Investigating CNN Viability and Performance within Hematopoietic Cell Classification

Whitehead Matthew[1], Ramirez Nicolas[2]

Engineering Sciences, Simon Fraser University
8888 University Drive, Burnaby, Canada

[1]mbwhiteh@sfu.ca
[2]nicolas_ramirez@sfu.ca

Keywords – Neural Networks, Biomedical Engineering, Cell Classification, Convolutional Neural Networks

I. INTRODUCTION

Hematopoietic classification plays an important role in identifying the presence of blood cancers such as leukemia and lymphoma [13]. With approximately 6700 leukemia diagnosis in Canada alone [13], the accuracy and speed of diagnosis is crucial to patient treatment and well-being. Therefore, within this paper we focus on investigating the task of automating the classification of blast (BLA), band neutrophil (NGB), segmented neutrophil (NGS), and lymphocyte (LYT) immature blood cells. In Section II, we discuss related research works in the domain of hematological classification, highlighting many advances in applying convolutional neural networks (CNNs) to medical imaging problems. In the following Section III, we analyze a dataset containing over 171, 374 bone marrow smears collected from 945 patients at the MLL, Munich Leukemia Laboratory [15]. Highlighting the relevance of why we choose to focus on BLA, NGB, NGS, and LYT class types along with the required development of a data generator.

Section IV begins with the proposal of a simple sequential CNN, which is used to evaluate the performance of two optimizers, Adadelta and Adam [2, 3]; this work can be found in Section IV-F. Following our proposed 6-layer architecture, we investigate improving model performance by varying filter sizes within convolutional and max pooling layers. Furthermore, in Section IV-C, we find our model's performance is invariant to down sampling on the input image. Subsequently leading to significant increases in computational performance. Our focus is then pivoted towards applying basic geometric image transformations to training data to mitigate a class imbalance and improve generalization. Once class imbalances are resolved increasing model depth is analyzed in Section IV-D, [6, 10] have highlighted depth is a crucial factor in obtaining optimal accuracy on classification tasks. This increase in depth leads to another proposed architecture, namely MV3 (Model Variant 3) which increased NGB cell type F_1 scores by nearly 51.5%. Finally, Section IV is concluded with the empirical determination of hyperparameters and minimization of overfitting through regularization techniques such as the addition of Gaussian noise. Our final model's recall, precision, and F_1 scores are summarized in Section V. Where we compare BLA, NGB, NGS, and LYT cell type classification metrics against Matek et al. [8], and VGG16 [10], a state-of-the-art CNN. We conclude with Section VI which suggests possible next steps in increasing class recall and precision through network ensamblement and increases in depth.

II. RELATED WORKS

Hematopoietic cell detection and classification through various forms of convolutional neural networks (CNN) is a current and growing area of research. Many researchers have created their own implementations of these CNNs, making transfer learning-based neural networks using various pre-existing models such as VGG16, Resnet, AlexNet, or Regional CNN and Support Vector Machine Architectures [19].

One team of researchers, namely Matek et al. [8], used the ResNeXt-50 architecture and applied it to analyze the 171,374-image dataset of hematopoietic cells, annotated by a team of cytologists from the Munich Leukemia Laboratory which is the same dataset [15] we have trained our model on. Matek's team found that, after utilizing data augmentation techniques by performing "clockwise rotations by a random continuous angle in the range of 0° to 180°, as well as vertical and horizontal shifts" [8], their classifier became more robust and general compared to training on the original non augmented dataset. Furthermore, they performed an 80% to 20% training and validation data split and used 5-fold cross-validation to help remove training biases. When providing their results, they used precision and recall as key metrics to analyze the performance of their algorithm. Notably, they created tolerances for all cell classes as some images contained ambiguities that would lead to their misclassification. These ambiguities are due to certain cell classes being descendants of others causing them to have very similar morphologies and can be considerably difficult to distinguish even for professional human examiners. From there, Matek's team obtained a tolerant precision of 91%, 95%, 90%, and 79% for the band neutrophils, segmented neutrophils, lymphocytes, and blast cells, respectively. They also found a tolerant recall of 91%, 85% 72% and 69%. Finally, they recommended that for future research, a binary classifier could produce more accurate results.

Another research effort on this topic has been done by Ridoy's team [20] where they looked at classifying four white blood cell types using a CNN based on the LeNet-5 model but with a fewer number of parameters and layers. To mitigate computational limitations without sacrificing the accuracy of the algorithm, they found that selecting an appropriate kernel size was key when performing feature extraction on the dataset. As such, they proposed a 7-layer model with 3 convolutional blocks including a Relu activation function, batch normalization function and a max-pooling function, followed by a max-pooling layer, a dropout layer and lastly a dense output layer where the Softmax activation function is applied. They then selected a dropout rate of 0.5, a learning rate of 0.0001, a batch size of 32, 30 epochs and Adam as their optimizer, and applied early stopping to reduce unnecessary training time. With this configuration, they found that their lymphocyte and neutrophil cell classes would obtain a precision of 97% and 81%; and a recall of 100% and 89% respectively.

With this information, we will attempt to obtain higher levels of precision, recall and F1 scores particularly for the band neutrophils, segmented neutrophils, lymphocytes, and blast cells which still experienced significant misclassification. To achieve this, we will also explore ways to improve our hyperparameter selection and configuration while keeping the methodology of the aforementioned publications in mind.

III. Dataset Collection and Pre-Processing

The dataset used is comprised of 171, 374 hematological single cell RGB JPEG images taken from 945 patients according to Matek et al., composers of the bone marrow dataset [15]. Presented in Table 1 is 21 cell classes that a sample may be labeled. Each sample contains $250x250x3$ pixels and have been expertly annotated by morphologists [8].

