**Image Classification for Leukemia Cells**

**Introduction and Data Exploration**

Imagine you or a loved one, possibly a child in your life, are experiencing new symptoms of fatigue, bruising easily, rashes, frequent illness, and perhaps fever with no other signs of illness. As you visit a physician’s office, blood labs are drawn and sent off to a laboratory for analysis. During that time, the blood sample is examined to determine if it is possible that it indicates you or a loved one have a disease that could take your life in months if not caught quickly. Knowing it’s notoriously difficult to identify the diseased blood samples, would you want every possible medical or technological advancement that was possible exhausted to help examine the blood sample?

Acute lymphoblastic leukemia, or ALL, is a blood cancer that develops from lymphocytes, a type of white blood cell, and starts in the bone marrow, but can quickly move to the blood or other organs. “Acute” within the name of the condition indicates that the disease can progress quickly without treatment ([American Cancer Society](https://www.cancer.org/cancer/types/acute-lymphocytic-leukemia/about/what-is-all.html#:~:text=Acute%20lymphocytic%20leukemia%20(ALL)%20is,type%20of%20white%20blood%20cell.)). Further, ALL is the most common childhood cancer, making up approximately 25% of childhood cancer cases ([NLM](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4250840/#:~:text=Acute%20lymphoblastic%20leukemia%20(ALL)%20is%20the%20most,ALL%20over%20the%20past%2050%20years%2C%20which)). The disease, however, can also occur in adults. One of the challenges presented in diagnosing the disease is due to the similarity of normal and ALL cells. Drops from blood samples are smeared on slides and stained to make them easier to differentiate blood cell types. When examined under the microscope, ALL cells and normal cells can look very similar and vary from patient to patient. This typically requires the review of a trained expert pathologist or oncologist.

Because machine learning models have been used in the medical field to aid in screening and decision-making for physicians ([NLM](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9020906/)), one use case for machine learning is screening blood smear samples to differentiate between normal and ALL cells, especially due to the challenging nature of appropriately classifying them. The IEEE International Symposium on Biomedical Imaging proposed such a challenge in their [2019 conference challenge](https://biomedicalimaging.org/2019/challenges/). The goal of the challenge was to classify cells as normal or malignant, specifically for B-Cell ALL. The organization provided a dataset with classification labels (normal or ALL), as identified by an oncologist. Here we examine several approaches to classification of this type of data using convolutional neural nets, or CNNs. These deep learning models are commonly used for image recognition problems because of their ability to learn from pixels without the need to feature engineering, the ability to extract hierarchical information from images (capturing edges and horizontal and vertical lines first before moving on to finer details), and because the convolutional layers have fewer features than fully connected layers, it allows the large amount of information to be processed and learned in an efficient manner. Convolutional layers within the network provide the ability to better learn the image by using filters (or kernels) to analyze a portion of the image and create an activation map for its features.

We obtained the C-NMC dataset from [Kaggle](https://www.kaggle.com/datasets/andrewmvd/leukemia-classification). The data is structured in 3 folders - training data, validation data, and testing data. Within the training data folder, there are three folds of data, each containing multiple blood smear images from multiple patients. Between the folds, there is no patient overlap, though. In the testing and validation folders are additional files from patients not seen in the training set. As mentioned above, the morphological characteristics between blood cells vary from subject to subject. Because of this, the structure used to distribute the images will allow the trained model to view images from subjects who may have different morphological characteristics, providing a real-world test of its ability to generalize, or perform well on unseen data. In total, between the training and validation sets we will use, the dataset provides 13,247 images of cells from 90 subjects. Initially, we planned to use training, validation, and test sets provided, however as we further explored the data and folder structures created by the challenge authors as well as documentation on the original challenge, we confirmed there are no ground truth labels to evaluate the model performance for the test set. Because of that, we opted to use what was designated as a validation set as our test set and to split the training data into train and validation sets. Between the remaining images in the training and validation folders, there are 13,084 images.

