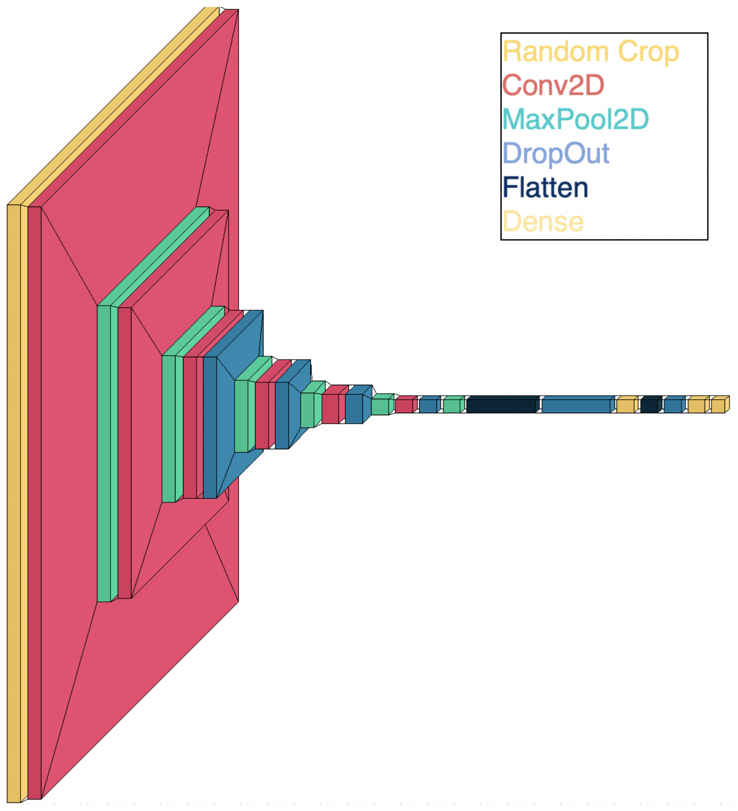


Figure 4: CNN Model Architecture

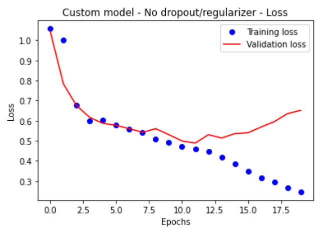
6.2 Training

As a sanity check, we ran our model with no dropout or regularization. As can be seen in Figure 5, at first, the training and validation accuracy match, but at around 10 epochs, the results for the training set keep getting better, while the validation set results start getting worse.

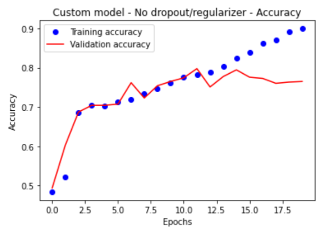
When we added the regularization and dropout, we got slightly more stable results, but our test set accuracy was mostly the same. It was slightly better but that could be due to random chance.

7 VGG-16 Imagenet Transfer Learning

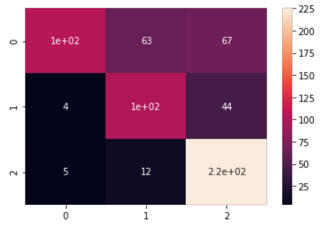
We used the VGG16 library provided by Keras to apply transfer learning to the data set. We removed the top layer, and all layers within the VGG network were set to not trainable. We added two



(a) Loss

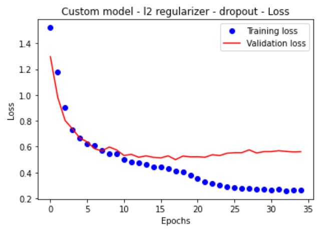


(b) Accuracy

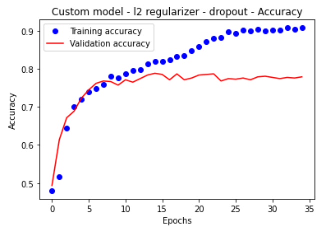


(c) Confusion Matrix

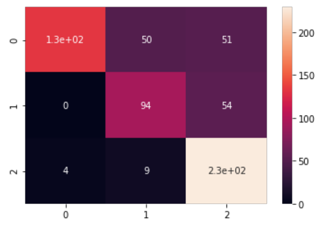
Figure 5: Convolutional Neural Network with no Regularization or Dropout



(a) Loss



(b) Accuracy



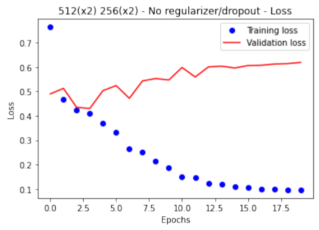
(c) Confusion Matrix

Figure 6: Convolutional Neural Network with Regularization and Dropout

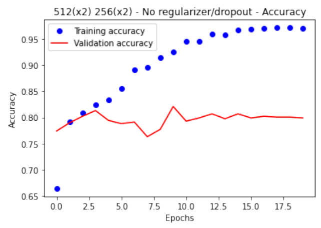
512 Dense layers followed by two 256 Dense layers. We continued to use the Adam optimizer and ReduceLROnPlateau provided by Keras. Reducing the learning rate during training helped the model converge faster and achieve a lower training loss.