Table 1 - Class labels, samples, and percentage of dataset

Class label	Samples	% of dataset
(ABE) Abnormal eosinophil	8	0.005
(ART) Artefact	19630	11.455
(BAS) Basophil	441	0.257
(BLA) Blast	**11973**	**6.987**
(EBO) Erythroblast	27395	15.986
(EOS) Eosinophil	5883	3.433
(FGC) Faggot cell	47	0.027
(HAC) Hairy cell	409	0.239
(KSC) Smudge cell	42	0.025
(LYI) Immature lymphocyte	65	0.038
(LYT) Lymphocyte	**26242**	**15.313**
(MMZ) Metamyelocyte	3055	1.783
(MON) Monocyte	4040	2.357
(MYB) Myelocyte	6557	3.826
(NGB) Band neutrophil	**9968**	**5.817**
(NGS) Segmented neutrophil	**29424**	**17.170**
(NIF) Not identifiable	3538	2.065
(OTH) Other cell	294	0.172
(PEB) Proerythroblast	2740	1.599
(PLM) Plasma	7629	4.452
(PMO) Promyelocyte	11994	6.999

One key constraint early on in model development was the limited compute resources available to handle such a large and unbalanced dataset. A learning strategy that could handle such data imbalances would require increasing model depth; subsequently increasing training time. On a preliminary 6-layer architecture with over 1,000,000 trainable parameters training time was 14 hours per epoch when trained on a 2.9 GHz Quad-Core i7 Intel based processor.

Thus, we have chosen to focus on a significant subset, 45.3 %, of available data. Specifically: blast (BLA), lymphocyte (LYT), band neutrophil (NGB), and segmented neutrophil (NGS) cells. The choice of these cell classes was not random. NGS and LYT cells account for 17.17 %, and 15.31 % of all available samples. While identification of abnormal BLA, LYT, NGB, and NGS cells are extremely important in diagnosis of various infectious diseases and cancers such as acute myeloid leukemia [7].

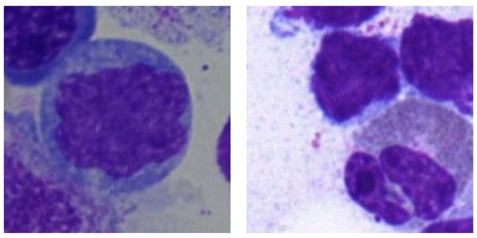
Fig. 1 - BLA & LYT Samples [15]

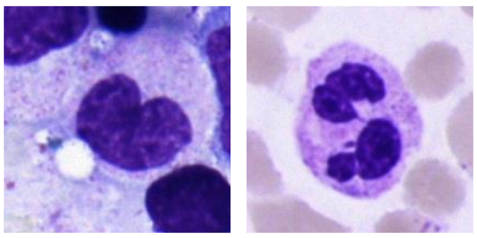
Fig. 2 - NGB and NGS Samples [15]

The new sample space of 77, 607 images was split into 90% training and 10% testing sets by randomly choosing $k = \{26482, 8721, 23618, 10776\}$ sample images from NGS, NGB, LYT, and BLA classes respectively. The training set was further split into a validation (20%) and core training set (80%). The purpose of having a test set separate from our validation set was to promote lower model bias. Validation data is explicitly used for hyperparameter tunning and thus can implicitly add bias to our proposed model through choice of hyperparameters. The entire training set was shuffled using Python's random class method *shuffle* before choosing 13, 968 samples out of the training set to compose the validation set.

Due to the inherent size of training data a custom data generator had to be implemented. This data generator inherited TensorFlow's Keras Sequence class [18] to support CPU parallelization during training. As mentioned in [18], Sequences protect against training on the same sample during an epoch. The initialization of a training batch requires the data generator to be called producing a user defined number of training samples with subsequent labeling. Normalization is used to map each pixel $p_i \in \{0, 255\} \rightarrow \{0,1\}$ such that it could be used within the Softmax activation function as discussed in Section IV.

IV. MODEL ARCHITECTURE AND PERFORMANCE

A. Proposed Model Architecture

Our first model, which can be considered as MV1 (Model Variant 1), has 6 layers with $\approx 1.33 \times 10^6$ trainable parameters and is shown in table 2. Based on our findings in Section IV-F, MV1's optimizer was chosen to be Adam with an initial learning rate of 0.001 and first order moment decay of 0.8.

TABLE 2 - PRELIMINARY 6 LAYER CNN ARCHITECTURE

L_i	Input Layer
	$Input\ 250 \times 250 \times 3$
0	$Conv2D\ 256\ (3 \times 3) \rightarrow Relu \rightarrow MaxPooling\ 3x3$
	Hidden Layers
1	$Conv2D\ 256\ (3 \times 3) \rightarrow Relu \rightarrow MaxPooling\ 3x3$
2	$Conv2D\ 256\ (3 \times 3) \rightarrow Relu \rightarrow MaxPooling\ 3x3$
3	$Conv2D\ 64\ (3 \times 3) \rightarrow Relu \rightarrow MaxPooling\ 2x2$
4	$Dense\ \ 64 \rightarrow Relu \rightarrow Dropout\ 0.5$
5	$Dense\ \ 32 \rightarrow Relu$
	Output Layer
6	$Dense\ \ 4 \rightarrow Softmax$

