White Blood Cell Classification using Convolutional Neural Networks

Gurraj Atwal
College of Engineering & Computer Science
California State University, Sacramento
Sacramento, CA
ga584@csus.edu

Sirish Prabakar
College of Engineering & Computer Science
California State University, Sacramento
Sacramento, CA
sirishprabakar@csus.edu

***Abstract*—Accurate classification of white blood cells is critical for diagnosing blood-based disorders. Traditional methods of classifying white blood cells require experts or expensive machinery. Recent advancements in image classification have made deep learning a viable option for this task. In this project, we explored three different approaches for developing an accurate model. Using a small-scale dataset of only 365 images and heavy use of image augmentation, dropout, and other regularization techniques, our best convolutional layer-based model was able to reach a testing accuracy of 94.3%.**

Keywords—Classification, CNN, VGG-19, Transfer Learning

I. Introduction

White blood cells are an important part of the body's immune system. They are produced in the bone marrow and mature into one of five major types: Neutrophils, Eosinophils, Lymphocytes, Monocytes, and Basophils. Too few or too many of any type indicates a disorder [1].

The process of counting white blood cells can be tedious if done manually and expensive if done automatically. A method that is accurate, quick, and cost-effective when it comes to diagnosing blood-based disorders is invaluable. In general, there are a few steps in counting white blood cells: detection, segmentation, and classification. In this project, we explored the classification of white blood cells using deep learning methods.

We tried three different approaches for finding a deep learning model that takes an image of a white blood cell as input and accurately outputs one of the five categories. First, we implemented a basic multilayer perceptron (MLP) neural network. It performed surprisingly well but was lacking with an accuracy on test data of 85.7%. However, implementing an MLP allowed us to gain a better understanding of how neural networks work. Next, we trained a convolutional neural network (CNN). A CNN is better suited for image classification because it is able to learn low level features such as edges and curves and then build up to more abstract concepts through a series of convolutional layers. After experimenting with a variety of parameters and architectures, our best CNN model was able to reach an accuracy on test data of 94.3%. Finally, in an effort to improve performance even further, we created another CNN model that takes advantage of transfer learning. We used VGG-19, a popular image classification model, as a preliminary feature extractor. Then, we trained our own final layers towards our problem. We expected this model to perform the best but we were only able to achieve an accuracy on test data of 64.3%.

II. Dataset

We used the BCCD (Blood Cell Count and Detection) dataset for this project [2]. The dataset is hosted on Kaggle and contains two directories [3]. The first directory, "dataset-master", consists of 365 images of patient blood samples, a CSV file with labels for each image, and XML files with bounding box annotations for each image. The second directory, "dataset-master2", consists of augmented images of the same blood samples found in the first directory. For this project, we used the first directory of raw images instead of the second directory so that we could gain more experience with image augmentation. The images themselves consist of one or more purple-colored cells-of-interest among other red-colored blood cells.

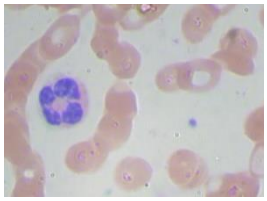

Fig. 1. Neutrophil

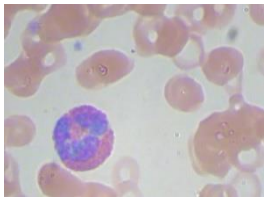

Fig. 2. Eosinophil

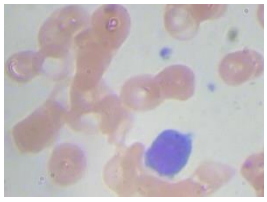

Fig. 3. Lymphocyte

Fig. 4. Monocyte

There are a couple things to make a note of regarding the dataset. Most images contain one white blood cell but there are 15 images containing two white blood cells. Given that handling images with more than one white blood cell would require sophisticated pre-processing, we made the decision to filter out those 15 images. Also, there are only 3 images for the Basophil category so we decided to filter out those 3 images as well. As a result, our dataset consists of only 347 images.

Convolutional neural networks typically require a large number of images per class. Given the small number of images

in our dataset, we relied on data augmentation to create new training data.

III. Related Work

Recently, Jung et al. in [4] proposed a CNN-based architecture coined W-Net for white blood cell image classification. The model consists of three convolutional layers, two fully-connected layers, and a soft-max classifier. They used a private dataset of 6,562 real white blood cell images to achieve an average accuracy for all five classes of 97%. Unfortunately, our dataset is much smaller. However, we used this paper as a reference and attempted to achieve similar results.