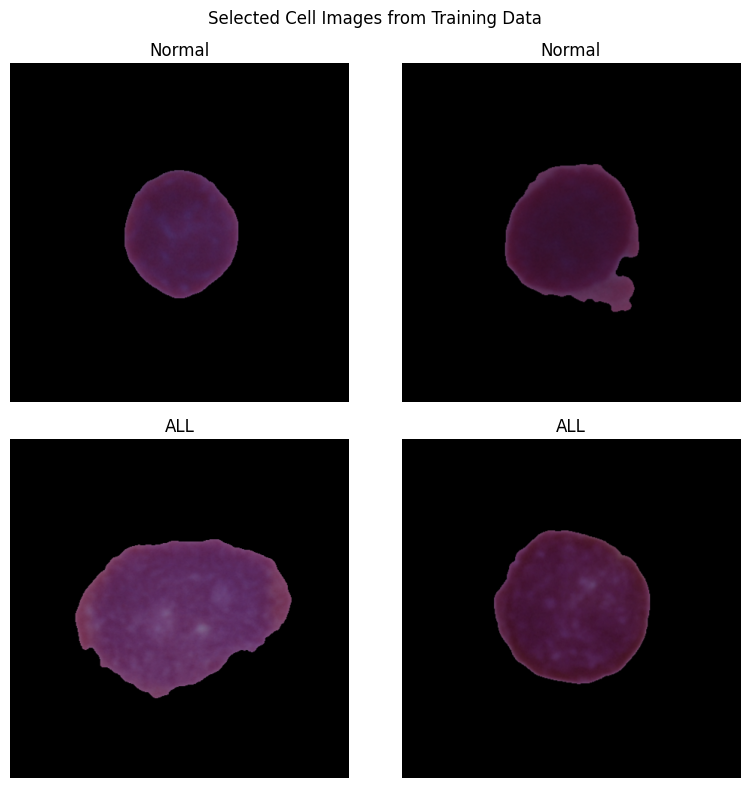
Despite the image labeling done by the authors of the challenge that indicates the patient number, we decided to combine all images from all folds from the training data into a single pandas DataFrame, a structure which would allow us to iterate over the image file paths in a succinct manner. We used the folder labeling of “all” or “hem” (hem denotes normal cells) for each image to assign its ground truth label within the DataFrame.

*Table 1. Structure of DataFrame of Cell Images*

|  |  |
| --- | --- |
| **Column Name** | **Example Values** |
| ‘filepath' | /content/drive/My Drive/C-NMC\_Leukemia/trainin.. |
| ‘label' | all |

Of note, there are no other details regarding the lymphocyte images, such as patient demographics (age, gender, socioeconomic status), test results, symptoms, etc. that could serve as other predictors. The only piece of information we have are the images themselves.

Once we structured our training/validation data and test data into our code notebook, we created a visualization of randomly selected images from the training set, seen below in Figure 1. They appear on a black background with a red-purple appearance (likely from Wright’s stain, a common stain used in blood smears that causes lymphocytes to appear this color). Reviewing these images and the apparent differences between the normal and ALL lymphocytes, we can understand why it is notoriously difficult to distinguish between the two classes. The margins on the top right normal cell are irregular, somewhat more so even than the bottom left ALL cell. But then, to the untrained eye, the top left normal cell and bottom right normal cell appear very similar.



*Figure 1. Randomly selected images from training data with ground truth labels.*

*The images highlight the difficulty of distinguishing between the two classes.*

We also observed the relative size difference in cells and that while placement of the cells within images is centered, that much black space remains unused and does not add to our model. As our model will examine each pixel within the images to distinguish between classes, we can create some computational efficiency by cropping the images.

Further reviewing the training and test set, we observed that our training data is imbalanced There are 7,272 images that are labeled as “ALL” and 3,389 labeled as “hem” (or normal) in the training data and in our test set, 1,219 cells labeled as “ALL” and 648 labeled as “hem”. Both sets of data have roughly a 1:2 ratio between healthy and cancerous cells. Figure 2 below visually displays the underrepresentation of normal cells within the dataset and the challenge this may represent to our model in accurately identifying normal cells. One way that we can improve this disparity is to augment our data with the current images by rotating, flipping, or otherwise manipulating the images to increase exposure of the model to normal images.