TABLE 3 - MODIFIED 6 LAYER CNN ARCHITECTURE

L_i	Input Layer
	$Input\ 128 \times 128 \times 3$
0	$Conv2D\ 256\ (3 \times 3) \rightarrow Relu \rightarrow MaxPooling\ 2x2$
	Hidden Layers
1	$Conv2D\ 256\ (3 \times 3) \rightarrow Relu \rightarrow MaxPooling\ 2x2$
2	$Conv2D\ 128\ (3 \times 3) \rightarrow Relu \rightarrow MaxPooling\ 2x2$
3	$Conv2D\ 128\ (3 \times 3) \rightarrow Relu \rightarrow AveragePooling\ 2x2$
4	$Dense\ \ 64 \rightarrow Relu \rightarrow Dropout\ 0.4$
5	$Dense\ \ 32 \rightarrow Relu$
	Output Layer
6	$Dense\ \ 4 \rightarrow Softmax$

Both non-linear activation functions used are ReLu, equation 1, and Softmax, equation 3. ReLu is used to minimize the possibility of a vanishing gradient as the derivative is constant for all values $x \in \{-\infty, 0\} \cup \{0, \infty\}$. Whereas an activation function like Sigmoid has a derivative of 0 for values near its asymptotic bounds. Softmax is utilized as the last activation function as the domain of our problem is categorical and requires the output vector of dimension 4 to be mapped into a probability distribution in the range of $\{0, 1\}$. Finally, categorical cross-entropy loss, equation 2, is utilized to compute the error across all training samples.

$$(1)\ ReLu(x) = \max\{0, x\}$$

$$(2)\ Cross\ Entropy\ Loss = -\sum_{i=0}^{N-1} [\sum_{j=0}^{3} Y_{ij} \ln(\hat{Y}_{ij})]$$

$$(3)\ Softmax(\boldsymbol{Y_{ij}}) = \frac{e^{\boldsymbol{Y}_{ij}}}{\sum_{k=0}^{|\boldsymbol{Y}_i|} e^{k}}$$

B. Evaluation Metrics

Recall, equation 4, precision, equation 5, and an F_1 score, equation 6, will be used to evaluate each individual class when obtaining final evaluation metrics.

$$(4)\ Recall = \frac{TP}{TP + FN}$$

$$(5)\ Precision = \frac{TP}{TP + FP}$$

$$(6)\ F_1 = 2 \times \frac{Recall \times Precision}{Recall + Precision}$$

C. Training and Further Architectural Design

Due to the considerable execution time for one epoch (5 hours) we ran MV1 for a total of two epochs. Achieving a final training accuracy of 77%, figure 4, and precision and recall scores in the range $\{0, 0.89\}$ depending on the class, figure 3. As expected, segmented neutrophil (NGS) and lymphocytes (LYT) show higher recall and precision compared to blast (BLA) and band neutrophil cells (NGB). This result is most likely attributed to the higher number of training samples for LYT and NGS classes. One interesting result of MV1 is its inability to identify a NGB true positive. Shown in figure 3, the precision and recall metrics are zero for epochs 1 and 2 which negatively affects our overall training accuracy. This poor performance may be attributed to NGB classes being classified as a NGS cell as a NGS cell is derived from a NGB cell.

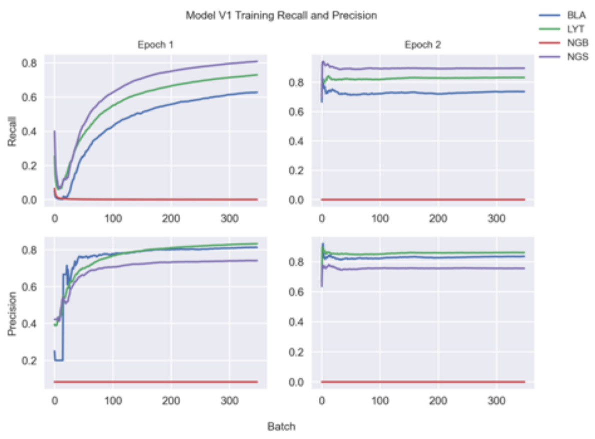

Fig. 3 - MV1 Training Precision & Recall

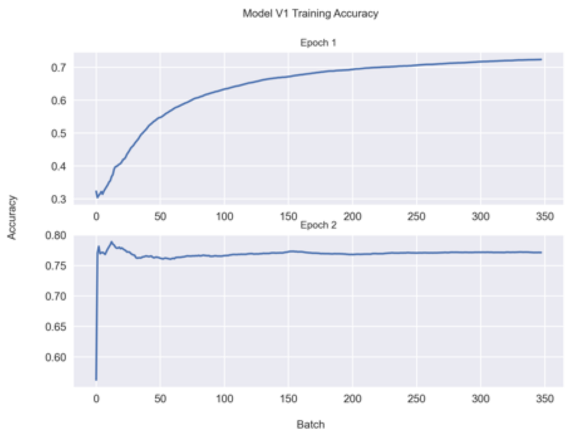

Fig. *4* - MV1 Training Accuracy

To increase classification accuracy two methods are imposed. The first is modification of MV1's architecture by reducing the max pooling kernel size to 2x2 and increasing the number of filters within L_3 (layer 3). Batch normalization was added after each convolution output preceding the ReLu activation function, as suggested by Ioffe and Szegedy [6]. Moreover, the learning rate and first order moment decay was increased to 0.003 and 0.9 respectively. Input images were resampled to fit a $128 \times 128 \times 3$ shape using nearest neighbors interpolation. Notably reducing the number of computational resources required to execute each training epoch. The summary of modified architectural changes is shown in table 3.