There are a handful of Kaggle kernels written for the dataset we are using [3]. However, most are using the "dataset-master2" directory of pre-augmented images. The top kernel by user "placidpanda" was used as a reference. The kernel described the effects of various parameters and architecture decisions which helped us hit the ground running. It reached a 97% accuracy on the validation set but only an 84% accuracy on the testing set. Our hypothesis for why the testing set accuracy was so low is that the validation set had high correlation with the training set. The testing and validation sets were augmented images from the same common pool of raw images. We expected to outperform this kernel by using the "dataset-master" directory instead and performing our own image augmentation after splitting for training, validation, and test.

IV. Implementation and Results

We created three different models that classify images of white blood cells into four different categories: Neutrophil, Eosinophil, Lymphocyte, or Monocyte. We implemented a multilayer perceptron (MLP) neural network and two convolutional neural networks (CNNs). The first CNN was built from scratch and the second one leveraged transfer learning with the popular VGG-19 pre-trained image classification model. However, before training any models, we performed data exploration and augmentation.

A. *Data exploration and augmentation*

As mentioned earlier, through data exploration, we found that 15 images in our dataset contained two white blood cells instead of just one. Also, there were only 3 images for the Basophil category. The first action we took was to filter out those 18 images from the dataset.

Each image was consistently 640 by 480 pixels. For this project, we had access to a GeForce RTX 2070 SUPER which only has 8 GB of GDDR6 memory. To avoid exceeding the memory available to us, we resized each image to 160 by 120 pixels. The resized images also helped decrease training time without significantly reducing visual integrity or model performance.

Finally, given that our dataset has so few images, we took advantage of the Keras ImageDataGenerator to perform automatic image augmentation [5]. The generator generates "batches of tensor image data with real-time augmentation." As a result, while training, our models received batches of randomly augmented images. Augmentation was done based on a set of parameters we chose.

TABLE I. ImageDataGenerator Parameters

Parameter	Value
rotation_range	30
horizonal_flip	True
vertical_flip	True
width_shift_range	0.2
height_shift_range	0.2
shear_range	0.1
zoom_range	0.1
fill_mode	constant

We applied rotations and flips generously because neither negatively impacts the recognizability of a circular white blood cell. Other augmentations like shifts and zooms were modest to avoid destroying our data.

B. *Normalization*

An image has three channels for red, green, and blue. Values in each channel range between 0 and 255. The only normalization we performed was we scaled each value so that it lies between 0 and 1. The purpose of doing this was to allow gradient descent to more efficiently learn underlying patterns among the dataset.

C. *Train-validation-test stratified split*

Of the 347 images in our database, we reserved 20% for testing and the remaining 80% for training. Then, of the 80% for training, we took another 20% and reserved it for validation. We applied the augmentation strategies mentioned earlier to only the training data. The testing and validation data were left largely unchanged outside of resizing and normalization. The same training data was used to train all models. The validation data was used to evaluate loss on models at the end of each epoch. Finally, the testing data was used after training to evaluate model performance on brand new, never-before-seen data.

D. *Class imbalance*

We noticed that certain white blood cell types (Neutrophil and Eosinophil) appeared more often in the dataset compared to others (Lymphocyte and Monocyte). To address this, we randomly oversampled minority classes until their representation in the dataset matched that of the majority class. We only oversampled the training set. The validation and testing set was left unchanged in order to more closely reflect reality.

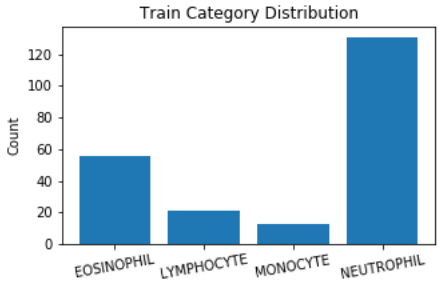

Fig. 5. Train Category Distribution

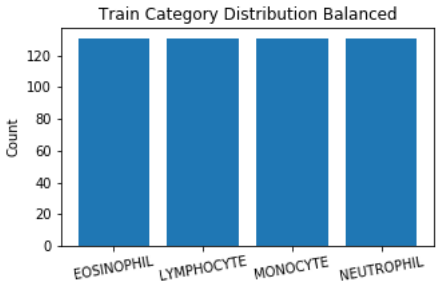

Fig. 6. Train Category Distribution Balanced

E. MLP

The first model we developed was a multilayer perceptron (MLP) neural network. Typical neural networks are able to learn "any mapping function and have been proven to be a universal approximation algorithm" [6]. The best model we found consisted of four dense layers each using the Rectified Linear Unit (ReLU) activation function.

