Project Report - ECE 176
Enhancing White Blood Cell Classification through GAN-based Data Augmentation

Rupashi Sangal
MS ECE: Machine Learning and Data Science
A59019496

Vyshnavi Sankaran Krishnan
MS ECE: Machine Learning and Data Science
A59019452

Abstract

White blood cell classification plays a critical role in the diagnosis of blood-related diseases. Deep learning models can be used for this purpose effectively. However, the shortage of labeled medical imaging data hinders the development of these models. We propose to use General Adversarial Networks to generate synthetic blood cell images to improve the classification performance using CNN and CNN+RNN Models. We also test its performance against conventional image augmentation techniques. CNN+RNN model is found to perform the best with an ensemble augmentation using GAN and traditional techniques with a test accuracy of 87.77%.

1 Introduction

The human body contains five distinct types of white blood cells, namely Neutrophils, Lymphocytes, Monocytes, Eosinophils and Basophils. Each of these types is endowed with unique characteristics that contribute to the overall maintenance of the body's health and proper functioning. The accurate classification of white blood cells plays a critical role in the diagnosis and treatment of various blood-related disorders. However, manual classification of blood cells by trained professionals is a time-consuming and expensive task, and can also be prone to errors.

Deep learning based image classification models can be employed to solve this problem effectively. However, limited annotated medical imaging data is a persistent challenge for training these deep learning models.

In this project, we propose a solution to enhance white blood cell classification accuracy with Deep learning frameworks by using Generative Adversarial Networks (GANs) for augmenting the limited available data. We understand that traditional data augmentation techniques such as rotation, flipping, and cropping have limitations in generating diverse and realistic data. To overcome these limitations, we propose the use of GAN-based data augmentation, which can generate synthetic images that closely resemble the real ones, and can also add diversity to the dataset.

The key parts of our proposed approach involve training a GAN model on a limited set of blood cell images to generate this synthetic data. We then combine this synthetic data with the original dataset to train deep convolutional neural networks (DCNN) and CNN-RNN frameworks to boost the overall performance accuracy. The effectiveness of our proposed solution will also be evaluated and compared with traditional data augmentation methods.

Our results demonstrate that the GAN-based data augmentation approach leads to significant improvements in the overall accuracy. Specifically, we achieved an accuracy of 87.77% on a publicly available blood cell dataset using CNN+RNN model with augmentation, which is a significant improvement over the baseline accuracy of 48.21% using only CNN without any augmentation.

This research holds the potential to provide a cost-effective and efficient solution for accurate white-blood cell classification, which can ultimately help in the timely diagnosis and treatment of blood-related disorders. This is our underlying motivation of solving this problem.

2 Related Work

Deep learning techniques, especially convolutional neural networks (CNNs), have demonstrated exceptional performance in various image classification tasks, including medical image analysis [1, 2]. Several studies have employed CNNs for the classification of white blood cells (WBCs) from blood smear images. Liang et al. [3] proposed a method that combines CNNs with recursive neural networks (RNNs) for the classification of blood cell images. Ghosh and Kundu [4] combined neural network models for blood cell classification using features extracted by VGG-16 and InceptionV3 networks. Habibzadeh et al. [5] used pre-trained deep learning models ResNet and Inception for automatic white blood cell classification. In another study, Khamparia et al. [6] proposed a deep-learning framework for the classification of WBCs based on cell morphology, achieving an accuracy of 96.17% on the publicly available ALL-IDB dataset.

Data augmentation is a commonly used technique to address the issue of limited annotated data. Traditional data augmentation methods such as rotation, translation, and flipping have been widely used in medical image analysis [7, 8]. However, these methods may not capture the complex variations in the data distribution, resulting in limited diversity in the augmented data. To overcome this issue, some studies have proposed the use of Generative Adversarial Networks (GANs) for data augmentation in medical images. For instance, Gopinath et al. [9] proposed a GAN-based data augmentation method for retinal vessel segmentation, which improved the segmentation accuracy compared to traditional methods. Wu et al. [10] proposed a GAN-based data augmentation method for pulmonary nodule detection from CT images, which also outperformed traditional techniques. Specifically, Rana et al. [11] proposed a GAN-based data augmentation method for imbalanced blood cell image classification, while Bailo et al. [12] utilized conditional GANs for generating red blood cell images for data augmentation.