Training this modified architecture (MV2) with smaller input dimensions for 10 epochs boosted validation recall of the NGB class to 0.810 and precision to 0.623. A significant improvement compared to MV1. We also saw significant gains in validation recall and precision for BLA, and LYT classes, as shown in figure 5.

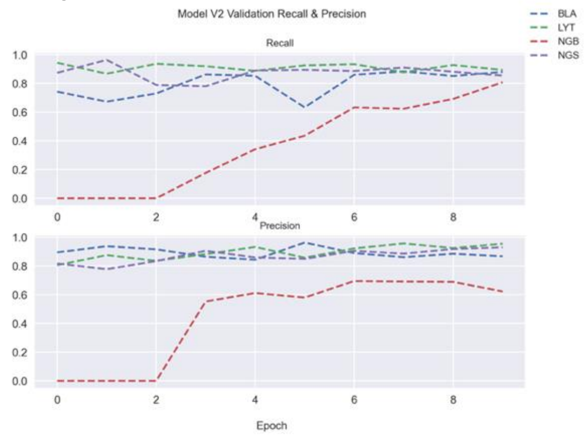

Fig. 5 – MV2 Validation Recall and Precision

Furthermore, as depicted in figure 5, down sampling does seem to have prevalence in improving model accuracy. Therefore, we further down sampled the input image to $64 \times 64 \times 3$ to investigate if our proposed model was invariant to a lower input size. To promote the maximum number of epochs that would result in constructive learning early stopping was implemented to monitor changes in validation loss with a patience of 10. Training with the further down sampled input size resulted in a maximum validation accuracy of 88.2 % and loss of 0.306 at epoch 7. Comparing this with an input size of $128 \times 128 \times 3$ showed a minute classification gain of ≈ 0.2 %. As we are still in the early stages of empirical model development the change in accuracy is not enough to promote changing the structure of MV2 to have an input size of $64 \times 64 \times 3$.

An issue that must be mitigated is the large data imbalance with NGB class types. To alleviate this, we perform a simple 90-degree rotation on all NGB training samples increasing respective class samples to 15, 698 images. Notably, using a simple rotation ensures the label of the sample does not change. Although basic, such geometric transformations are popular augmentation choices among researchers within the deep learning medical field [8]. Moreover, data augmentation has shown to act as a form of model regularization promoting further generalization to unseen data [9]. Following the application of offline data augmentation, we first modified MV2 architecture to reduce the number of max pooling layers as the sample spatial dimensions were becoming significantly reduced, constraining model depth. We further increased depth by adding two convolutional layers with 64 3×3 filters, one 128 neuron densely connected layer and finally increased the number of filters in L_5 to 64.

Training was performed on MV2's modified architecture for 15 epochs until early stopping criteria was met; the validation loss failed to converge for 5 sequential epochs. Although early stopping criteria was met, the results achieved are quite promising, we obtained a substantial increase in precision, 0.87, and recall, 0.83, for the NGB class while maintaining previously obtained metrics for LYT, NGS, and BLA classes. Shown in figure 6 is the categorical accuracy which reaches a maximum of 88.97 % at epoch 7. Therefore, by using a basic geometric transformation and modestly expanding model depth we were able to increase our model's NGB F_1 score to 0.415 from 0.352.

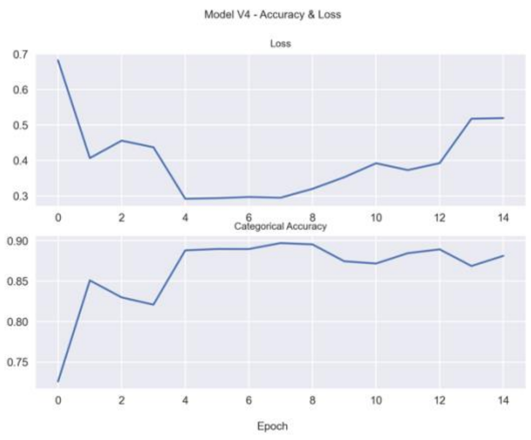

Fig. 6 – MV2 Modified with Data Augmentation Loss and Accuracy

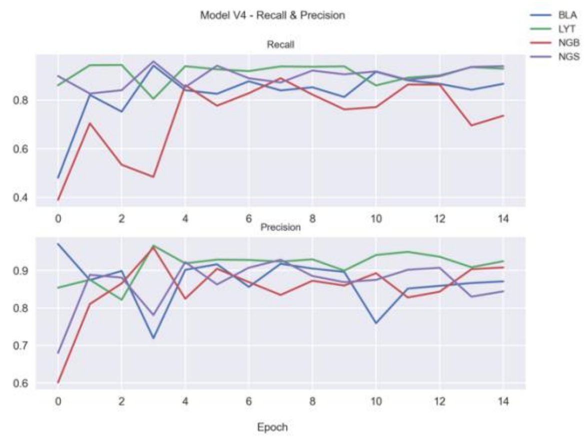

Fig. 7 – MV2 Modified with Data Augmentation Validation Recall and Precision

D. Expanding Model Depth & Developing a New Architecture

An architectural characteristic dominated by nearly all high performing deep learning networks such as VGG16 [10], and ResNet [6] is model depth. Simonvan and Zisserman [10] were able to show that by increasing model depth while maintaining a small 3 × 3 convolutional filter resulted in state-of-the-art classification performance. Following this research, we inherit this depth wise strategy and turn our focus onto developing a 3rd architecture called MV3 (Model Variant 3).