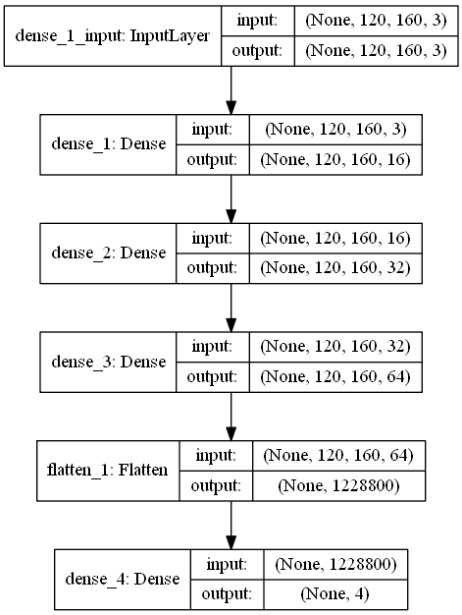

Fig. 7. MLP Design

F. MLP Results

The MLP model was trained for 500 epochs with model checkpointing. To our surprise, the model performed quite well with a classification accuracy of 85.7%. The downside of an MLP approach is that the model is not able to learn the relationships between pixels. We also found that the model was starting to overfit on training data after around 200 epochs. We used what we learned for our next model.

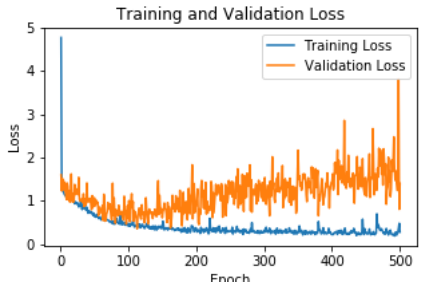

Fig. 8. MLP Loss

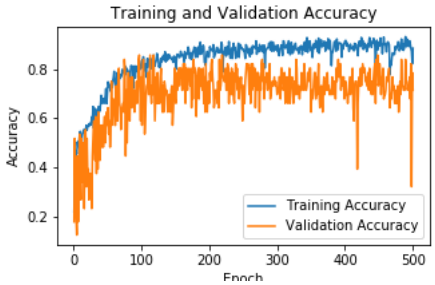

Fig. 9. MLP Accuracy

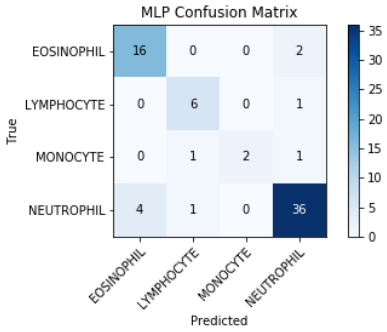

Fig. 10. MLP Confusion Matrix

G. CNN

Convolutional neural networks (CNNs) are able to "successfully capture spatial and temporal dependencies in an image through the application of relevant filters" [7]. CNNs employ convolutional layers which use kernels/filters of various sizes to scan images and perform convolution operations. Convolution operations extract high-level features such as edges and curves to fully understand input images. These characteristics make CNNs ideal for our problem.

The best CNN model we developed consisted of three convolutional layers each followed by batch normalization, max pooling, and dropout layers. A small training dataset paired with a complex model typically leads to overfitting. We saw this happen with our MLP model so we took preventative measures. Dropout layers with a high probability were key. They are responsible for randomly setting a fraction of input units to 0 at each update which helps prevent overfitting [8].

