Effects of Data Augmentations on Binary Classification of Leukemic Images

Taasin Saquib
UCLA

Marlin Rodriguez
UCLA

1 Abstract

We attempted to use data augmentations to boost performance for classifiers choosing between ALL (diseased) and HEM (healthy) white blood cells. We found that larger networks such as InceptionV3 and Resnet50 overfit and simply always guess diseased. Using different augmented datasets to perform transfer learning on a smaller network didn't show much difference in accuracy. Finally, by accounting for class imbalance in the loss function and re-training all layers in the smaller network we saw that adding noise resulted in the highest accuracy.

2 Introduction

Leukemia is a cancer of blood-forming tissues[1], and our project deals with a specific form of the disease known as acute lymphoblastic leukemia, or ALL. The 'acute' name means that the disease spreads quickly, and the success of treatment depends on early detection. Machine learning has promising applications in healthcare, and we set out to find if convolutional neural networks (CNNs), which are typically used for image recognition, could be used to differentiate between diseased and healthy white blood cells.

Good training data is imperative to get good performance in these classification tasks but can also be expensive and difficult to obtain. Because our data was relatively limited, we explored data augmentation to boost classification performance. After applying simple transformations to our data, we observe if they have an impact on the accuracy of our CNNs.

3 Dataset

Our dataset is taken from Kaggle[4] and consists of images for two classes of cells, ALL (diseased) and HEM (healthy). We've included examples of both classes in Figure 1. The testing data unfortunately did not come with existing labels and therefore could not be used. As a result, we had decided to split the data ourselves. First, we took the validation dataset and simply used that as our test dataset. Next, we put 20% of our training data aside to use as validation data while training.

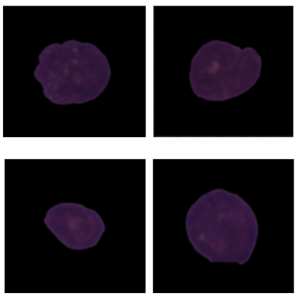

Figure 1: ALL cells (top) and HEM cells (bottom)

Another potential problem with the dataset is that there are about two times as many ALL cells as there are HEM cells, which could affect classification accuracy. Therefore, data augmentation and other techniques that we will discuss later become more important.

After initial visual inspection, it seems that the diseased cells do not have many notable differences from the healthy ones. This turns out to be one of several challenges of image recognition for cellular images. In fact, an expert oncologist is required to label these cells correctly.

4 Methods

4.1 Neural Networks

Initially we decided to use InceptionV3[2] as our baseline model. This was in part because it was used in a Kaggle notebook associated with our image data and

had pre-trained weights that we could download. Moreover, InceptionV3 is a network architecture that is commonly studied for applications of image classification and recognition. The model is defined and implemented from Tensorflow Keras API calls.

We also wanted to compare the prediction accuracies between multiple network models. Because of this, we decided to compare our InceptionV3 model with ResNet50[3]. We wanted to determine if the residual skip layers would help with tracking features that get lost during training and inference. Like InceptionV3, ResNet is also widely studied for its use in image classification. We also implemented the ResNet model using Tensorflow Keras API calls and used pre-trained weights from ImageNet.

Because both models were very large and we did not have extended access to hardware accelerators, we decided to use transfer learning on both of these models to fine-tune to our dataset. One observation that we made during training was that both models were overfitting. While this will be discussed more in detail later in this paper, one cause for this behavior is likely the depth and complexity of these networks. In order to have more meaningful results for which we could test our augmentations, we looked for other architecture options. We decided to use a custom CNN model from another Kaggle[7] notebook. This architecture is less complex, with 25 layers and about 16 million parameters. Like the previous two models, our new model consists of a combination of CNN, activation, and pooling layers.

Everything was implemented in Python3.6, using Google Colab to write notebooks and using GPU resources. Our code can be found at: https://github.com/taasinsaquib/ALL-Augmentations

4.2 Augmentations

CNNs depend on having access to a large amount of training data to extract different features that can distinguish between classes. Often, however, there are not enough training images. In these cases, researchers often apply different augmentations to images such as rotation, flipping, blurring, etc. This is a way to synthesize new data that the CNN can be trained on. It also reduces the chance of overfitting, as various image artifacts make sure that the training images cannot be "memorized."

