Leukocyte classification based on neural network

1st Shilong Yin
Columbia University
Biomedical Engineering
UNI: sy2792
sy2792@columbia.edu

2nd Diwei Xiong
Columbia University
Biomedical Engineering
UNI: dx2183
dx2183@columbia.edu

***Abstract*—White blood cells (also called leukocytes) play an indispensable role in protecting the body from infections of disease and foreign invaders. In modern blood test, the number of White Blood Cells (WBC) will be included with specification to each cell type, which offers us a glimpse into our body immune system and any potential risk we are facing. Thanks to the development of microscopy, we are able to capture the actual image of different kind of white blood cell. In order to further understand the mechanism of our body immune system, we need to look deeper into the morphology and distribution of white cells based on the information provided by images [1].**

Therefore, after investigating various kinds of model and method, we used convolutional neural network (CNN) model to design an automated model with the ability of automatically categorizing leukocytes so that physicians and scientists may have more time on medical care and research. As a result, we have achieved the test accuracy score of 0.8832, which is fairly good in predicting the specific kind of leukocytes from four classes.

Github repository: Leukocytes Classification

***Index Terms*—leukocytes, classification, deep learning, convolutional neural network.**

I. Introduction

In this project, we are aimed to categorize four major kinds of leukocytes. The first one is Neutrophil, which composes of almost 50% of all leukocytes. Neutrophil is also the first cell that foreign invaders will encounter after them entering the human body; it is expected to release chemical signal as a messenger to inform other leukocytes. Morphologically, the neutrophil has nuclei and the nuclei are segmented by chromatin connected by filaments. Then comes with the Eosinophils, which only composes of less than 1% of all leukocytes but are important in fighting with invaders especially the parasites. They are usually aggregated in intestinal tract and other digestive systems, and may cause some allergic symptoms when high in concentration. Eosinophil granule are usually orange, and its nuclei is shorter and less segmented than that of neutrophil. The third one is Monocytes, which composes of 5% of leukocytes. They are known as scavenger in human body, hovering around the body and cleaning dead tissues or cells. The chromatin of Monocytes is not as dense as that of neutrophil, and they are large in size, but their size and shapes vary a lot, making them hard to discern. The last, but also maybe the best-known leukocyte is lymphocyte, which can be divided into T lymphocyte and B lymphocyte. The function of T cell is to kill the foreign invaders through the process of phagocytosis; B lymphocytes is widely used in producing vaccine, because it is able to produce antibodies to target specific type of bacterial or virus, thus responsible for humoral immunity. Lymphocytes are usually small in size, and has nuclei with very dense and smooth chromatin and small rim of light blue cytoplasm. Fig.1 are typical examples of cell images from these four types [1].

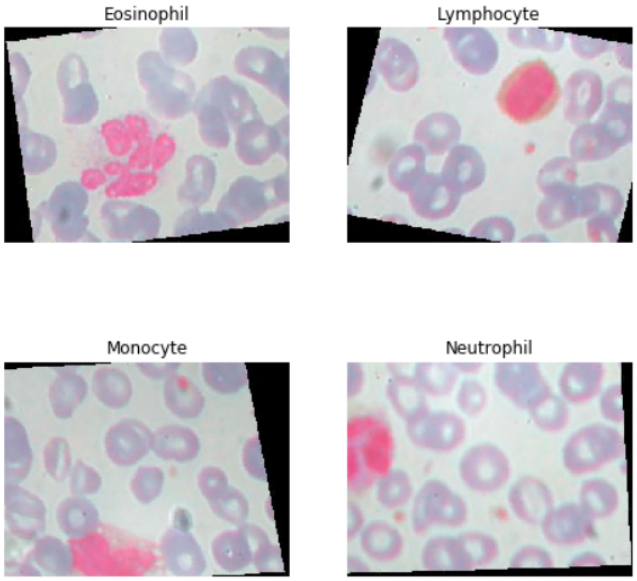

Fig. 1. Exmaples of cell images [2], [3]

Therefore, based on the differences of morphology of leukocytes, we believe that the CNN model is able to recognize different types of leukocytes based on their morphological features in the image.

II. Methods

A. Data overview and preprocess

The data we use is from an openly available Kaggle dataset about blood cell images. There are about 12500 pictures containing four kinds of leukocytes and the size of each image is 320*240 with standard RGB channel. The original dataset is divided into 4 folders, each stands for a different cell type, for both train and test data [2].

