Ruchi Sarkar, Nick Tapp-Hughes, Marcos Zhang Solis
Introduction to Machine Learning
Dr. Jorge Silva
May 3rd, 2020

Using Neural Networks to Classify White Blood Cell Images

1. Introduction

Blood cell imaging has been an important tool in diagnosing blood-based illnesses. The classification of white blood cell types in a patient's blood is pivotal to understanding the immune response. Neutrophilia, for example, is a condition caused by having an excess of white blood cells (Rifkin, 2020). In order to diagnose this condition, a doctor must be able to correctly differentiate neutrophils from other cells in the body. Manually classifying blood cells is always prone to errors, inconsistencies, and subjectivity, but white blood cells are especially prone to this because their structures are not inherently stable (Elen & Turan, 2019). For this reason, computer-based alternatives are being developed in order to provide faster and more accurate results.

Previous researchers have used a variety of machine learning algorithms to classify blood cells in the body. Sanei & Lee used PCA and Bayesian classification to sort bone marrow cell images into roughly 40 classes. Their algorithm classified mature cells with 96% accuracy and immature cells with 85% accuracy (Sanei & Lee, 2003). Sarrafzadeh et al. used an SVM with 18 features relating to color, geometry, and statistical properties to classify white blood cells with 93% accuracy (Sarrafzadeh et al., 2014). Theera-Umpon & Dhompongsa used Bayesian classifiers and an artificial neural network to classify leukocytes with 77% accuracy (Theera-Umpon & Dhompongsa, 2007). In this project, we train a neural network to classify images as 1 of 4 different white blood cell types: neutrophil, eosinophil, lymphocyte, and monocyte. We attempt multiple trials to find the best model by using different loss functions in each trial: sparse categorical cross-entropy, mean squared error, and Huber.

2. Methods

We used images from the Blood Cell Images dataset on Kaggle, which contains about 12,500 unique images of shape (240, 320, 3). Approximately 10,000 of the images were used for training while 2,500 images were used for validation. NumPy was used to shuffle and normalize the image data as well as generate the labels according to the file paths of each image, and the Keras library in TensorFlow was used to build a neural network as the model. Our classification network took the form of the first half of a U-net (an "encoder"), with 8 convolutional layers, 4 pooling layers, and a final dense layer with 4-dimensional output, with a total of 150,012 parameters. ADAM (as implemented in TensorFlow) was used for optimization. We were interested in whether using different loss functions would affect the accuracy of the model, and to what degree. Three models were created, differing only in the loss function that was used:

sparse categorical cross-entropy (from logits), mean squared error, and Huber. The models were trained on the same training set and validated on the same validation set.

Training was completed on the UNC Longleaf research computing cluster, using one NVIDIA Tesla V100 16GB GPU, one CPU, and 20 GB memory. Each model was trained for 100 epochs, taking roughly 5 minutes each. During training, the loss, accuracy, validation loss, and validation accuracy were recorded, where accuracy is the proportion of images that were classified correctly. These values were then plotted to compare the validation accuracy across all three loss functions, as well as compare the training vs. the validation accuracy within each trial. These graphs can be located in the Appendix.

3. Results

It was found that the sparse categorical cross-entropy loss function provided the best accuracy on both the training and validation sets. Figure 1 shows that the validation accuracy for sparse categorical cross-entropy converges around 70% accuracy , while Figures 2 and 3 show that mean squared error and Huber loss respectively both converge around 25%. Similarly, Figures 2 and 3 also show that the model was not able to accurately predict the training data either - the training accuracy for both models also converged around 25%. However, the training accuracy for sparse categorical cross-entropy converged to 1.0, or 100% accuracy.