In our case, we wanted to avoid overfitting as all the images are very similar. We decided to examine the effects of a few types of augmentations: flips, rotation, blur, and noise. We have examples of some augmentations in Figure 2. It is important to note that sometimes certain augmentations don't make sense to use; in the case of autonomous cars vertically flipping a road would confuse the driving engine. Rotation and flipping are easy ways to create the illusion of new cells because our data is mostly in the form of circles. Blur is usually used to make the image less sharp and allow the CNN to examine overall trends. Noise is usually used for its regularizing effect on the network weights.

We do the augmentations in-place, meaning that the number of training images is not increased. Instead, we apply our augmentations with a 33% chance of replacing an image with its augmented version.

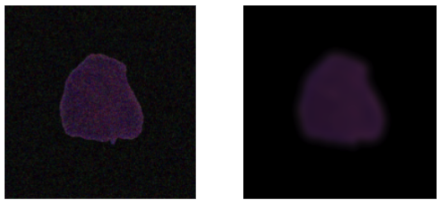

Figure 2: Augmented cells with noise and blur

4.3 Experiments

To start, we used InceptionV3 and Resnet50 as our baselines. As mentioned before, we performed transfer learning using our newly split training data. Our goal was to see which augmentations could perform better in terms of loss function value, accuracy, and/or precision on the test set.

For our custom, smaller CNN architecture, we also performed transfer learning. This was done by freezing the weights in the first 19 layers, leaving 6 layers to be tuned. We trained different models for each augmentation to evaluate if different augmentations had different effects on performance.

Finally, we accounted for the class imbalance by modifying the loss function and retraining all layers of the custom model. To determine the ideal number of epochs to use, we trained using non-augmented data in increments of 5 epochs and watched for signs of overfitting (decreasing validation accuracy with increasing validation loss). Once we found a suitable number of epochs to use, we trained different classifiers on each type of augmentation.

5 Results

During training of both the InceptionV3 and ResNet models, we noticed that both models were overfitting to the training data early on. As shown in Figure 3, validation accuracy drops and validation loss increases per epoch until not changing at all. As a result, when presented with testing data, both models would classify all of the test images as 1 or ALL as shown in our confusion

Data Augmentation	Loss	Accuracy	Precision
Baseline (Inception)	6.9853	0.6529 ± 0.02159	-
ResNet	2.9765	0.6535 ± 0.02158	-
Flipped	2.1606	0.3996 ± 0.02219	0.6303
Noise	2.0918	0.3519 ± 0.02176	0.7647
Blur	1.4774	0.3621 ± 0.02163	0.6707

Table 1: Comparison of loss and accuracy for transfer learning models

matrices for each classifier in Figure 4. This behavior would account for the high accuracy for each classifier in Table 1 and Figure 5 as most of the test labels are actually ALL. However, these results are not meaningful since if a test image is not ALL, then the classifier's prediction would almost always result in a false positive. While this means that both models could not learn how to distinguish between the two classes, we see that InceptionV3 has no false negatives and ResNet50 has very few.

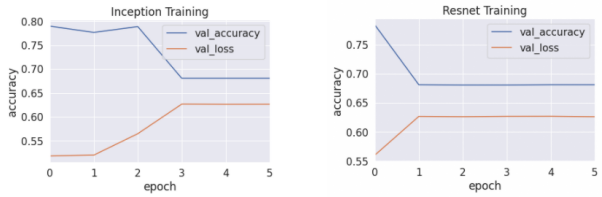

Figure 3: Validation loss and accuracy for InceptionV3 (left) and ResNet (right) during training.

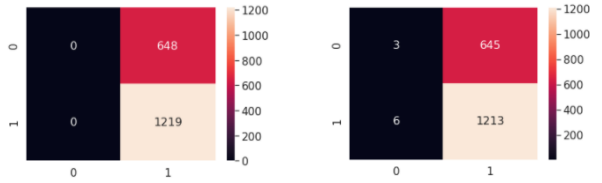

Figure 4: Confusion matrices for InceptionV3 (left) and ResNet model (right)

Next, as shown in the bottom three rows of Table 1, we evaluate the custom models that use transfer learning in order to compare their performance to InceptionV3 and ResNet50. At a significance level of 95%, We found that the model with flipped data augmentations resulted in the best prediction accuracy. However, blur data augmentations result in the best loss. One observation that we made from the data provided is that for the data augmentations, implementations with better prediction accuracy generally result in worse precision. It appears that different augmentations introduce data artifacts that would make our classifiers more unsure of their decisions. This would make sense as in each augmentation, we are introducing images within our training set that appear more "different" than the test images which the classifiers are being tested on.