Drawing inspiration from these previous works, we recognize that while CNN-based models have achieved success in WBC classification, data augmentation remains a significant challenge, especially for small and imbalanced datasets such as the BCCD. GAN-based data augmentation has shown promising results in other medical image analysis tasks, and thus, in this work, we propose that it could be a promising approach for improving WBC classification accuracy on the BCCD dataset, beyond just employing traditional augmentation techniques.

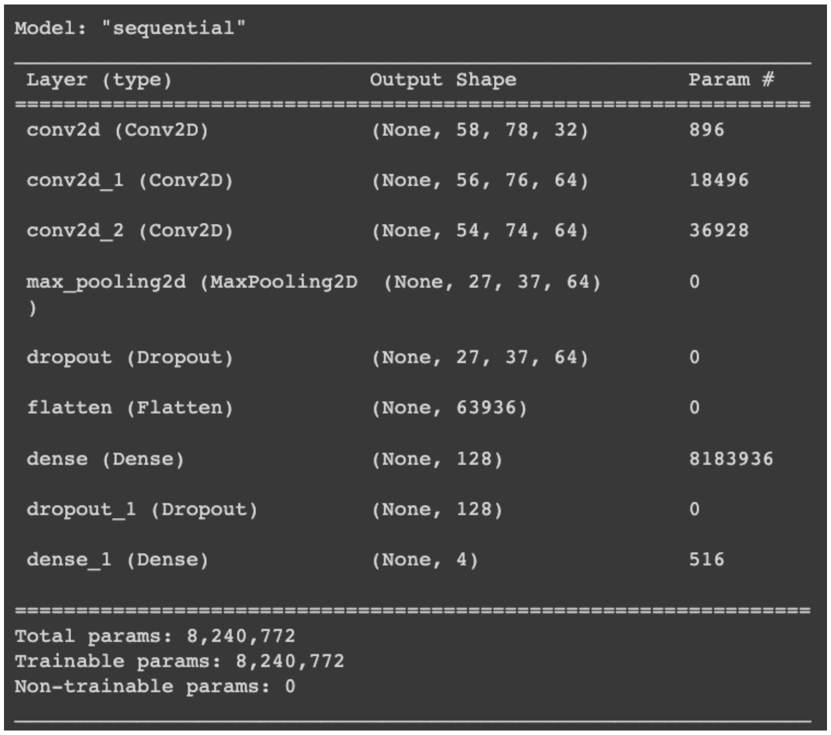
3 Method

3.1 Stage 1: Baseline Establishment

A classifier is built using Deep CNN & CNN-RNN frameworks, providing a baseline for the study. This will give a clear understanding of the performance of the classifier when trained on the original dataset. Deep Convolutional Neural Network (CNN) has shown good results in the classification of medical images. We further propose the CNN-RNN framework, which combines the strengths of both networks to better comprehend the image content and identify structured features.

3.1.1 Custom CNN Architecture

The CNN architecture consists of an input layer followed by three convolutional layers with increasing number of filters (32, 64, 64) and kernel sizes of 3x3. Each convolutional layer uses the ReLU activation function to introduce non-linearity in the model. The output of the third convolutional layer is fed into a max-pooling layer to reduce the spatial size of the output. A dropout layer with a rate of 0.25 is then added to prevent overfitting. The output is then flattened and fed into a dense layer with 128 units and ReLU activation function. Another dropout layer with a rate of 0.5 is added before the final dense layer with the number of units equal to the number of classes in the dataset. The softmax activation function is used to output probabilities for each class. The model is trained with the categorical cross-entropy loss function, the Adam optimizer, and accuracy as the evaluation metric. The architecture is shown below in Figure 1.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 58, 78, 32)        896

 conv2d_1 (Conv2D)           (None, 56, 76, 64)        18496

 conv2d_2 (Conv2D)           (None, 54, 74, 64)        36928

 max_pooling2d (MaxPooling2D  (None, 27, 37, 64)       0
 )

 dropout (Dropout)           (None, 27, 37, 64)        0

 flatten (Flatten)           (None, 63936)             0

 dense (Dense)               (None, 128)               8183936

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 8,240,772
Trainable params: 8,240,772
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: CNN Model

3.1.2 Custom CNN-RNN Architecture

The CNN-RNN architecture is a combination of CNN and RNN layers to classify white blood cell images based on fixed low level and high level features in sequence. The CNN layer identifies features such as lines, curves, and nuclear shapes, while the RNN layer takes advantage of the predictable feature hierarchy in these images.