4. Discussion

The results showed that Huber and mean squared error loss functions had low accuracy both in the training and validation sets. There are multiple reasons why this could have been the case. There were only 10,000 samples in the training data, which may not have been enough data to expose the model to the full range of features that the validation set could have had. Furthermore, it may have been the case that 100 epochs was too short a time for the network to achieve high accuracy. Typically, a plateau in accuracy during loss can occur before a "breakthrough," where the model's accuracy starts to increase. We also acknowledge the possibility of unlucky initialization of the model parameters combined with an improper learning rate. It's possible that the loss optimization algorithm was trapped in a local minimum. Given that Figures 2 and 3 both appear to plateau around 25% without later increasing, it is possible that these loss functions could have been more accurate if the model had trained over more samples and/or more epochs.

We conclude that the model with sparse categorical cross-entropy loss was overfitted to the training data, as the training accuracy was 100% while the validation accuracy converged to around 70%. This may have been due to the number of parameters in the model--150,012--which was large enough to encode the features of the training set. While the large number of parameters is hard to avoid when working with image data, it is reasonable to assume that these parameters were able to account for every peculiarity and variation in the training data, which may have not been present in the validation data, making the training accuracy perfect while making the validation accuracy much lower.

Another reason the accuracy level may have been low for this trial is that the choice of labels is not well-suited to the problem. Another set of researchers who trained a neural network to classify white blood cell images assigned the images to 6 classes rather than 4 and were able to achieve 99% accuracy (Hegde et al., 2019). One of the 6 classes accounted for abnormal cells that do not easily fit into other categories - the Kaggle dataset may have also included abnormal cells that could not be reliably classified into the existing 4 classes. It is plausible that the Kaggle dataset is insufficiently labelled for this type of problem and that this could have affected the overall accuracy of the model.

Overall, there were many different factors in the structure of the dataset and the choices made when building and training the model that could have affected the overall accuracy of each loss function.

5. Conclusion

This project sought to classify white blood cell images to 1 of 4 types. Using a neural network and varying loss functions, our results showed that sparse categorical cross-entropy provides the greatest accuracy in classifying these images. These results suggest that anyone who wishes to classify blood cell images in a similar way should use this loss function over others to more consistently classify white blood cell images correctly.

The neural network using sparse categorical cross-entropy loss had the best accuracy at around 70%. Compared to the existing literature, using a neural network to classify these images was not as effective as other types of models such as Bayesian classifiers, which often resulted in accuracy levels over 90%. However, some researchers who also used neural networks to approach this problem had a similar accuracy of 77% (Theera-Umpon & Dhompongsa, 2007). This suggests that our choice of architecture may be ill-suited for this task, and that choosing a different model could result in a higher accuracy rate.

These findings can be used by others to create more efficient algorithms to correctly classify white blood cells. In the future, this model can be extended to also classify Basophils, which are white blood cells that occur in relatively small concentrations. The methods used in this project can be extended and improved upon and hopefully will allow medical professionals to more quickly and accurately make medical diagnoses based on cell image data.