MV3 contains 12 layers in total, 8 are convolutional and 4 are fully connected. The filter size within each convolutional layer is 3x3, following that of VGG16 [10]. However, one notable difference to VGG16 is we decrease the number of filters as model depth increases. Each convolutional layer is followed by batch normalization to standardize the weights of each convolutional layer, promoting a more Gaussian like distribution [4]. One feature mentioned in [4] is how batch normalization can make deep learning networks more resilient to changes in hyperparameters and weight initialization methods. Our weight initialization method is HeNormal which is inspired by He et al. in [11] and initializes weights using a normal distribution with a standard deviation dependent on the convolutional layer weight size. Each layer is followed by a ReLu activation function and max pooling is incorporated after some layers with a filter size of 2 × 2 and stride of 2 to reduce dimensionality and make highly weighted features within a cell sample more prominent. A summary of MV3 is shown in table 4; it should be noted that we used a learning rate of 0.003, batch size of 32, and Adam as the optimizer.

TABLE 4 – MODEL VERSION 3 (MV3) ARCHITECTURE

L_i	Input Layer
	$Input\ 128 \times 128 \times 3$
0	$Conv2D\ 256\ (3 \times 3) \rightarrow Relu$
	Hidden Layers
1	$Conv2D\ 256\ (3 \times 3) \rightarrow BN \rightarrow Relu$
2	$Conv2D\ 256\ (3 \times 3) \rightarrow BN \rightarrow Relu \rightarrow MaxPooling\ 2x2$
3	$Conv2D\ 128\ (3 \times 3) \rightarrow BN \rightarrow Relu$
4	$Conv2D\ 128\ (3 \times 3) \rightarrow Relu$
5	$Conv2D\ 128\ (3 \times 3) \rightarrow BN \rightarrow Relu \rightarrow MaxPooling\ 2x2$
6	$Conv2D\ 64\ (3 \times 3) \rightarrow BN \rightarrow Relu$
7	$Conv2D\ 64\ (3 \times 3) \rightarrow BN \rightarrow Relu \rightarrow MaxPooling\ 2x2$
8	$Dense\ 256 \rightarrow Relu$
9	$Dense\ 128 \rightarrow Relu \rightarrow Dropout\ 0.5$
10	$Dense\ 64 \rightarrow Relu$
	Output Layer
11	$Dense\ 4 \rightarrow Softmax$

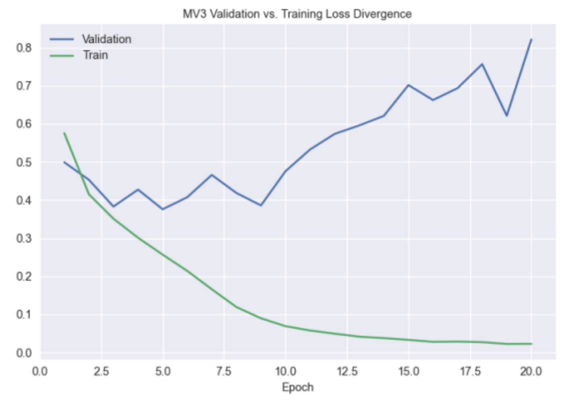

Fig. 8 – MV3 Training and Validation Loss Comparison

E. MV3 with Regularization & Hyperparameter Tunning

MV3 was trained for a total of 29 epochs and resulted in the highest yet training accuracy. However, figure 8 depicts the validation loss diverging from training loss near epoch 10. Hinting at possible overfitting as the validation loss continues to increase with training.

To prevent further overfitting, we look at adding regularization methods such as inserting Gaussian noise between certain layers and reducing the learning rate. Based on what is shown in figure 8, the validation loss is non monotonic which may be attributed to the learning rate being too large. Cereris paribus we will decrease the learning rate from 0.003 to 0.001 to investigate correcting our observed overfitting problem. A decrease in the step size taken during gradient updates should permit the gradient to coverage slower. This modification will affect α in equation 7 reducing the impact of the derivative of the error, shown in equation 8, with respect to $\boldsymbol{W^l}$, the weight matrix of the respective convolution layer.

$$(7)\ W^l := W^l - \alpha \frac{\partial E}{\partial \boldsymbol{W^l}} \boldsymbol{W^l}$$

$$(8)\ \frac{\partial E}{\partial \boldsymbol{W}^l} = \sum_{i=0}^{N} \left(\frac{Y_i}{\widehat{Y}_i} \times \frac{\partial \widehat{Y}_i}{\partial \boldsymbol{W}_l}\right)$$

As expected, a slight adjustment in the learning rate yielded a lower validation loss. The arithmetic mean of the validation loss for MV3 with different learning rates is shown in table 5. The mean was calculated using values within epochs 7 to 20 to compensate for the initial skewed losses. Lowering the learning rate to 0.0001 resulted in the lowest validation loss of 0.5040, a -16.64 % change compared to a learning rate of 0.003.

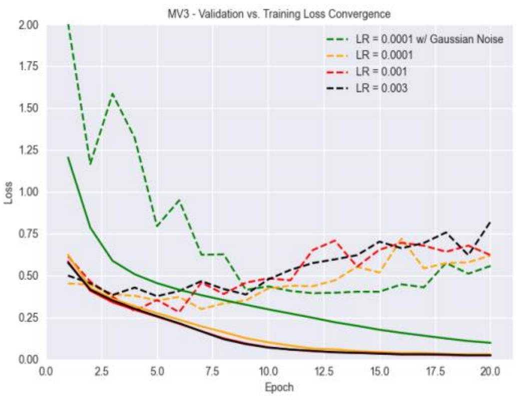

Fig. 9 – MV3 Validation & Training Loss Comparison

TABLE 5 – MV3 LEARNING RATE VARIATION MEAN LOSS

Learning Rate	Mean	% Change	Gaussian Noise
0.003	0.6046	0	False
0.001	0.5905	-2.33	False
0.0001	0.5040	-16.64	False
0.0001	0.4610	-23.75	True