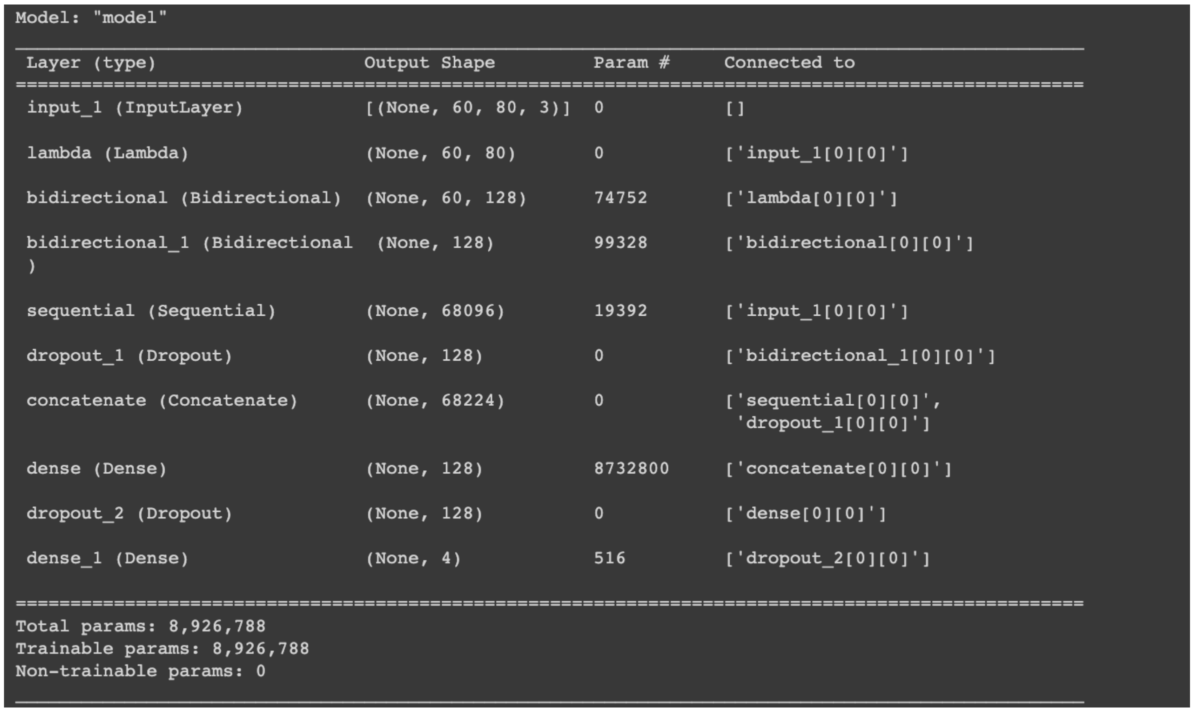
Specifically, this architecture comprises of five layers: Input Layer, CNN Layer, RNN Layer, Merge Layer, and Output Layer. The Input Layer receives image data and converts it into a tensor of appropriate size. The CNN Layer contains two convolutional layers with 32 and 64 filters and uses a 3x3 kernel. It is followed by pooling, dropout, and a 2D flattening layer. The RNN Layer has two LSTM layers, each with 64 LSTMs, that detect recurring features in grayscale images to reduce overfitting. The Merge Layer takes the elementwise product of the CNN and RNN layer outputs and contains a 128 cell layer with ReLU activation followed by a dropout layer. The Output Layer uses the softmax function to evaluate which class the given input image falls into. This architecture is depicted in Figure 2.

3.2 Stage 2: Data Expansion

In Stage 2 section of this work, the dataset will be expanded using conventional and GAN-based data augmentation techniques, which will increase the size and diversity of the data set. We experiment on both traditional augmentation techniques as well our custom GAN-generated synthetic images to add to our dataset. The details of both these methods are explained below.