A graph of a number of cells

Description automatically generated

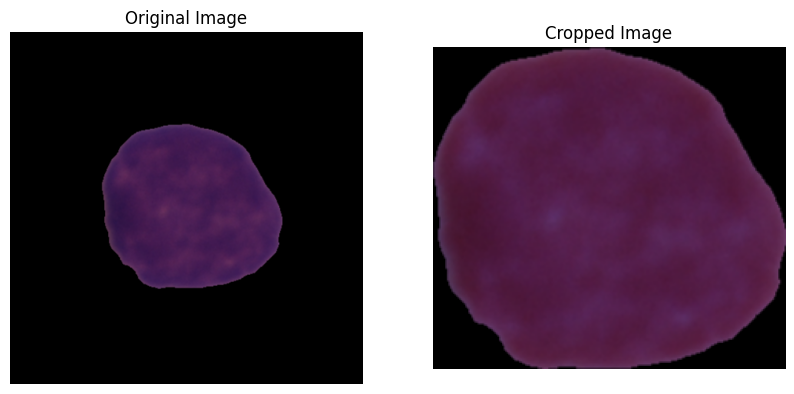
A graph of a number of cells

Description automatically generated

*Figure 2. Distribution of normal (hem) and ALL cells within data.*

**Image Processing and Data Cleaning**

We noticed significant blank space in our original images and sought to enhance our dataset quality, and improve our processing power, by cropping them. Cropping offers several benefits, including reducing data size, focusing on relevant features, improving performance, and enhancing model generalization. Our method involves multiple steps: reading images using OpenCV, converting them to grayscale, applying Otsu's thresholding to isolate foreground from background, extracting foreground regions, computing bounding boxes, and ultimately cropping the original images based on these boxes. Additionally, we've integrated optional resizing functionality to standardize cropped image dimensions. This systematic approach optimizes computational resource usage while retaining pertinent image details for subsequent analysis or modeling tasks. Below, in Figure 3, we can see a randomly selected image before and after cropping.

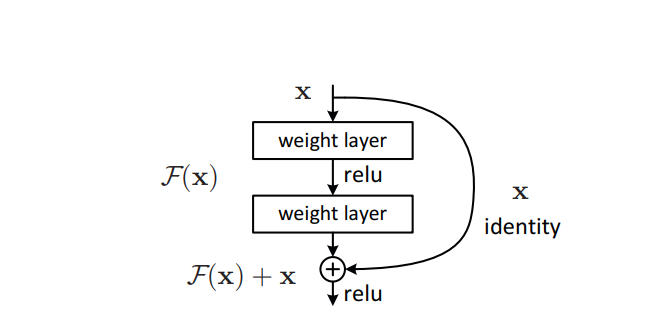


*Figure 3. Randomly selected training image before and after cropping. A significant portion of the original image has been removed, allowing our model to focus on the relevant features in the image for distinguishing between cancerous and non-cancerous cells.*

We processed the training and test sets using the function built for cropping in batches in order to efficiently process the large amounts of images/data. We used a similar approach in a preprocessing step using the Keras library. Images were preprocessed in batches and then converted to tensors for use in the future models. These steps are necessary for use of the data in the deep learning models. Finally, we saved the DataFrames of cropped images for both the train/validation and test sets as CSV for future use (and to avoid running the cropping of images again each time we continued to work to build and refine our deep learning models. The initial input of the deep learning models below is the cropped images themselves, which go through a series of pre-processing steps, and the output is a probability of the positive class.