Figure 9 shows that training MV3 with a lower learning rate yields promising results but slight over fitting is still present. This can be seen as the validation loss diverges after the 10th epoch in figure 9. Thus, we further our regularization experiments by adding Gaussian noise between certain convolutional layers. The addition of Gaussian noise is a computationally efficient optimization technique for complex architectures [12]. Decreasing training loss and increasing model accuracy on learning tasks such as MNIST classification [12]. Each layer of Gaussian noise was centered with a mean of zero, standard deviation of 1 and inserted before L_1, $L_3, L_6, and\ L_8$ in table 4. The qualitative approach to inserting noise at these specific instances was to regularize perturbed prominent features between convolution blocks after max pooling. Training MV3 for 20 epochs with Gaussian noise and a learning rate of 0.0001 resulted in a mean validation loss of 0.4610 over epochs 7 to 20, a -23.75% change compared to MV3 with no noise and a learning rate of 0.003. Comparing these promising results with MV3 when trained using a learning rate of 0.0001 and no noise resulted in a -8.53% change in validation loss. Therefore, MV3 will adopt a learning rate of 0.0001 with Gaussian noise layers added before $L_1, L_3, L_6, and\ L_8$.

Depicted in figure 9 is a green solid curve that shows a steady increase in validation loss. Although diminishing, MV3 still shows overfitting, even with Gaussian noise. Leading to the introduction of L2 regularization. within all fully dense and convolutional layers. The L2 norm is using a standard penalty factor of 0.01 as default in Keras [18]. Initially, L2 regularization showed no improvement over MV3 with no penalty. However, as training progressed, MV3 with L2 regularization started to exhibit a slow convergence of validation loss towards training loss.

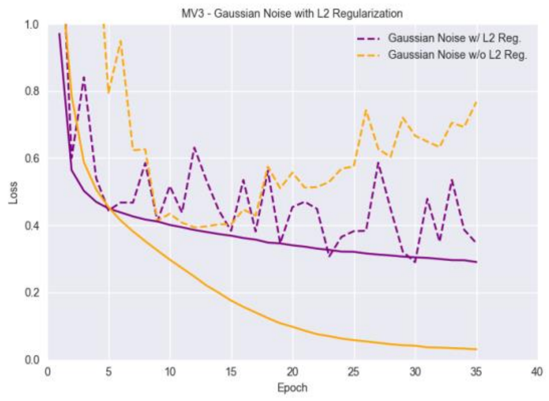

Fig 10 – MV3 Validation Loss with L2 Regularization

Figure 10 shows a purple dashed curve representing a slight non-monotonic convergence of the validation loss towards the training loss, the solid purple curve. Whereas MV3 with no regularization, dashed yellow curve, is diverging from the respective solid yellow training loss.

In summary, we find that by training MV3 with a lower learning rate 0.0001, rather than the proposed 0.003, adding Gaussian noise, and utilizing an L2 norm penalty with a factor of 0.01 resulted in a converging but erratic behaving validation loss curve. Due to the preceding results MV3 will be chosen as our final model to compare against other works within the cell classification and CNN field, namely [8] and [10].

F. Choice of a Gradient Descent Optimizer

The choice of an optimizer was made by examining two stochastic gradient decent (SGD) optimization algorithms, Adadelta [2], and Adam [3]. Adadelta was initially chosen to implement an adaptive learning rate strategy that prevents continual learning rate decay; a significant deterrence in using an optimizer like Adagrad [2]. Adadelta uses a root mean square of a windowed sum of squared gradients to scale the learning rate during parameter updates. The windowed sum of squared gradients is implemented as the expected value of exponentially decaying squared gradients as shown by Zeiler in [2]. To control the rate of decay Adadelta uses a parameter ρ to control the fraction $(1-\rho)$ of the computed gradients to include within the updated expected value of exponentially decaying gradients. To choose an optimal value for ρ our model was trained with 8000 samples over one epoch while varying ρ and maintaining a constant learning rate of 0.1. Depicted in figure 11 is the training loss as a function of mini-batches, batch size 32, for different values of $\rho \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. Based on our findings having $\rho = 0.9$ yielded the lowest training loss.