3.2.1 Traditional Augmentation

In this work, we employed data augmentation techniques to expand the size and diversity of our BCCD dataset. Data augmentation is a widely used technique to address the issue of limited annotated data. We first explored several traditional augmentation techniques, including rotation, translation, and flipping, using the Keras ImageDataGenerator function.



```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 60, 80, 3)]  0           []

 lambda (Lambda)                (None, 60, 80)       0           ['input_1[0][0]']

 bidirectional (Bidirectional)  (None, 60, 128)      74752       ['lambda[0][0]']

 bidirectional_1 (Bidirectional  (None, 128)         99328       ['bidirectional[0][0]']
 )

 sequential (Sequential)        (None, 68096)        19392       ['input_1[0][0]']

 dropout_1 (Dropout)            (None, 128)          0           ['bidirectional_1[0][0]']

 concatenate (Concatenate)      (None, 68224)        0           ['sequential[0][0]',
                                                                  'dropout_1[0][0]']

 dense (Dense)                  (None, 128)          8732800     ['concatenate[0][0]']

 dropout_2 (Dropout)            (None, 128)          0           ['dense[0][0]']

 dense_1 (Dense)                (None, 4)            516         ['dropout_2[0][0]']

==================================================================================================
Total params: 8,926,788
Trainable params: 8,926,788
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 2: CNN-RNN Model

We experimented with various augmentation parameters, such as rotation range, width shift range, and height shift range, among others. After several trial runs, we observed that only the horizontal flip and rotation range were sufficient to generate diverse training data that could improve the model's performance for the CNN and CNN-RNN framework built in the previous section without overfitting or affecting model performance.

Here, we randomly rotate images within 10 degrees and randomly flip them horizontally. We did not perform vertical flipping as the orientation of the blood cells in the images remains the same regardless of whether the image is flipped vertically or not.

We also experimented with other augmentation techniques, but we found that these techniques did not add much diversity to the dataset. Therefore, we decided to only use horizontal flip and rotation range for our final training data.

3.2.2 Custom Generative Adversarial Network

Generative Adversarial Networks (GANs) are deep learning models that can generate synthetic data that resembles the training data. In a GAN, two models are trained simultaneously: the generator model and the discriminator model. The generator takes random noise as input and generates synthetic samples that are then fed into the discriminator along with real samples from the training data. The discriminator tries to distinguish between real and fake samples. The generator is trained to produce samples that can fool the discriminator. The two models are trained in an adversarial manner to improve their performance. [13]