By first checking the image data, we found that there are some duplicate pictures in these files (different files from

different classes but with the same name), so we decided to merge all images only into 2 folders: train and test, and delete images with duplicate names. The reason we do this is that we would like to split the whole training dataset into 80% training and 20% validation data and use image generator to output image batches, rather than read whole dataset into memory, which is adopted by original kernel [11].

Besides, since we no longer split images into 4 folders to stand for their class, we created two .csv files to document the image name and its numerical labels for reference.

B. Image generator

As mentioned above, the data set contains 12500 images of size 320*240. If we directly load them into the computer memory without resize, it is very likely that there will be a memory exhaust problem. But resize, as a compromise alternative, also impairs the amount of information an image can provide. Therefore, we decide to use an image generator that read images directly from hard drive during the training process so that we can use full size images without causing memory exhausting.

After reading the image, the generator will also do some pre-processing to improve the efficiency and robustness of the model. First, it will normalize the image and divide it by 255, which allows the network to more quickly learn the optimal parameters for each input node. Then it will augment the image from training dataset by rotating or flipping them with a random parameter. This has the same effect as providing more training data and it will provide better robustness and generalization ability to our model [7].

In the result evaluation part, we will try different batch size and resize scale to test whether it will influence the final performance of the network.

C. Model selection

To construct deep neural network, we consider using conv2D, dense, max-pooling, dropout, flatten and batch normalization, with ReLU as our activation function and Adam optimizer due to its superior performance. We start with applying pretrained model because it has already been trained on a large amount of data so that it can serve as a perfect base for our model. Then we pass the output of pretrained model into a 2-dimension convolution layer. Then, we use batch normalization to reduce the covariance shift. We also insert a max pooling layer to reduce the spatial dimension. After that, a dropout layer is applied to output so that it will lose certain percent of units. And we flatten the result and apply a dense layer with softmax function to pick up the result with maximum likelihood. The structure of our model is shown in Fig.2 and we will introduce some of methods we use [11].

- Pretrained Model: Pretrained model is a model that has been trained on some kinds of data. Although it may not perform well on a specific task, it can be a good corner stone for a model that is aimed for a similar task. In our case, we want to classify the leukocytes, which is a task that can be categorized into general image classification. Therefore, using a model pre-trained on millions of images as a base will be a better choice rather than training a new model from scratches [5].
  Fig.3 is a list of image classification model provided by Keras. In this case, we use InceptionResNetV2 as our pretrained model because it has very good validation accuracy and the deepest net structure, which means that it is highly optimized for general use [9], [10].

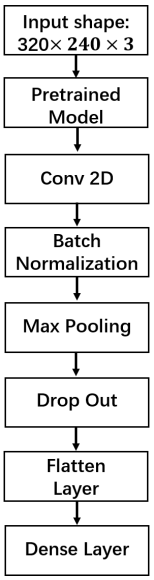

Fig. 2. Model structure

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
ResNeXt50	96 MB	0.777	0.938	25,097,128	-
ResNeXt101	170 MB	0.787	0.943	44,315,560	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

Fig. 3. Comparison of Keras image classification model [9]

- Convolutional Network: Convolutional network is used to extract features from 2-D images. The basic principle behind it is to apply filters on the matrix to achieve a certain image processing like blur or edge detection. This process is called hierarchical feature learning, which is similar to how brain identifies the objects. Thanks to the existing libraries, we are not required to produce filters by ourselves but only to decide the number and size of

filters. After filtering, the extracted features will later be used to calculate the weights for each neuron [6].

- Batch Normalization: Generally, what normalization do is to adjust and scale the input from a much larger range to [0,1]. The reason why we need this normalization layer is that by normalizing the inputs, the amount by which the hidden unit values shift around will reduce, and this means less time for the model to reach the optimal parameter.
  Batch normalization can also help the model to increase resistance for overfitting because it has a slight regularization effect [7].
- Max Pooling: Max pooling is a sample-based discretization process. It helps to down sample the input representation and at the same time reduce the dimensionality. By doing so, max pooling layer not only lower the computational cost but also increase the resistance of model against overfitting. Besides, max pooling also provides basic translation invariance to the internal representation, which helps to improve the robustness of the network [8].