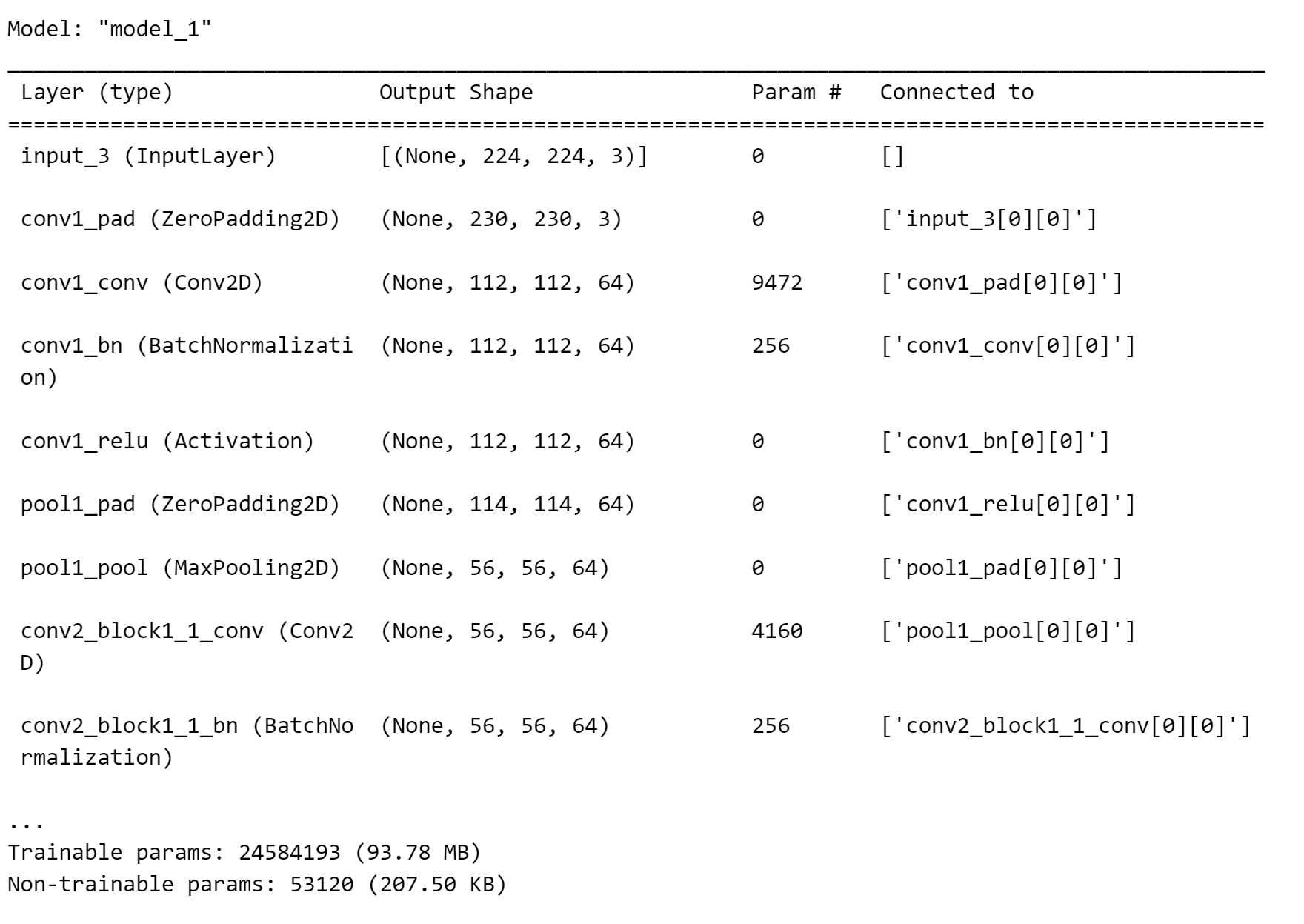
**Commercial Model Evaluation**

The next step in our analysis was to fit a pre-trained commercial model to our training data to get a baseline evaluation for our future work. We chose the ResNet50, a convolutional neural network available in the Keras library. ResNet, or residual network, was created by He, et al. and first reported in their paper, “[Deep Residual Learning for Image Recognition](https://arxiv.org/pdf/1512.03385.pdf)", using the concept of residual learning. It was created to help address the vanishing gradient problem in deep learning, allowing the model to be trained very deep. Vanishing gradients are a common problem in deeper networks and that is why this advancement has been valuable to the deep learning community. Residual learning uses shortcut connections, or skip connections, comprised of convolutional identity layers, which are not trained at first, but when the network is re-trained, all layers are expanded and the residual parts are able to explore the feature space of the image, better learning the nuances of the image. Below in Figure 4 is a diagram from the initial publication demonstrating how the connection provides a shortcut.



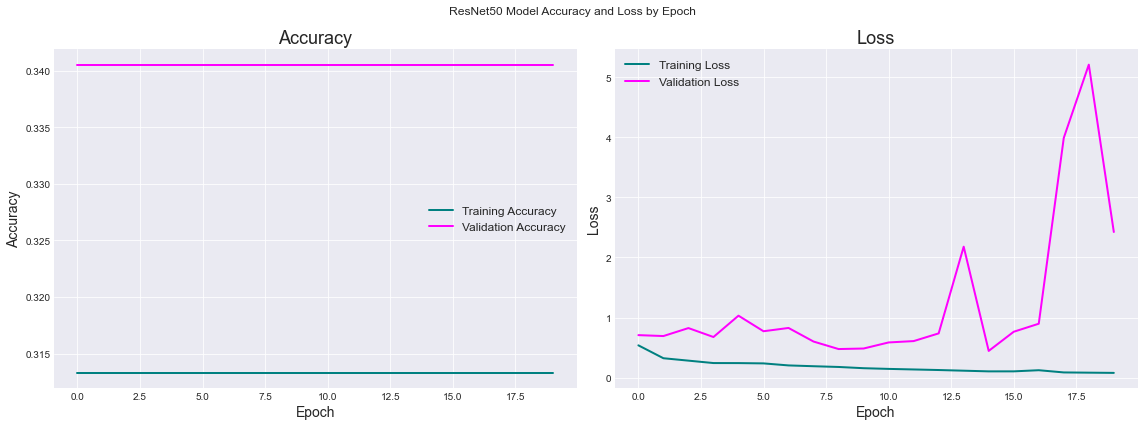
*Figure 4. Residual learning building block, from the paper “Deep Residual Learning for Image Recognition”*

On top of the model, we added additional layers to customize the model to our data and help it better learn and predict. We used similar layers to what we used in our custom built CNN in order to best compare the two approaches. Below is the truncated model summary from Keras. The full model summary, when opened in a text editor, is many pages (536 lines) long. The basic structure of the network appears as above, alternating between convolutional layers, batch normalization, and activation (ReLU layers), in the structure above with residual layers.