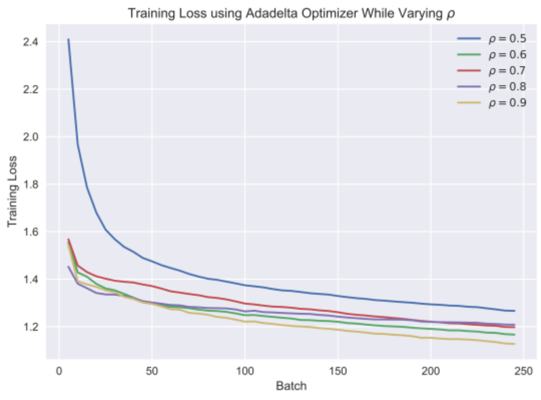

Fig. 11 - Training loss using Adadelta while varying decay rate

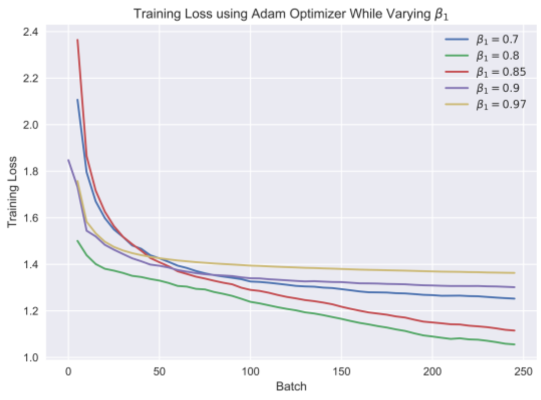

Fig. 12 – Training loss using Adam while varying β_1

Next Adam, proposed by Ba and Kingma, [3] was tested within our architecture by varying the parameter β_1 which controls the exponential decay rate of the estimated gradient mean. Using $\beta_1 \in \{0.7, 0.8, 0.85, 0.9, 0.97\}$, $\beta_2 = 0.999$, and a learning rate of 0.001 resulted in the following training losses presented in figure 12. To guarantee an accurate comparison to Adadelta training was done on the same 8000 samples over one epoch.

We found that having $\beta_1 < 0.9$ resulted in better convergence to a global minimum on the error surface compared to $\beta_1 \geq 0.9$. One explanation is derived from having a larger upper bound on the magnitude of parameter updates. As explained by Ba and Kingma [3], the upper bound is shown in equation 9 where α is the learning rate.

$$(9)\ |\Delta_t| \leq \alpha \cdot \frac{(1-\beta_1)}{\sqrt{1-\beta_2}}$$

Due to $1-\beta_2 = 0.999$ a higher β_1 will result in a larger maximum bound. As shown in figure 12 a value of $\beta_1 = 0.8$ resulted in the lowest training loss which would permit a maximum difference of $6.329 \cdot 10^{-3}$ in training parameter values, subsequently speeding up convergence.
Empirically Adam with a learning rate of 0.001 and $\beta_1 = 0.8$ achieves a lower training loss compared to Adadelta with a

learning rate of 0.1 and $\rho = 0.9$. Although both training categorical cross entropy losses were close 1.06 and 1.12 respectively, we decided to proceed with Adam as our final optimizer using the optimal parameters stated above.

V. Final Results and Comparison

Training our empirically determined architecture, MV3, in section IV-E for 60 epochs resulted in the following metrics, table 6, after evaluation on 7,763 test samples.

TABLE 6 – FINAL MV3 TEST RESULTS

Class Name	Recall	Precision	F1
NGB	0.7302	0.6431	0.6839
NGS	0.8342	0.9296	0.8793
LYT	0.9208	0.9152	0.9180
BLA	0.8431	0.8821	0.8622

We can see that the lymphocyte (LYT) class has the highest F_1 score followed by segmented neutrophil (NGS) cells, then blast (BLA) cells, and finally band neutrophil (NGB) cells.

Matek et al. [8] who implemented a sequential CNN to classify the same dataset achieved the following recall, precision and F_1 metrics presented in table 7, it should be noted that these metrics are based on mean values using strict evaluation. As mentioned in [8] strict evaluation differs from tolerant evaluation, which exhibits higher metrics, in not penalizing classification errors where cell types are difficult to distinguish between. The idea of "tolerance classes" within classification comes from the works of Krappe et al. [14].

TABLE 7 – MATEK ET AL. [8] CLASSIFIER RESULTS

Class Name	Recall	Precision	F1
NGB	0.65	0.54	0.59
NGS	0.71	0.92	0.80
LYT	0.70	0.90	0.79
BLA	0.65	0.75	0.70

Comparing results from tables 6 and 7 we can see that our 12-layer model outperforms Matek et al. [8] classifier in all cell classes. An exciting result in support of using a more cell specific classifier within hematological cell classification. The increased accuracy can be attributed to focusing on a smaller subset of the 21-class dataset [15]. However, our model may not be superior as the subset of data MV3 was trained contains less samples and classes than Matek et al.

Furthermore, VGG16, a state-of-the-art architecture, [10] was trained on the same augmented dataset as MV3 using Adam [3] as the optimizer with a learning rate of 0.0001. With VGG16 [10] achieving 2nd place in Image Net's Large Scale Visual Recognition Challenge (ILSVRC) 2014 we expected MV3 would be inferior which was indeed the case. Shown in table 8 is the recall, precision, and F_1 metrics using a VGG16 architecture.

TABLE 8 – VGG 16 [10] TEST RESULTS

Class Name	Recall	Precision	F1
NGB	0.8114	0.6856	0.7432
NGS	0.8532	0.9493	0.8986
LYT	0.9501	0.9132	0.9313
BLA	0.8331	0.9148	0.8729

VI. Future Works

Further progress can be made on MV3 architecture by first expanding training to include the entire 21 class bone marrow smear dataset [15]. Hopefully, increasing generalizability and usability of the model. A second area of consideration is increasing the depth of MV3 while maintaining small convolution filter sizes, as done with VGG 16 [10]. Finally, model ensembalment which entails using the results of multiple neural networks to make a final prediction, has shown great success in ILSVRC [16] to increase classification accuracy.

VII. Conclusion

In summary, we began with investigating the relevance of down sampling to increase classification accuracy due to initial computational limitations. Showing that our proposed model was invariant to a 77% reduction in dimensionality. Subsequently increasing training and validation performance. Secondly, we utilized data augmentation methods, geometric transformations to be exact, to reduce dataset imbalances resulting in a promising 18 % increase in the F_1 score of band neutrophil cells. Thirdly, model depth was increased to 12 layers leading to a new proposed architecture MV3 which was trained with different combinations of hyperparameters to determine methods in reducing over-fitting. After settling on a set of well performing hyperparameters Gaussian noise was added after certain convolution layers to promote generalization [12]. Finally, L2 regularization was added to all convolution kernels leading to a final proposed model for evaluation and comparison in Section V. Within Section V our test results were compared to the sequential CNN developed by Matek et al. [8]. Showing great promise as F_1 scores were higher for all class types. MV3 was further benchmarked against VGG16 and proved to be inferior. Leading to our final discussion within section VI on the possible ways of furthering the work on MV3 to increase classification accuracy, recall, precision, and F_1 metrics.