D. Regularization and Optimization

- Adaptive learning rate: Learning rate is one of the most important hyperparameters for the network training since it has great influence on the model performance. It is hard to tune a fixed learning rate since when it is too small, it will take a long time for model parameters to get to the optimal values, while when it is too large, the model parameters may be stuck at a local minimum. Therefore, it is advised to use adaptive learning rate during training process. To be specific, the loss will be monitored during training and if the model performance doesnt improve for some epochs, the learning rate will be lowered [4].
- Early stop: If a model is trained on a small set of data, it is often the case that the model will overfit and perform poorly on test data, because the model will adopt some noise features from training data and apply them on prediction. Overfitting is represented as a huge gap between training loss and validation loss. Early stop is a strategy that stops model training when the monitored validation loss or accuracy doesnt improve anymore. It serves as a ideal method against overfitting [4].
- Drop out: Drop out is another regularizing method that randomly sets a fraction of input units to 0 in order to prevent overfitting. The reason why this works is that it helps the model to reduce the dependencies on a certain unit. Generally, the drop out layer is used after the max pooling layer as well as on the fully connected layer [4].

III. Result

Based on this model structure above, we build a neural network model that achieves at best 88.32% test accuracy on this four-class classification task. The loss and accuracy curve are shown in Fig.4. The training process is done with a NVIDIA GTX 1070 within one hour.

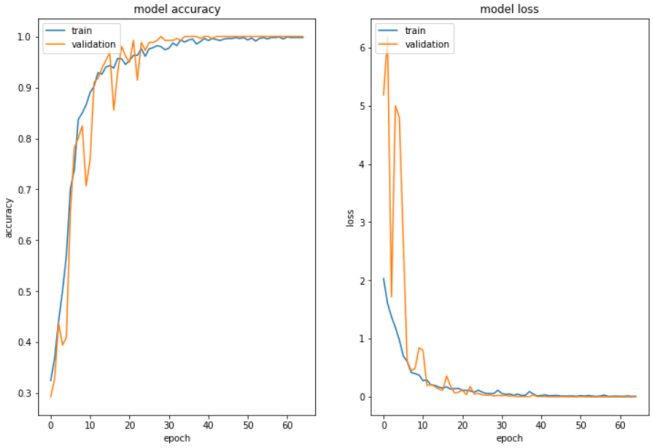

Fig. 4. Loss and accuracy curve of our model

It is noticeable in the results that at the end of training, the validation accuracy is almost 1. This is due to the fact that the training and validation dataset is augmented from the same original dataset (The original dataset containing 410 pictures is split into training and test data and augmented to 12500 images) and they may share similar features. So in this case, validation loss is not a very good criterion for our model performance.

As comparison, we lists two kernels on the same dataset with their performance below in Fig.5 and Fig.6.

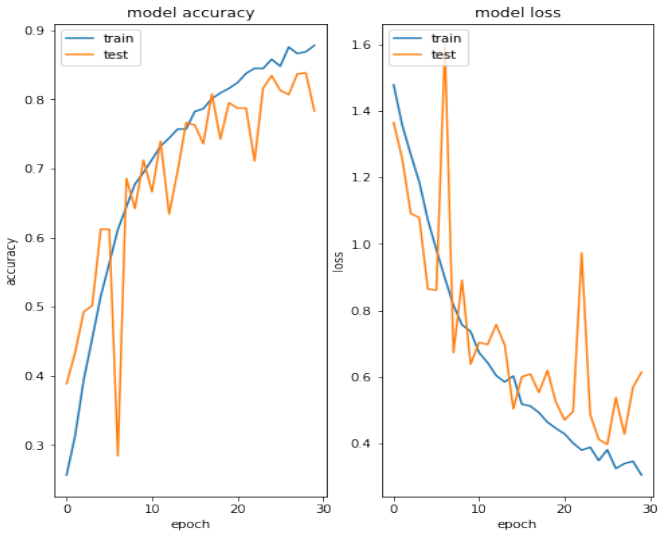

Fig. 5. Loss and accuracy curve of Paul Mooney's model [11]

Paul Mooneys kernel reaches the test accuracy of 78.32%. It resizes the image to (60,80,3) and the model doesnt include a pretrained model layer [11].

nh4cls kernel reaches test accuracy of 83.76%. Compared to Mooneys kernel, it increases the depth of model with more layers and adopts image normalization before training [12].

In my opinions, our model surpasses their model performance mainly because we use a pretrained model and the

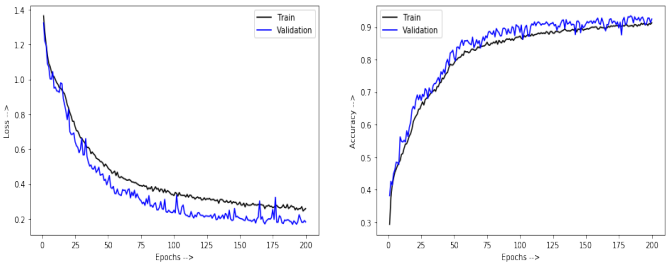

Fig. 6. Loss and accuracy curve of nh4cl's model [12]

full-size images to train the network.