6. Appendix

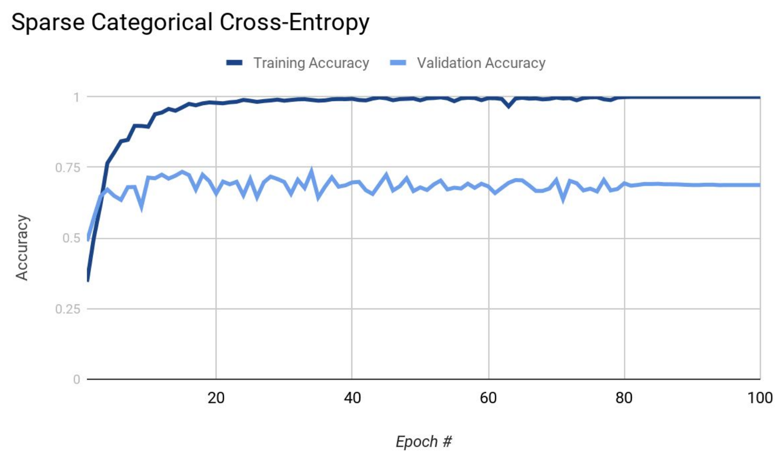

Figure 1

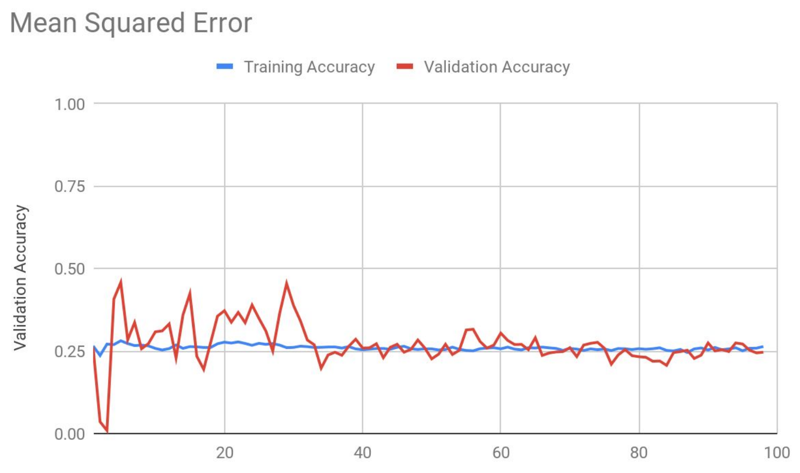

Figure 2

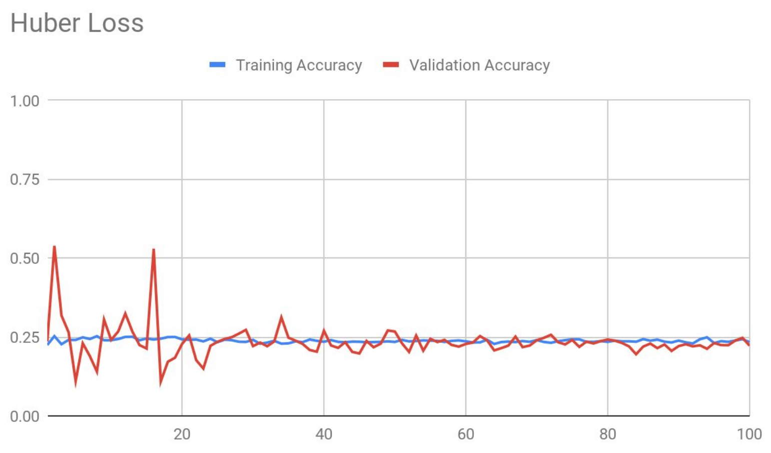

Figure 3

Github Repository: https://github.com/tapphughesn/COMP562FinalProject

References

Elen, Abdullah & Turan, Muhammed. (2019). Classifying White Blood Cells Using Machine Learning Algorithms. Uluslararası Muhendislik Arastirma ve Gelistirme Dergisi. 141-152. https://doi.org/10.29137/umagd.498372.

Hegde, R., Prasad, K., Hebar, H., & Singh, B. (2019). Comparison of traditional image processing and deep learning approaches for classification of white blood cells in peripheral blood smear images. Biocybernetics and Biomedical Engineering, 39(2), 382–392. https://doi.org/10.1016/j.bbe.2019.01.005

Rifkin, R. (2020). Neutrophilia. ScienceDirect; Elsevier B.V. https://www.sciencedirect.com/topics/medicine-and-dentistry/neutrophilia

Sanei, S. & Lee, T. K. M. (2003). Cell Recognition Based on PCA and Bayesian Classification. 4th International Symposium on Independent Component Analysis and Blind Signal Separation, 239-243.

Sarrafzadeh, O., Rabbani, H., Talebi, A. & Yousefi-Banaem, H. (2014). Selection of the best features for leukocytes classification in blood smear microscopic images. Medical Imaging 2014: Digital Pathology, California, 1-8.

Theera-Umpon, N. & Dhompongsa, S. (2007). Morphological Granulometric Features of Nucleus in Automatic Bone Marrow White Blood Cell Classification. IEEE Transactions on Information Technology in Biomedicine, 11(3), 353-359.