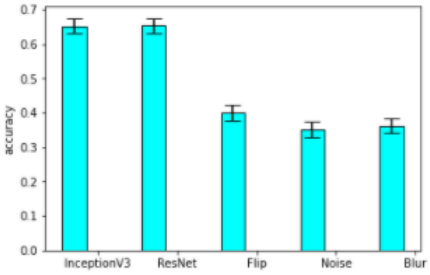

Figure 5: Accuracy comparison for our transfer learning models vs the baseline

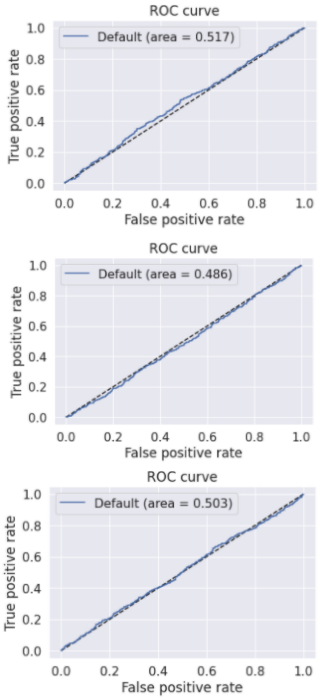

Figure 6: ROC curves for flipped (top), noise (middle), and blur (bottom) augmentations

The ROC curves for the flipped, noise, and blur data augmentations are shown in Figure 6. These plots help reinforce the observation we made from the table above that out of all of the data augmentation strategies, flips have resulted in the best prediction accuracy. While these models were less prone to overfit than InceptionV3 and ResNet, all three plots were close to the 45-degree line indicating that the accuracies of the transfer learning models were not very good. Despite these results, the AUC values for InceptionV3 and ResNet were 0.501 and

0.499 respectively. This indicates that while prediction accuracies are not as high as InceptionV3 and ResNet for the test data, our models are as effective given that these models are able to distinguish between ALL and HEM images. Our classifier trained on flipped augmentations was most successful in pushing the ROC curve to the left of the diagonal, suggesting that these augmentations were effective.

When doing full retraining of our custom network, we found that performance does not increase much after about 5 epochs. Our results in Table 2 were interesting as all three fully-retrained classifiers have higher accuracy on our test data than the classifiers that did not account for class imbalance. These classifiers also improve upon the classifier that was trained on non-augmented data as shown in Figure 7. However, our loss function values are much higher and the precision is lower. It appears that the augmentations introduced artifacts that made our classifiers more unsure of their decisions as compared to the baseline.

Data Augmentation	Loss	Accuracy	Precision
No Augmentation	2.85591	0.36047 ± 0.02177	0.73584
Flip & Rotate	4.09511	0.37707 ± 0.02198	0.63333
Noise	4.79922	0.39314 ± 0.02215	0.64827
Blur	7.65976	0.38778 ± 0.02210	0.63194

Table 2: Comparison of loss, accuracy, and precision using our modified loss function

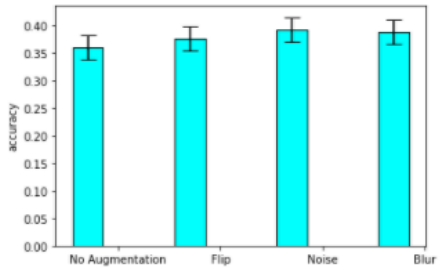

Figure 7: Accuracy comparison for our retrained models

We compare our best performing classifier, noise, to the classifier without data augmentations using ROC curves and confusion matrices in Figures 8 and 9. Again we notice that the ROC curves are close to the 45-degree line, but the noise curve is slightly pushed to the left. This shows that while the predicted accuracy is not very high, it shows some improvement upon previous training methods. Interestingly, from the confusion matrices we can see that the classifier without augmentations predicted a class label of 0, or HEM, for all of the training data. This is the exact opposite of what the InceptionV3 classifier did. Our noise classifier has many false negatives, which makes this classifier unusable when diagnosing diseases. However, we did get high performance through the noise augmentation and modified loss function.