IV. Discussion

Batch size decides upon the quality of every parameter update. The larger the batch size is, the better an update will be in improving model performance. However, considering the memory constraint of GPU, there will be a limit on number of images per batch. Sometimes, we have to resize pictures so the model can be trained on a larger batch size. But it also reduces the information provided by each picture [4].

To balance between these two hyperparameters, we try using different image size and batch size to see its influence on the model performance. The result is shown below in table.1.

TABLE I
Model performance with image size and batch size

Image size	Batch size	Test accuracy
(240,320,3)	20	88.32%
(150,200,3)	50	83.84%
(90,120,3)	100	77.18%

Based on the result, we can see that test accuracy is more concerned with image size rather than batch size. Therefore, in our final model, we use the full-size images as our input. Besides, we also plot the confusion matrix to see the model performance on predicting images from a certain class. The result is shown in Fig.7.

It is noticeable that many eosinophils and monocytes are predicted as neutrophil. It may be due to the fact that in the original dataset, the neutrophil takes the most part (about 50%) and when the classifier is not so sure about the prediction, it will predict its label as neutrophil. Another interesting fact is that monocyte is the class on which the model performs worst. As mentioned in the introduction part, monocyte is the cell type being most difficult to discern among all leukocytes, and this is supported by our prediction result [1].

V. Conclusion

In this project, we designed a leukocyte classifier based on neural network that is able to categorize among neutrophil, eosinophil, monocyte and lymphocyte micrographs. And our final model reaches a test accuracy of 88.32%, which is much better than the baseline: 25%. Compared with other kernels, our model also performs better mostly because we

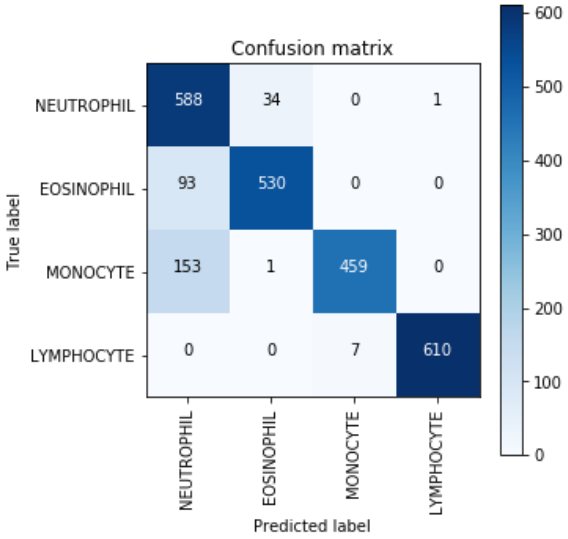

Fig. 7. Prediction confusion matrix

adopted a more complex structure and full-size images as our input. Based on the positive outcome achieved, the model may serve as an auto classifier for different leukocytes so that the physicians and scientists will have more time on research and medical care. In the future, we proposed to test the model on a huge amount of data and improved it into a classifier that can be applied on images containing more than one kind of white blood cells.

Acknowledgement

This project use data from Blood Cell Images. These cell images are augmented from BCCD Dataset, which contains 410 white blood cell images with annotation. The augmented dataset increases the number of samples as well as balances the ratio of images from different cell types. The final data set contains about 12500 images for training and test use.

Our model is based on kernels of Paul Mooney and nh4cl. We show our gratitude to them as well as all others that help us in our research.

References

[1] Lynne Eldridge, Types of Function of White Blood Cells [Online], verywellhealth, March, 2019.
[2] Paul Mooney, Blood Cells Images [Online], Kaggle dataset.
[3] Shenggan, BCCD Dataset [Online], Github.
[4] Ian Goodfellow, Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016.
[5] Sulaiman Vesal, Quora Question: What is meant be pre-trained model in CNN? Are they already trained on that particular classes? [Online], April, 2017.
[6] cs231n, Convolutional Neural Networks (CNNs / ConvNets) [Online].
[7] Jeremy Jordam, Normalizing your data (specifically, input and batch normalization) [Online], January, 2018.
[8] Unknown, Max-pooling/Pooling [Online], Computer Science Wiki.
[9] Unknown, Applications [Online], Keras Documentation.
[10] Alex Alemi, Improving Inception and Image Classification in Tensor-Flow [Online], Google AI Blog, Aug 31, 2016.
[11] Paul Mooney, Identify Blood Cell Subtypes From Images [Online], Kaggle kernels.
[12] nh4cl, Deep Learning from Scratch + Insights [Online], Kaggle kernels.