*Figure 5. Initial Model Summary*

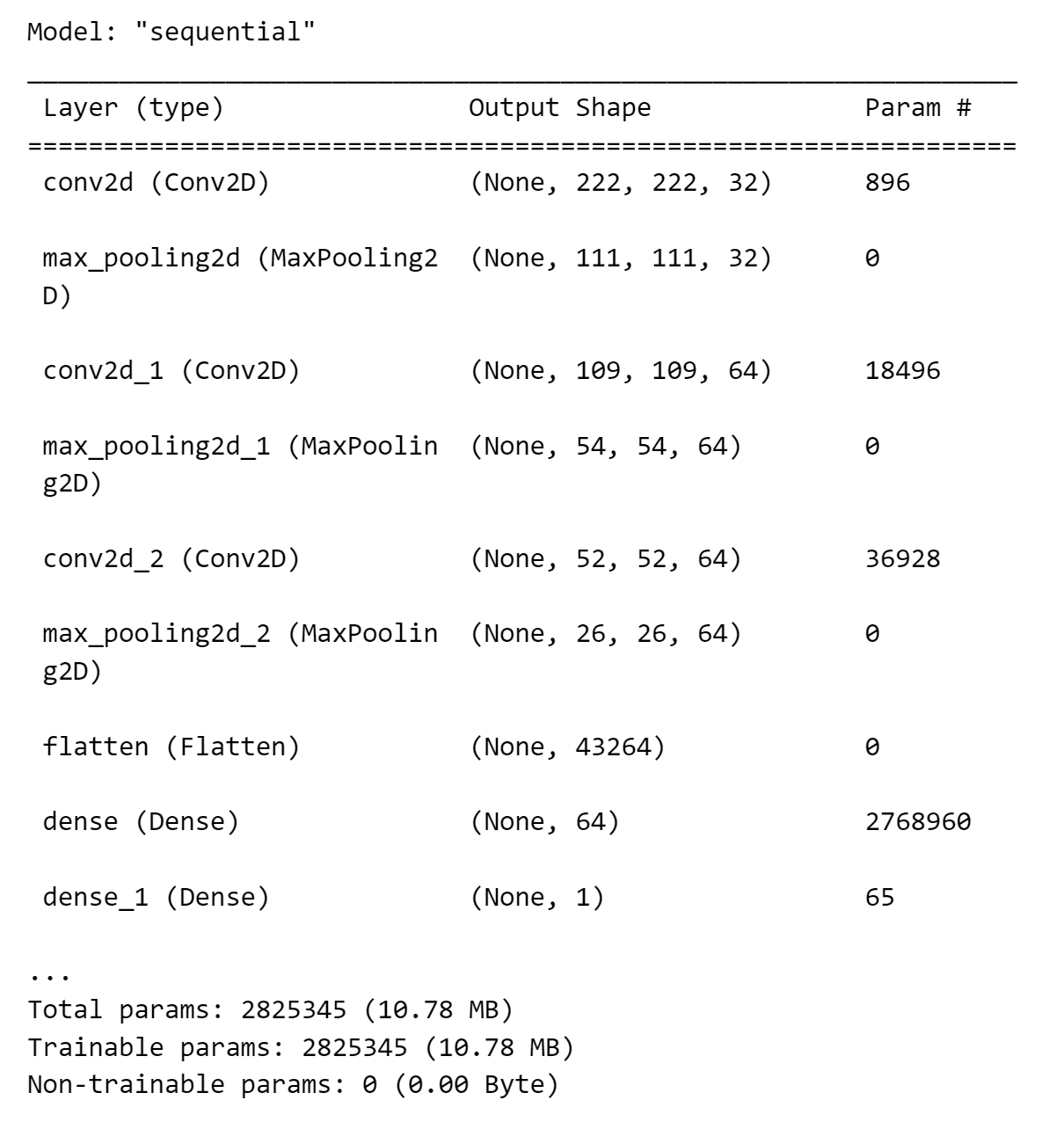
The performance of our ResNet50 model trained on our data was monitored over 20 epochs. Despite efforts to improve accuracy and mitigate loss, the model demonstrated limited improvement, plateauing around an accuracy of approximately 31.33% across both training and validation datasets. Similarly, the loss fluctuated but remained relatively high throughout training, indicating challenges in effectively learning from the data as demonstrated in Figure 6.