Based on the above principle, we built a custom Generative Adversarial Network (GAN) to generate synthetic images of different types of White Blood Cells. This Custom GANs consist of 1 generator and 1 discriminator that are trained simultaneously in an adversarial manner. We used a simple variant of a Vanilla GAN model with 1 Generator and 1 Discriminator due to limited computational resources(rather than a better and more complex one like Conditional or DCGAN). The images from the original dataset were downsized to (256, 256) to prevent RAM crashes than trying to generate and use higher original sizes.

The generator network takes random noise as input and generates synthetic images that resemble the real images. The discriminator network is trained to distinguish between real and fake images. The generator starts with a Dense layer that maps the input noise vector to a large feature map, which is then reshaped and upsampled using UpSampling2D layers. The model then applies two Conv2D layers with BatchNormalization and ReLU activations to produce higher-resolution feature maps. Finally, the generator uses a Conv2D layer with a tanh activation to output the generated image. The discriminator uses several Conv2D layers with LeakyReLU activations, followed by Dropout and BatchNormalization layers to prevent overfitting.

Both networks were built using Keras Sequential API and optimized with binary cross-entropy loss using the Adam optimizer with a learning rate of 1e-4.

Overall, the generator architecture learns to transform the random noise vector into a high-dimensional representation that can be used to generate a realistic image using convolutional layers and up-sampling layers while introducing some randomness to produce new and diverse images.

Hyperparameters: latent dimension size is set to 100, the dropout rate is set to 0.25, and the learning rate is set to 1e-4 for the optimizer.

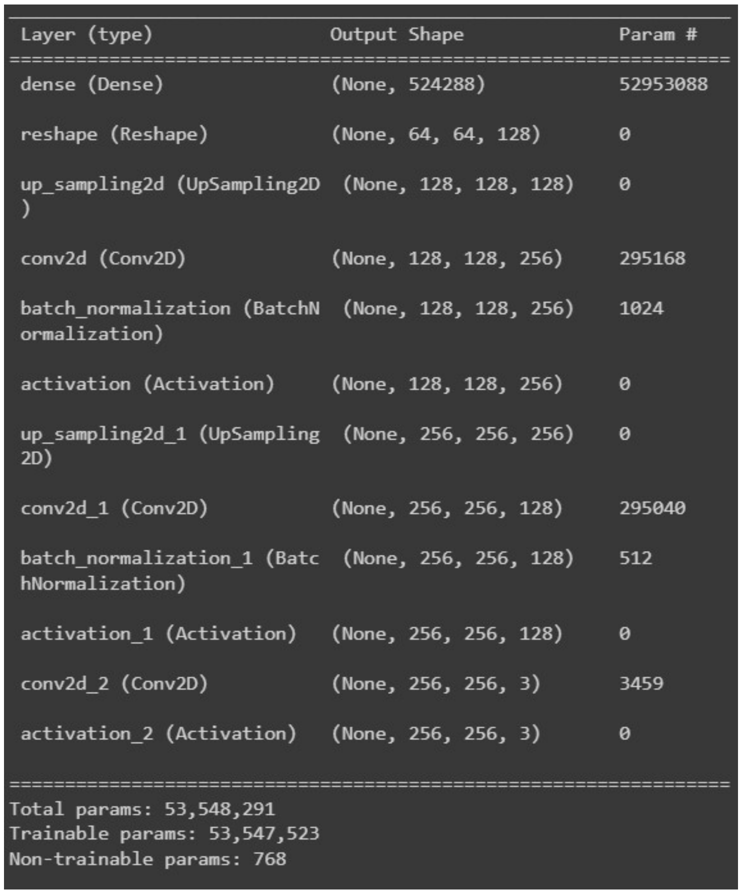
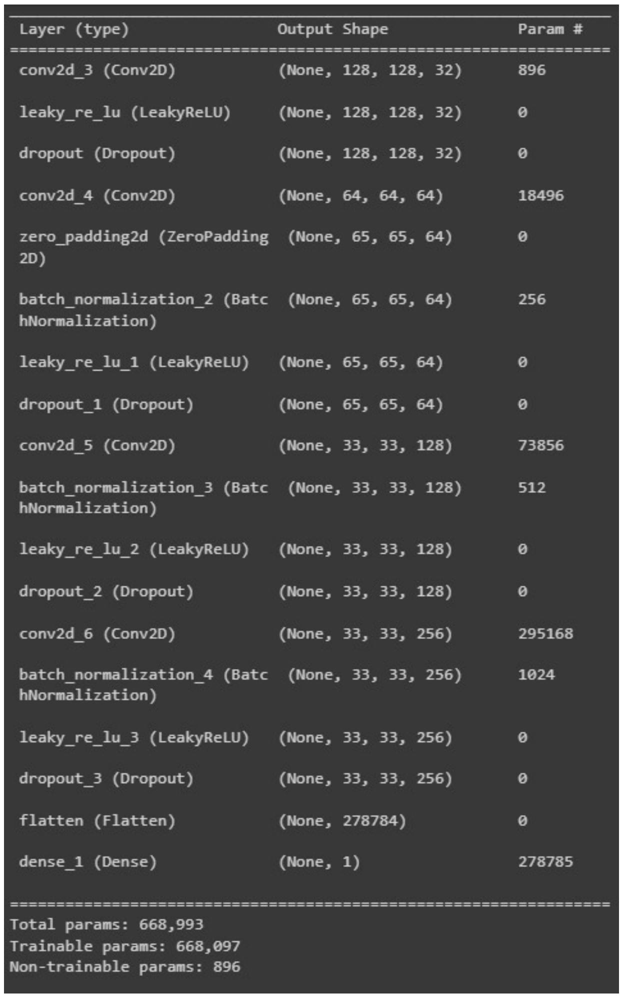

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 524288)            52953088

 reshape (Reshape)           (None, 64, 64, 128)       0

 up_sampling2d (UpSampling2D  (None, 128, 128, 128)    0
 )

 conv2d (Conv2D)             (None, 128, 128, 256)     295168

 batch_normalization (BatchN  (None, 128, 128, 256)    1024
 ormalization)

 activation (Activation)     (None, 128, 128, 256)     0

 up_sampling2d_1 (UpSampling  (None, 256, 256, 256)    0
 2D)

 conv2d_1 (Conv2D)           (None, 256, 256, 128)     295040

 batch_normalization_1 (Batc  (None, 256, 256, 128)    512
 hNormalization)

 activation_1 (Activation)   (None, 256, 256, 128)     0

 conv2d_2 (Conv2D)           (None, 256, 256, 3)       3459

 activation_2 (Activation)   (None, 256, 256, 3)       0

=================================================================
Total params: 53,548,291
Trainable params: 53,547,523
Non-trainable params: 768
```

