**Classification of Leukemia Blasts**

Nick Samsel

MSDS686\_X40\_Deeplearning

Kaggle Project

6/30/24

**Dataset Overview**

For this project I chose to build a CNN to classify images of cells as either normal or cancerous leukemia blasts. This dataset contains 15,135 images of cells from 118 different subjects, of which 49 patients did not have cancer and the remaining 69 do. These images are split across three different folders: validation\_data, training\_data, and testing\_data. Within the training data folder there are three folders labeled fold\_1, fold\_2, and fold\_3. These folders are simply subsets of patients with no overlap between each fold. Each of the fold folders contain an all and hem folder which denotes whether the images contained are cancerous (all) or normal (hem). The training dataset contains a total of 10,661 cell images from 73 subjects, the validation data contains a total of 1867 cell images from 28 subjects, and the test dataset contains 2586 cell images from 17 subjects. Not that the data in the testing\_data folder does not contain any labels, according to the datasets CNMC\_readme file the only way to validate final results with the testing data is to submit the code to a now closed competition. Therefor I will be evaluating my CNN via the validation data included in the Kaggle dataset and will discard the testing data as there is no way to evaluate any classification.

**Data Preprocessing and Modeling**

**Preprocessing**

Per the dataset’s read me file, much of the preprocessing of the images was done by the researchers themselves. This was done via a proprietary stain normalization method which the researchers developed. The only additional preprocessing done was via To have more data to train my model I chose to utilize data augmentation by utilizing the scalar rotation, shifts, shear transformations, zooms and fill mode.

**Modeling**

I first began modeling by utilizing the Keras applications pre-trained InceptionResNetV2 as a base model. After this I added batch normalization, an initial dense layer which also contains regularization, five dropout out layers and four more dense layers. I compiled this initial model via adamax with a specified learning rate of 0.001. This model gave me an overall test accuracy of 0.87875 and a loss value of .63450 which is pretty good for an initial attempt. I decided to switch to the pre-trained EfficientNetB3 pre-trained model to use as a base to see if that raises the validation accuracy. I chose EfficientNetB3 over the other pretrained models because of its relatively high accuracy when compared to other pretrained Keras models. After completing the test it seems utilizing EfficientNetB3 did improve the accuracy of the model by .01 and reduced loss by .1 which is excellent. I would have liked to try some of the larger more complex models like ConvNeXtXLarge but my GPU at home (4070ti) did not have enough memory (12GB GDDR6X) to actually run these models.

**Results**

Ultimately I would call this project a success, though I wish I could have developed a model with accuracy closer to what others have been able to achieve with this dataset. First, my InceptionResNetV2 based model performed well as can be seen in the screen shots below:

A graph with blue dots

Description automatically generated

A graph showing the results of training

Description automatically generated

A graph with numbers and a number of blue squares

Description automatically generated with medium confidence

These results show a pretty decent model, though there is some overfitting. The second EfficientNetB3 model has very similar results though with slightly better accuracy and loss metrics. Below are the screenshots showing that model’s results:

A graph of training loss

Description automatically generated

A graph showing the performance of training

Description automatically generated

A screenshot of a graph

Description automatically generated

Ultimately these results are promising though could use more fine tuning to achieve even better results. Below is a link to the Github repo where this code is published for public consumption.

Github Link: [NickSamsel/CNMC\_2019\_Classification: This python notebook contains CNNs which aim to classify cancer cells from the CNMC 2019 publicly available dataset (github.com)](https://github.com/NickSamsel/CNMC_2019_Classification)

**Sources**

Mourya, S., Kant, S., Kumar, P., Gupta, A., & Gupta, R. (2019). ALL Challenge dataset of ISBI 2019 (C-NMC 2019) (Version 1) [Data set]. The Cancer Imaging Archive. <https://doi.org/10.7937/tcia.2019.dc64i46r>

Ibrahim, A. W. (2023, February 19). *Leukemia cancer classification: 95.75 %*. Kaggle. https://www.kaggle.com/code/abdallahwagih/leukemia-cancer-classification-95-75