**

*Figure 6. ResNet model training and validation accuracy and loss over 20 epochs*

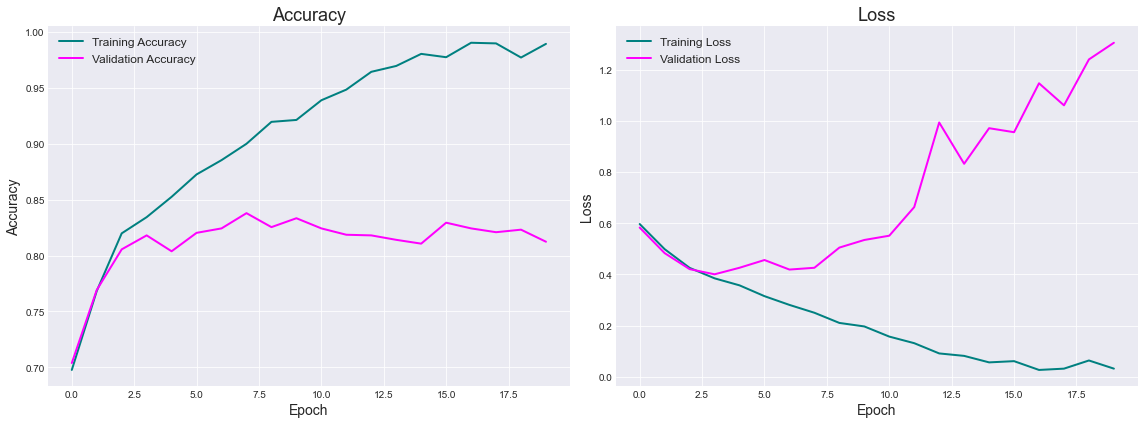
**Initial User Defined CNN**

We next fit a CNN model using the Keras Sequential model, with alternating convolutional (using ReLU activation) and max pooling layers of three layers each, followed by a flattening layer and two dense, or fully connected layers



*Figure 7. Initial CNN model parameters*

After training our model, we evaluated our initial training and validation accuracy as well as loss, as seen in Figure 8. For loss, we used binary cross entropy. Accuracy provides a measure of correct predictions versus overall predictions, expressed as a decimal (or a percentage). It is an intuitive value and provides overall performance. One challenge, though, is in imbalanced datasets, such as our own, accuracy may not provide the full picture of model performance. As we will address imbalance next, we will keep accuracy as our measure. Binary cross entropy can be applied as a loss calculation for binary classifications such as ours. It penalizes the model for large errors more than small errors.



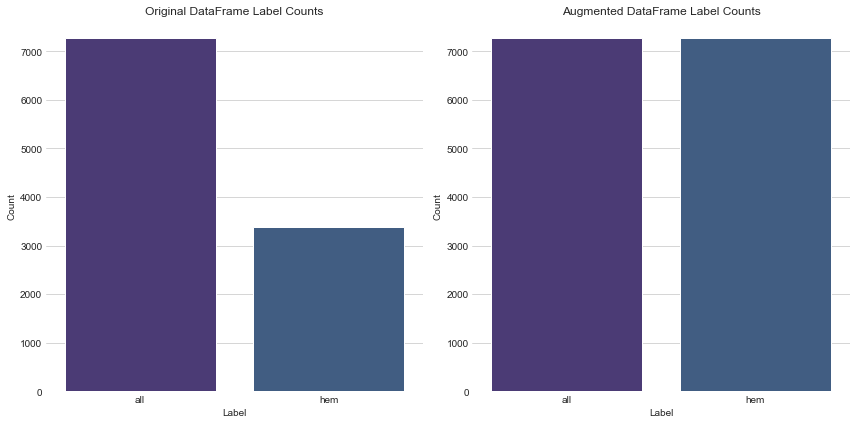
*Figure 8. Initial CNN model training and validation accuracy and loss over 20 epochs*

The plots show the training and validation accuracy and loss over 20 epochs. Initially, both training and validation accuracy increase, while the loss decreases, indicating that the model is learning. However, there seems to be overfitting as the training accuracy continues to increase, while the validation accuracy stagnates or decreases, and the validation loss increases. Next, we looked at the test set, which had a loss of approximately 3.8 and accuracy of approximately 0.67. The substantial difference between the test, validation, and training sets indicates potential overfitting in the model. While the test set results serve as a good starting point, the significant gaps between these accuracies underscore the need for improvement. To address overfitting, strategies such as regularization techniques, data augmentation, and hyperparameter tuning should be implemented. By iteratively refining the model, we can enhance its generalization performance and ensure it performs well on unseen data, thus strengthening its reliability and applicability.

**Model Refinement**

To enhance the model's performance, a series of iterative improvements were implemented. Our initial plan was to refine the model architecture to incorporate convolutional layers for feature extraction, complemented by max-pooling layers to effectively condense spatial dimensions and capture prominent features. To address overfitting, dropout regularization would be introduced (randomly deactivating neurons during training to promote better generalization). Finally, we would augment the data on our refined model. Our experience with this approach was that the model performance did not improve much, and actually worsened with regularization (both L1 and L2).