7.1 No L2/Dropout

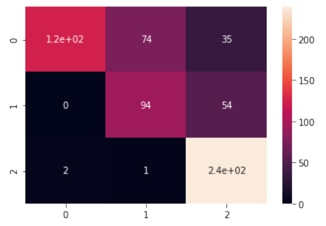
In our first experiment, we included no dropout or regularizers and a batch size of 64. The model did a great job fitting to the training data as it achieved 0.09 loss and 97% accuracy within 20 epochs, however, our validation loss could not get below 0.43 loss and 81% accuracy. This followed through to our test set, where we achieved a loss of 0.85 and accuracy of 73%. When we re-trained the network, we managed to achieve a validation loss of 0.4137 but our test loss ended up being 1.0270 with an accuracy of 72%.



(a) Loss



(b) Accuracy

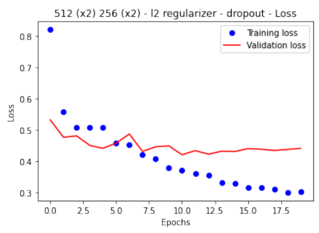


(c) Confusion Matrix

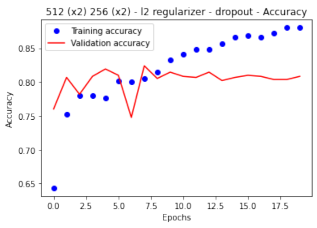
Figure 7: VGG-16 With No L2/Dropout and Batch Size 64

7.2 L2/Dropout

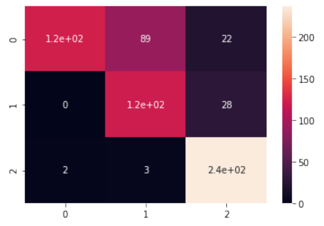
In an attempt to achieve a lower validation loss and reduce over fitting we introduced L2 kernel regularizers as well as dropout. We did experiment with the L1 kernel regularizer, however, that caused the model to under fit and proved to be no help at all. With the addition of L2 and dropout, we managed to get the validation loss slightly more in sync with the training loss and keep the validation loss low for more epochs. However, in the end, we were not able to achieve a validation loss that was lower than the previous method. The failure to improve our model was evident as well in our test results, where we achieved a loss of 1.00, but a better accuracy of 77%. This kind of pattern was common in our experiments where a lower loss did not always guarantee our accuracy would be higher.



(a) Loss



(b) Accuracy

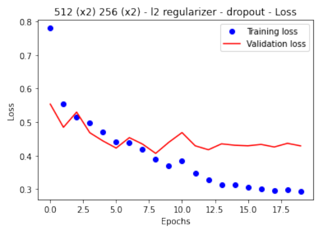


(c) Confusion Matrix

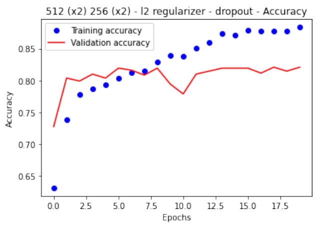
Figure 8: VGG-16 With L2/Dropout and Batch Size 64

7.3 Increasing Batch Size to 128

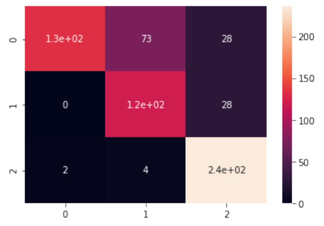
In another attempt to improve our model, we increased the batch size to 128. We were required to reduce the learning rate from 0.001 to 0.0005 since our model early stopped at 0.001. The difference between batch size 64 and 128 was very little, if any. Our lowest validation loss achieved was 0.4069 which was very close to our second time training with a batch size of 64. This improvement was most likely not a result from the increase in batch size. In addition, this did not prove to make much of a difference in our test results, as we achieved a loss 0.01 higher but a better accuracy of roughly 1%. In addition, when we re-trained the network we achieved a validation loss of 0.4133, proving that the difference may be attributed to how the model converged that time. Overall, generalizing our model further did result in a rather consistent improvement in our test accuracy, but no improvement was made to the validation loss, other than allowing it to stay low for more epochs.



(a) Loss



(b) Accuracy



(c) Confusion Matrix

Figure 9: VGG-16 With L2/Dropout and Batch Size 128

8 Conclusion

During this project, we wanted to explore image classification with convolutional neural networks and transfer learning. One of our team members being interested in radiology as a career, we thought that using the chest x-rays data-set from Kaggle and expanding it to the 3 classes it has instead of simply a binary healthy/infected classification would be perfect.

We spent a lot of time examining the x-rays in our data-set to figure out the best way to preprocess them. Seeing as the sides were mostly all-black, center-cropping them to a square seemed reasonable.

Our custom convolutional network did not perform that great if simply looking at the confusion matrix, but on a practical level, it is not completely useless. Figure 6c shows that even on the test set, our model has almost no false negatives, which is good because it could be used as an initial screening process, where negative predictions are given less attention (not none), and positive predictions are more closely examined to confirm the diagnostic.

Our experiments with the VGG transfer learning gave slightly better, but similar results to our custom convolutional network.

To conclude, while we were not able to get our models to perform as well as we would have liked, we got results that weren't completely useless, and it was very interesting to explore and learn about various preprocessing techniques, convolutional neural networks, memory management, and transfer learning.

9 GitLab Link

https://git.cs.dal.ca/sawchuk/csci4155-project/-/tree/master

10 References

1. Data set: Mooney, P. (2018, March 24). Chest x-ray Images (Pneumonia). Retrieved February 27, 2021, from https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
2. Thakur, R. (2020, November 24). Step by step VGG16 implementation in Keras for beginners. Retrieved February 24, 2021, from https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c
3. Steegstra, P. (2018, April 10). Tutorial: image classification with scikit-learn. Retrieved February 24, 2021, form https://kapernikov.com/tutorial-image-classification-with-scikit-learn/