References

[1] P. Chlap, H. Min, N. Vandenberg, J. Dowling, L. Holloway, and A. Haworth, "A review of medical image data augmentation techniques for deep learning applications," *J Med Imag Rad Onc*, vol. 65, no. 5, pp. 545–563, Aug. 2021, doi: 10.1111/1754-9485.13261.

[2] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” *arXiv:1212.5701 [cs]*, Dec. 2012, Accessed: Apr. 03, 2022. [Online]. Available: http://arxiv.org/abs/1212.5701

[3] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *arXiv:1412.6980 [cs]*, Jan. 2017, Accessed: Apr. 03, 2022. [Online]. Available: http://arxiv.org/abs/1412.6980

[4] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *arXiv:1502.03167 [cs]*, Mar. 2015, Accessed: Apr. 03, 2022. [Online]. Available: http://arxiv.org/abs/1502.03167

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, Massachusetts: The MIT Press, 2016.

[6] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *arXiv:1512.03385 [cs]*, Dec. 2015, Accessed: Mar. 29, 2022. [Online]. Available: http://arxiv.org/abs/1512.03385

[7] H. M. Amin *et al.*, “Having a higher blast percentage in circulation than bone marrow: clinical implications in myelodysplastic syndrome and acute lymphoid and myeloid leukemias,” *Leukemia*, vol. 19, no. 9, pp. 1567–1572, Sep. 2005, doi: 10.1038/sj.leu.2403876.

[8] C. Matek, S. Krappe, C. Münzenmayer, T. Haferlach, and C. Marr, “Highly accurate differentiation of bone marrow cell morphologies using deep neural networks on a large image data set,” *Blood*, vol. 138, no. 20, pp. 1917–1927, Nov. 2021, doi: 10.1182/blood.2020010568.

[9] N. Alofi, W. Alonezi, and W. Alawad, “WBC-CNN: Efficient CNN-Based Models to Classify White Blood Cells Subtypes,” *Int. J. Onl. Eng.*, vol. 17, no. 13, pp. 135 - 150, Dec. 2021, doi: 10.3991/ijoe.v17i13.27373.

[10] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv:1409.1556 [cs]*, Apr. 2015, Accessed: Apr. 05, 2022. [Online]. Available: http://arxiv.org/abs/1409.1556

[11] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” *arXiv:1502.01852 [cs]*, Feb. 2015, Accessed: Apr. 05, 2022. [Online]. Available: http://arxiv.org/abs/1502.01852

[12] A. Neelakantan *et al.*, “Adding Gradient Noise Improves Learning for Very Deep Networks,” *arXiv:1511.06807[cs, stat]*, Nov. 2015, Accessed: Apr. 05, 2022. [Online]. Available: http://arxiv.org/abs/1511.06807

[13] Canadian Cancer Statistics Advisory Committee in collaboration with the Canadian Cancer Society, Statistics Canada, and the Public Health Agency of Canada. Canadian Cancer Statistics 2021. Toronto, ON: Canadian Cancer Society; 2021, Accessed: Apr. 8, 2022. [Online]. Available: cancer.ca/Canadian-Cancer-Statistics-2021-EN

[14] S. Krappe, T. Wittenberg, T. Haferlach, and C. Münzenmayer, “Automated morphological analysis of bone marrow cells in microscopic images for diagnosis of leukemia: nucleus-plasma separation and cell classification using a hierarchical tree model of hematopoesis,” San Diego, California, United States, Mar. 2016, p. 97853C. doi: 10.1117/12.2216037.

[15] Matek, Christian, Krappe, Sebastian, Münzenmayer, Christian, Haferlach, Torsten, and Marr, Carsten, “An Expert-Annotated Dataset of Bone Marrow Cytology in Hematologic Malignancies.” The Cancer Imaging Archive, 2021. doi: 10.7937/TCIA.AXH3-T579.

[16] C. Szegedy *et al.*, “Going Deeper with Convolutions,” *arXiv:1409.4842 [cs]*, Sep. 2014, Accessed: Apr. 10, 2022. [Online]. Available: http://arxiv.org/abs/1409.4842

[17] G. Drałus, D. Mazur, and A. Czmil, “Automatic Detection and Counting of Blood Cells in Smear Images Using RetinaNet,” *Entropy*, vol. 23, no. 11, p. 1522, Nov. 2021, Accessed: Apr. 10, 22. doi: 10.3390/e23111522.

[18] Keras Team, “Keras Documentation: Python & numpy utilities,” *Keras*, 2022. [Online]. Available: https://keras.io/api/utils/python_utils/#sequence-class. [Accessed: 10-Apr-2022].

[19] C. Cheuque, M. Querales, R. León, R. Salas, and R. Torres, “An Efficient Multi-Level Convolutional Neural Network Approach for White Blood Cells Classification,” Diagnostics, vol. 12, no. 2, p. 248, Jan. 2022, doi: 10.3390/diagnostics12020248.

[20] M. A. R. Ridoy and M. R. Islam, "An Automated Approach to White Blood Cell Classification Using a Lightweight Convolutional Neural Network," 2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT), 2020, pp. 480-483, doi: 10.1109/ICAICT51780.2020.9333512.