Instead, knowing our dataset was imbalanced, we decided to augment the data first with additional “hem” labeled normal images prior to any additional refinement. We created a new DataFrame with normal images randomly rotated, shifted, or flipped to add to the initial data (Figure 9) and then revisited the distribution of images in train, validation, and test sets, as seen in Figure 10.



*Figure 9. Distribution of normal (hem) and ALL cells within data before and after augmentation*

A screenshot of a data

Description automatically generated

*Figure 10. Training, validation, and test sets after data augmentation and splitting*

Next, we fit our initial model on our augmented data, reviewed loss and accuracy (Figure 11), and determined the next steps for model improvement.

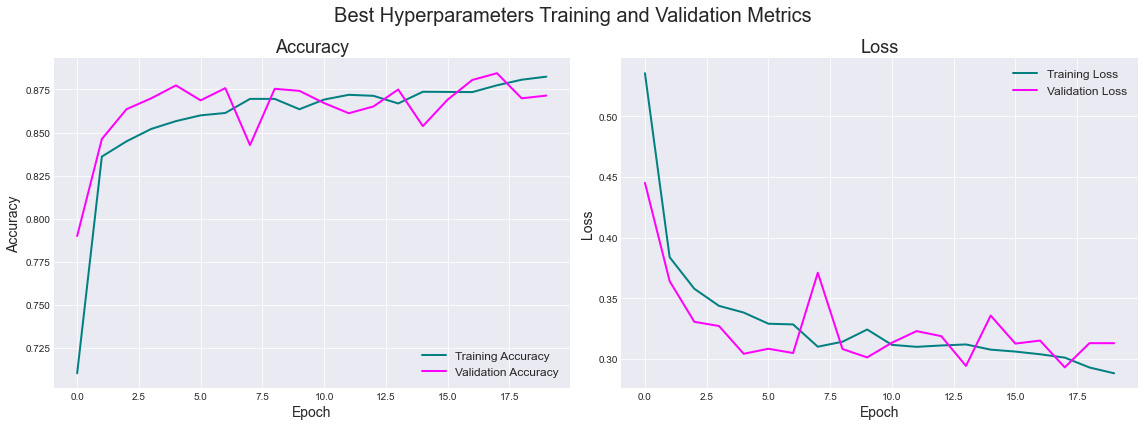


*Figure 11. Initial CNN model training and validation accuracy and loss over 20 epochs on augmented dataset*

After implementing data augmentation, we observed several notable improvements in the performance of our model. Firstly, both training and validation accuracies experienced a significant increase, indicating a marked improvement in the model's ability to generalize to unseen data. Secondly, there was a noticeable decrease in both training and validation loss, suggesting that the model had become more adept at learning representations from the data. Additionally, we observed a significant reduction in the gap between training and validation accuracies, indicating that the issue of overfitting had been effectively mitigated.

With these promising results from the augmented data, the next logical step was hyperparameter tuning. Despite the enhanced performance of the model, we recognized that fine-tuning hyperparameters could potentially unlock further optimization. We chose to explore the range of epochs, learning rates, and dropout rates to optimize our model's performance. By adjusting the number of epochs, we aimed to strike a balance between model complexity and training time, ensuring sufficient learning iterations without overfitting. Varying the learning rate allowed us to efficiently navigate the optimization landscape, seeking convergence to the optimal solution without excessive oscillations. Exploring dropout rates helped us regularize the model, enhancing its ability to generalize by preventing over-reliance on specific features.

We proceeded to implement hyperparameter tuning using grid search. Initially trained for 20 epochs with default settings for the learning rate (0.001) and dropout rate (0.0), the ideal hyperparameters were determined to be 20 epochs, a learning rate of 0.001, and a dropout rate of 0.25. The impact of finding the ideal hyperparameters for our model can be seen in Figure 12.