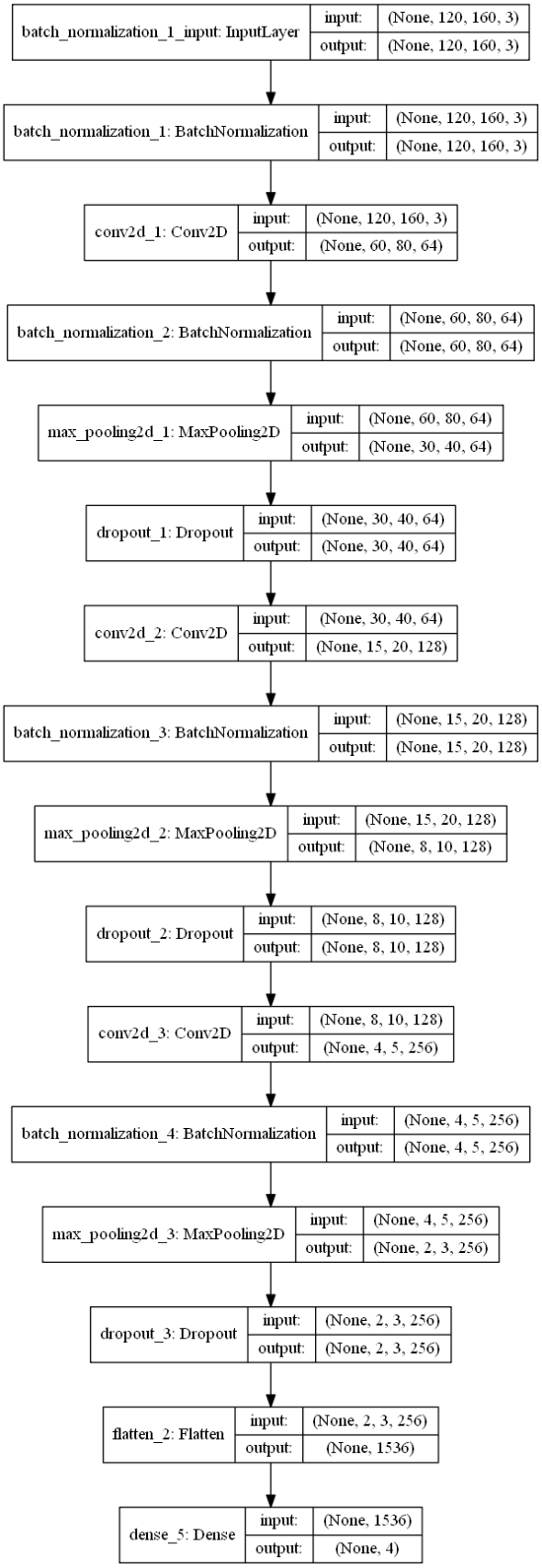

Fig. 11. CNN Design

H. CNN Results

As expected, batch normalization and dropout layers successfully helped avoid overfitting. After 500 epochs, our best model produced a classification accuracy on test data of 94.3%.

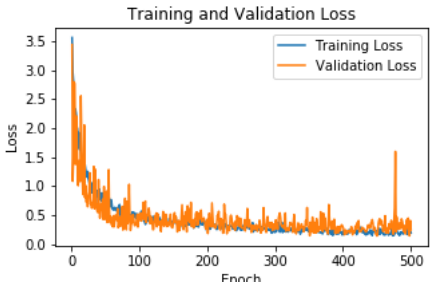

Fig. 12. CNN Loss

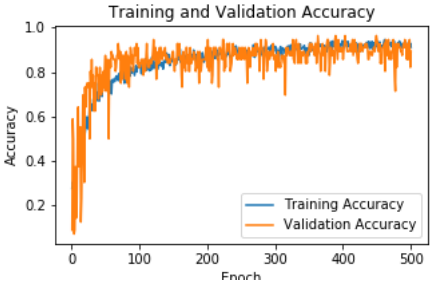

Fig. 13. CNN Accuracy

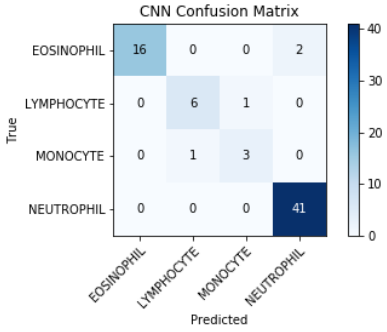

Fig. 14. CNN Confusion Matrix

I. Transfer Learning

Transfer learning is a popular approach in deep learning where a model developed for a task is reused as the starting point for a model for a second task [9]. The pre-trained model is used as a feature extractor for the new task even though it was never trained for said task. In our case, we used VGG-19 – a model introduced by Simonyan and Zisserman for image classification [10]. VGG-19 was trained on the ImageNet Challenge 2014 dataset making it ideal for general classification.

Our approach was to import the pre-trained model, remove the final layer, freeze all layers, and add our own dense layers. Freezing is important so that pre-trained weights are kept constant. We added three dense layers, a dropout layer, and a final soft-max layer. Then, we trained our layers using the same process we used for the last two models.

J. *Transfer Learning Results*

We were expecting the transfer learning model to outperform the other two models. In reality, our last model performed worse than the MLP model and received a testing accuracy of 64.3%. Adding or remove dense layers did not seem to significantly affect results. The model did not seem to be heavily overfitting either. One hypothesis for why the model performed so poorly is that the ImageNet dataset, while general, was too different from white blood cell images. It is also possible that our model required more complexity. Even after training for 1000 epochs instead of 500 epochs, the results were the same.