Figure 3: Generator Model

The generator and discriminator, architecture model summary, are given below in Fig 3, and Fig 4, respectively.

3.3 Training and Testing

To prepare for training and testing, we divided the Image Dataset into an 85% training set and a 15% testing set. The training set was further split into an 85% training subset and a 15% validation subset.



```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 128, 128, 32)      896

 leaky_re_lu (LeakyReLU)     (None, 128, 128, 32)      0

 dropout (Dropout)           (None, 128, 128, 32)      0

 conv2d_4 (Conv2D)           (None, 64, 64, 64)        18496

 zero_padding2d (ZeroPadding  (None, 65, 65, 64)       0
 2D)

 batch_normalization_2 (Batc  (None, 65, 65, 64)       256
 hNormalization)

 leaky_re_lu_1 (LeakyReLU)   (None, 65, 65, 64)        0

 dropout_1 (Dropout)         (None, 65, 65, 64)        0

 conv2d_5 (Conv2D)           (None, 33, 33, 128)       73856

 batch_normalization_3 (Batc  (None, 33, 33, 128)      512
 hNormalization)

 leaky_re_lu_2 (LeakyReLU)   (None, 33, 33, 128)       0

 dropout_2 (Dropout)         (None, 33, 33, 128)       0

 conv2d_6 (Conv2D)           (None, 33, 33, 256)       295168

 batch_normalization_4 (Batc  (None, 33, 33, 256)      1024
 hNormalization)

 leaky_re_lu_3 (LeakyReLU)   (None, 33, 33, 256)       0

 dropout_3 (Dropout)         (None, 33, 33, 256)       0

 flatten (Flatten)           (None, 278784)            0

 dense_1 (Dense)             (None, 1)                 278785

=================================================================
Total params: 668,993
Trainable params: 668,097
Non-trainable params: 896
```

Figure 4: Discriminator Model

We conducted a grid search on hyperparameters such as the number of filters in the convolutional layers, kernel sizes, learning rate of the Adam optimizer, and dropout rates to prevent overfitting. After experimentation, we established a fixed set of hyperparameters for both the CNN architecture and the CNN-RNN architecture.

All models were trained for a maximum of 50 epochs, and the best model was selected based on its validation accuracy before being tested on the testing set. We evaluated the performance of the tuned models on the training and testing set using a classification matrix, which provided class-wise reports of accuracy, precision, recall, and F1-score for each model. Additionally, we analyzed the confusion matrix of the models to identify commonly misclassified classes and determine the reasons for misclassification. Details of the same will be covered in the upcoming section. Finally, we plotted graphs of accuracy and loss vs. epochs to gain insights into how the training and validation sets performed in relation to each other and assess the possibility of overfitting.

Ablation studies were performed using both the CNN and CNN-RNN Architectures with varying augmentation techniques.

3.4 Novelty and Strengths

The proposed method for this work demonstrates a primary novelty and strength by utilizing Generative Adversarial Networks (GANs) to generate synthetic images to supplement the low and imbalanced dataset. This approach is novel in the BCCD dataset, where limited work has been done on using GANs for data augmentation and classification. Adding GANs for augmentation along with traditional techniques is hoped to boost the classification performance by using realistic-looking synthetic data pertaining to individual classes.

Furthermore, the base CNN and CNN-RNN architecture for this work has been adapted from [3]. However, the model has been improved and reimplemented by using additional convolutional and dropout layers and experimenting with different layers to obtain the best possible results.

The use of GANs for data augmentation combined with the modifications made to the base architecture results in improved performance for the proposed model. The model's performance is demonstrated through evaluation metrics such as accuracy, precision, recall, F1-score, and a comparative analysis with baseline models established in the initial stage of the work and will be discussed further in the next sections.

4 Experiments

4.1 Dataset

We will be using the original Blood Cell Count and Detection (BCCD) Dataset [14] [15]. It is a small-scale dataset consisting of 373 white blood cell images grouped into 4 different cell types, namely Eosinophil, Lymphocyte, Monocyte, Neutrophil and Basophil. We do not consider Basophil class for the classification task as the original dataset has very limited images (only 4) for this class.