**

*Figure 12. Best hyperparameters CNN model training and validation accuracy and loss*

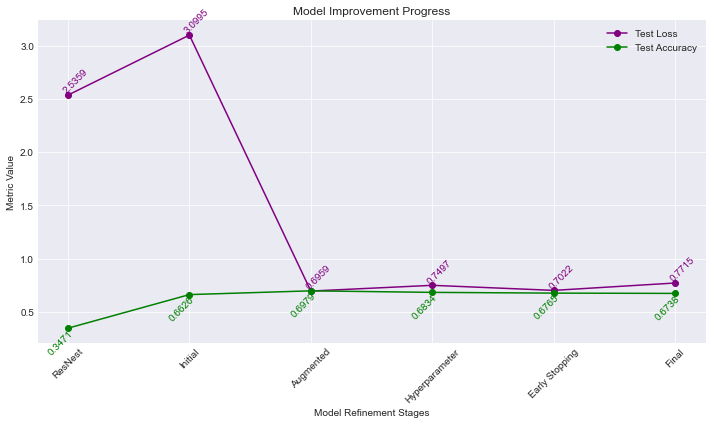
After hyperparameter tuning and analyzing the resulting configurations, our focus shifted to addressing the overfitting concerns when comparing our train & validate data performance in comparison to our test data. Implementing early stopping post-hyperparameter tuning ensured that the model was finely optimized in terms of architecture and hyperparameters. By halting training when the model's performance plateaued on a validation dataset (with patience = 3), early stopping acted as an effective regularization mechanism, preventing the model from memorizing noise in the training data and further improving its generalization ability as shown in Figure 13.

**

*Figure 13. Early Stopping CNN model training and validation accuracy and loss*

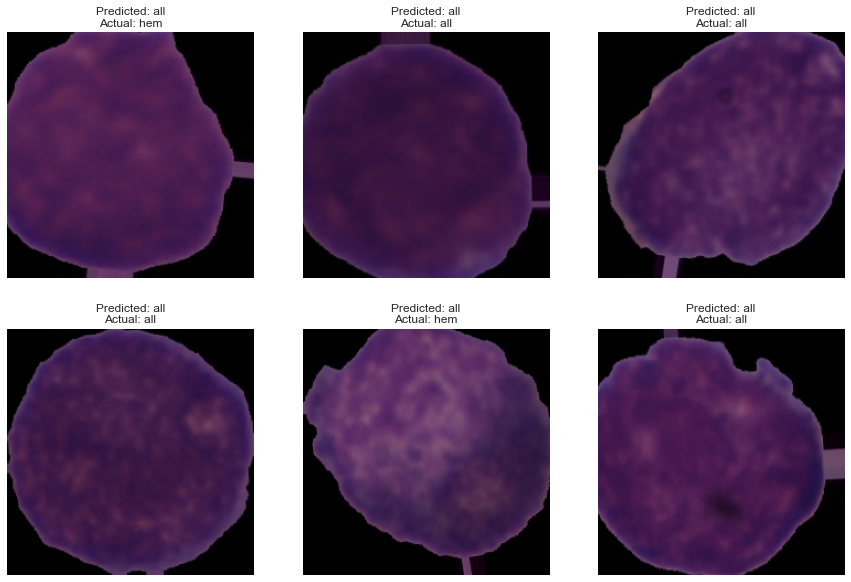
Subsequently, we moved on to cross-validate our model to obtain a more comprehensive evaluation of its performance and generalization capabilities. Cross-validation allowed us to validate the model's robustness across different subsets of the data, ensuring that its performance metrics were not biased by a specific train-test split. Additionally, cross-validation enabled us to verify the stability of the selected hyperparameters and select the final model with greater confidence. Overall, this iterative process of hyperparameter tuning, early stopping, and cross-validation contributed significantly to refining and validating our model for optimal performance on unseen data.

Finally, our CNN model reached its sixth and final form, embodying all the lessons learned along the way. Figure 14 visually captures the evolution of our model's performance across six stages of improvement, showcasing changes in test accuracy and loss.

**

*Figure 14. Six stages of CNN model improvement test accuracy and loss*

This illustrates the incremental enhancements achieved through each iteration of improvement, validating the effectiveness of our refinement approach in achieving enhanced model performance as represented in Figure 15.

**

*Figure 15. Generating predictions with final CNN model*

By harnessing our final trained model, we achieve a predictive accuracy of 0.6738 and a loss of 0.7715, enhancing decision-making processes and enabling well-informed actions. In comparison to the ResNet pretrained model (with a loss of 2.5359 and an accuracy of 0.3471) which we initially explored at the onset of our project, our custom-built model has doubled the accuracy while reducing the loss by 1.95.

The significance of our model lies in its ability to accurately discern between cancerous cells and non-cancerous "hem" cells, a pivotal task in medical diagnostics and treatment planning. Through this application, we affirm the robustness and reliability of our model in making these critical distinctions, demonstrating its potential to drive meaningful outcomes in medical research and clinical practice. This underscores the transformative impact of machine learning in advancing healthcare and enhancing patient outcomes. Given what we were able to accomplish with refinement of our simple model relative to the 50 layer deep ResNet50, further improvements with additional samples from more patients due to subject variability would be a worthwhile investment, especially given the aggressive nature of the disease.