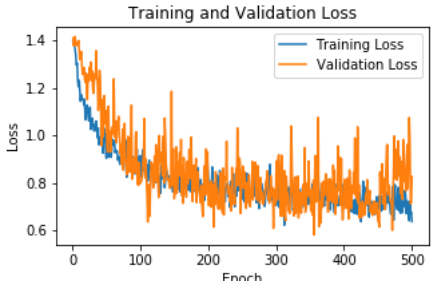

Fig. 15. Transfer Learning Loss

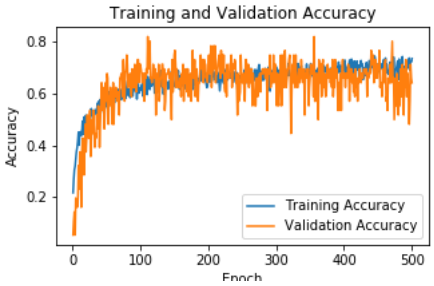

Fig. 16. Transfer Learning Accuracy

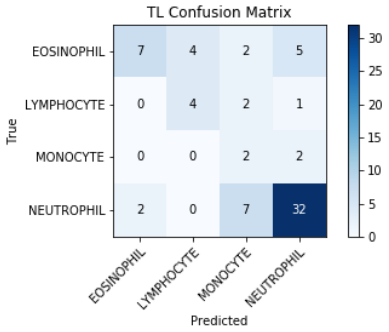

Fig. 17. Transfer Learning Confusion Matrix

V. Conclusion

In this project, we created three different models in search of a model that accurately classifies white blood cells. While the transfer learning results were underwhelming, the second model we trained, the CNN, performed well with a testing accuracy of 94.3%. The results are even more impressive considering the small-scale dataset of only 365 images of which 18 were filtered out. The combination of heavy image augmentation, batch normalization, and dropout layers were key in avoiding overfitting so that our accuracy on test data was as high as our accuracy on training data.

TABLE II. Results

Model	Accuracy	F1
MLP	85.7%	85.5%
CNN	94.3%	94.2%
Transfer Learning	64.3%	66.3%

VI. Future Work

There are a number of ideas we were not able to implement due to logistics or time constraints.

- First, we are interested in how much a larger dataset would have improved our results. CNNs, like other neural network models, perform best when trained on thousands of images instead of just hundreds in our case.
- Our transfer learning results were lackluster. Future work could explore improving results in this area, potentially using other pre-trained models. Models related to medical images may help performance.
- Instead of reserving 20% of our training data for validation, we could have used k-fold cross-validation.
- In this project, we tackled only classification. Future work could tackle detection and determining bounding boxes for white blood cells. The bounding boxes could be used to crop images and help train and improve classification even further.

References

[1] Territo, M. (2019). Overview of White Blood Cell Disorders. [online] Merck Manual Consumer Version. Available at: https://www.merckmanuals.com/home/blood-disorders/white-blood-celldisorders/overview-of-white-blood-cell-disorders.

[2] GitHub. (2019). Shenggan/BCCD_Dataset. [online] Available at: https://github.com/Shenggan/BCCD_Dataset.

[3] Mooney, P. (2019). Blood Cell Images. [online] Kaggle. Available at: https://www.kaggle.com/paultimothymooney/blood-cells.

[4] Jung, C., Abuhamad, M., Alikhanov, J., Mohaisen, A., Han, K. and Nyang, D. (2019). W-Net: A CNN-based Architecture for White Blood Cells Image Classification. Available at: https://arxiv.org/pdf/1910.01091.pdf.

[5] Keras.io. (2019). Image Preprocessing - Keras Documentation. [online] Available at: https://keras.io/preprocessing/image/.

[6] Brownlee, J. (2019). Crash Course On Multi-Layer Perceptron Neural Networks. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/neural-networks-crash-course/.

[7] Medium. (2019). A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way. [online] Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[8] Keras.io. (2019). Core Layers - Keras Documentation. [online] Available at: https://keras.io/layers/core/.

[9] Brownlee, J. (2019). A Gentle Introduction to Transfer Learning for Deep Learning. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/transfer-learning-for-deep-learning/.

[10] Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks For Large-Scale Image Recognition. [online] Arxiv.org. Available at: https://arxiv.org/pdf/1409.1556.pdf