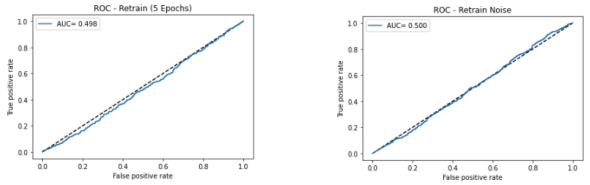

Figure 8: ROC curves for our classifier re-trained with normal data (left) and noise augmented data (right)

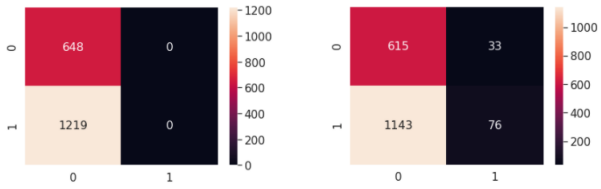

Figure 9: Confusion matrices for our classifier re-trained with normal data (left) and noise augmented data (right)

Because our output node is a sigmoid function, it outputs a likelihood for each data point. Our results thus far have used a threshold of 0.5, with confidences lower than 0.5 being classified as HEM and above as ALL. In figure 10 we explore different thresholds to see if we can get better separation of the predictions. We see that many predictions have confidences between 0 and 0.1, accounting for the large drop in the graph. The classes are not separated very well, accounting for the almost flat accuracy metrics across thresholds 0.3 to 0.7.

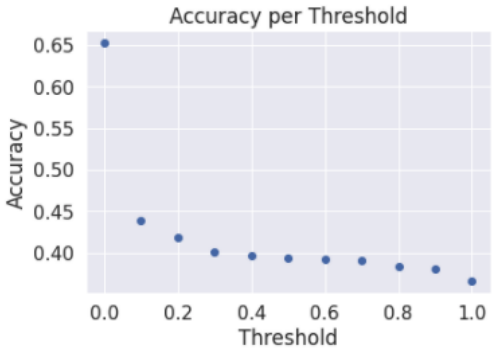

Figure 10: Accuracy of the class imbalance classifier trained with noise augmentations.

6 Conclusion

Assuming a 95% significance level, our hypothesis that data augmentation would improve the performance of image classification is not statistically significant. We tried modifying the loss function for class imbalance to see if it would help improve image classification. However, we observe that the error bounds for the accuracies intersect with each other. While the overlap is very minimal, it still deems these results as not significant. While data augmentations did help improve measured prediction accuracies, there was a tradeoff between the prediction accuracy and precision. When using a neural network model, one challenge was that it is easy for our network to overfit if it has too many layers or epochs. For this application, CNNs might not be the best approach for image classification as the images which we are analyzing are so similar. Because of this, we may want to add other features in addition to images that can help the models produce more accurate results. This paper helps to reinforce that image classification on biological cells is challenging. However, we were able to produce interesting results which can provide useful insight for future biomedical research.

7 Future Work

For future work, we would modify our custom architecture and take inspiration from larger models. Looking at ResNet, for example, we would like to implement skip layers to smoothen the gradient calculation and allow lower level features found early in the CNN to influence the classification. Another idea would be to train specialized neural networks on different parts of the image, such as different quadrants of the image or the perimeter of the cell. Bagging methods could be used to put these results together. Also, our augmentations were relatively simple and we would like to explore more complex augmentations such as vertical and horizontal shifts, brightness, masking, and addition of borders.

References

[1] Inaba H, Greaves M, Mullighan CG. *Acute lymphoblastic leukaemia*. Lancet. 2013;381(9881):1943-1955. doi:10.1016/S0140-6736(12)62187-4

[2] Christian S, Vincent V, Sergey I, Jonathon S, Zbigniew W *Rethinking the Inception Architecture for Computer Vision* arXiv. 1512.00567

[3] Kaiming H, Xiangyu Z, Shaoqing R, Jian S *Deep Residual Learning for Image Recognition* arXiv. 1512.03385

[4] Gupta, A. and Gupta, R. *ALL Challenge dataset of ISBI 2019 [Data set]*. The Cancer Imaging Archive. https://doi.org/10.7937/tcia.2019.dc64i46r

[5] Rahul Duggal, Anubha Gupta, and Ritu Gupta. *Segmentation of overlapping/touching white blood cell nuclei using artificial neural networks*. CME Series on Hemato-Oncopathology, All India Institute of Medical Sciences (AIIMS), New Delhi, India, July 2016.

[6] Rahul Duggal, Anubha Gupta, Ritu Gupta, and Pramit Mallick. *SD-Layer: Stain Deconvolutional Layer for CNNs in Medical Microscopic Imaging*. In: Descoteaux M., Maier-Hein L., Franz A., Jannin P., Collins D., Duchesne S. (eds) Medical Image Computing and Computer-Assisted Intervention MICCAI 2017, MICCAI 2017. Lecture Notes in Computer Science, Part III, LNCS 10435, pp. 435–443. Springer, Cham. DOI: https://doi.org/10.1007/978-3-319-66179-7_50

[7] https://www.kaggle.com/anteii/all-classifier