Table 1: Data distribution of Different Types of White Blood Cell Images

Type	**Eosinophil**	**Lymphocyte**	**Monocyte**	**Neutrophil**
Total Images	94	37	23	215

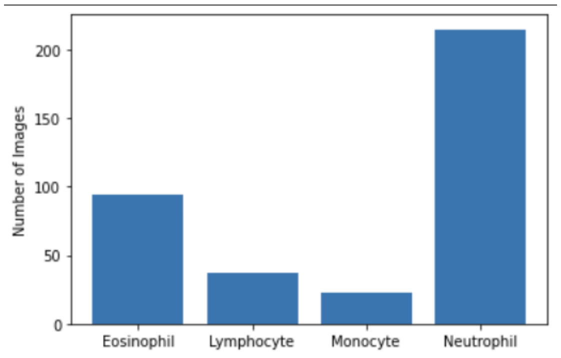


Figure 5: Graph depicting Imbalance in data set

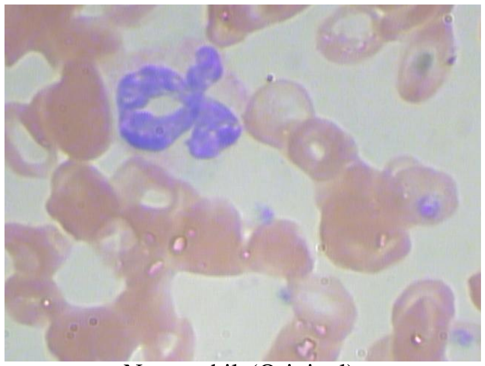
As we can see, the dataset is extremely small and highly imbalanced. This poses a major challenge to train deep learning models.

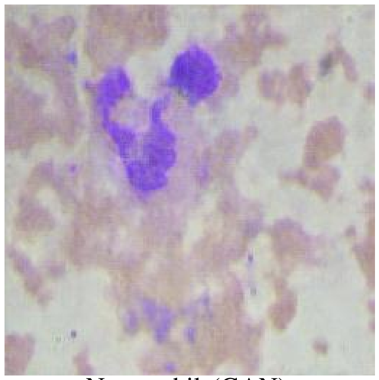
4.2 Augmented Data

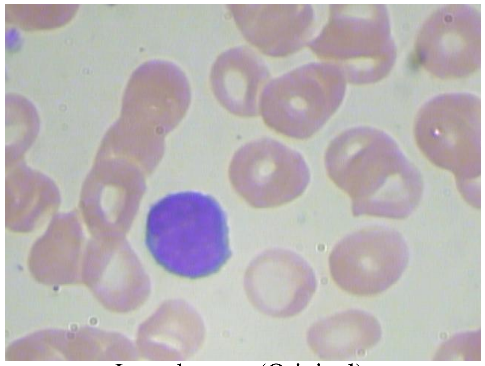
In the previous section, it was observed that the BCCD dataset suffers from severe class imbalance and a lack of diversity in smaller classes. Moreover, the dataset of only 369 images is not sufficient for any deep CNN models or larger/complex models to accurately differentiate between different classes based on their features without overfitting.

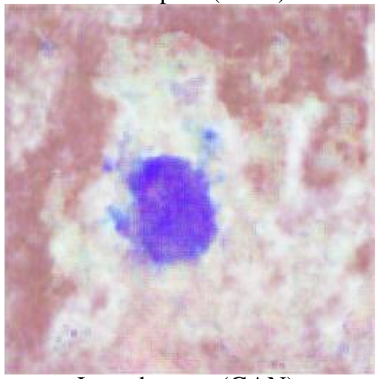
To address the issue of class imbalance, we used GAN-generated synthetic images to augment the dataset, adding additional images to each class to bring them up to a total of 300 images. With this augmentation, the new dataset now contains a balanced 1200 images, with each class having an equal number of images. This augmented dataset is now sufficiently large for both CNN and CNN-RNN models to learn the features of different classes without overfitting.

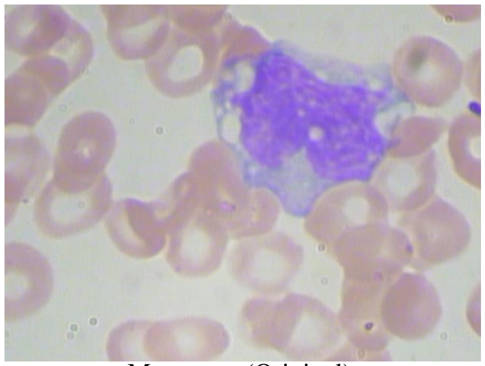
Therefore, this new augmented dataset of 1200 images serves as our primary dataset. The images below illustrate the comparison between real and GAN-generated images.

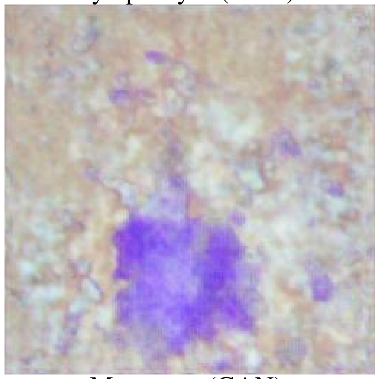

Neutrophils(Original)

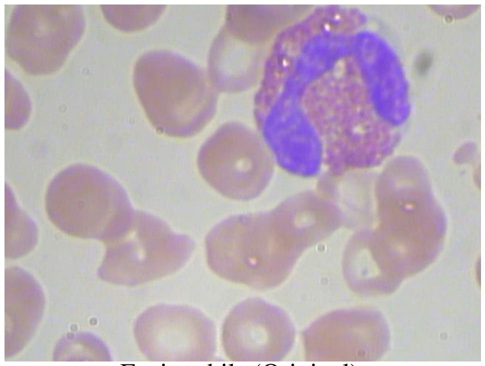

Neutrophils(GAN)

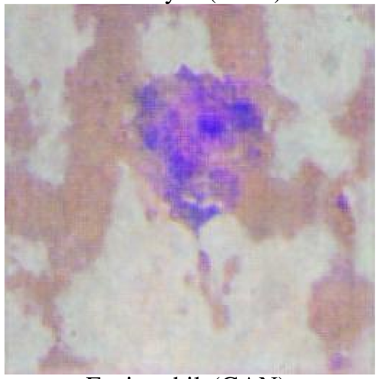

Lymphocytes(Original)


Lymphocytes(GAN)


Monocytes(Original)


Monocytes(GAN)


Eosinophils (Original)


Eosinophils(GAN)

4.3 Ablation Study

To evaluate the effectiveness of our proposed method, we conducted an ablation study on two models: CNN only and CNN+RNN. Four studies were performed on each of these models to determine the contribution of each augmentation technique to the performance of the model.

The four studies are as follows:

1. **Original data only:** This study used only the original dataset of 369 images without any augmentation techniques. This acts as our baseline, which we strive to improve upon using varying augmentation techniques.
2. **Original data + traditional augmentation techniques:** In this study, we used traditional inbuilt augmentation techniques described in section 3.2 to augment the original dataset.
3. **Augmented data (GAN generated + Original data):** This study used the GAN generated synthetic images in addition to the original dataset(1200 image dataset as described in section 4.2.
4. **Augmented data + traditional augmentation techniques:** In this study, we used both the GAN-generated synthetic images and traditional augmentation techniques to augment the original dataset.

For each study, the models were trained and tested on the same train-test split, and the classification metrics were evaluated.

The results and implications of the study are discussed below.

4.4 Results and Discussion

Table 2: Ablation Study

Type	CNN Train Acc	CNN Test Acc	CNN+RNN Train Acc	CNN+RNN Test Acc
OG Data Only	98.49	48.21	98.12	58.92
OG Data + Trad Aug	98.87	62.5	78.94	66.07
Augmented Data	99.76	83.33	99.76	87.22
Augmented + Trad Aug	98.84	86.67	96.30	87.77

The main results of the ablation study based on train and test accuracies are shown in Table 2. Four experiments were conducted on two models: CNN only and CNN+RNN. The experiments included using original data only, original data with traditional augmentation techniques augmented data (generated by GAN) only, and augmented data with traditional augmentation techniques.

The first experiment with original data only achieved high accuracy in training, but a significantly lower accuracy in testing for both CNN only (98.49% train acc and 48.21% test acc) and CNN+RNN (98.12% train acc and 58.92% test acc) models. This indicates the overfitting of the training data due to the small size and imbalance of the dataset. The same trend can be observed in the accuracy vs. epochs curves in Fig 6 and Fig 10.

The second experiment using original data with traditional augmentation techniques improved the test accuracy of the CNN-only model to 62.5%. Similarly, the test accuracy of the CNN+RNN model increased to 66.07%. This indicated using traditional augmentation techniques does help improve the performance and curb overfitting a little better. The same trend can be observed in the accuracy vs. epochs curves in Fig 7 and Fig 11.

The third experiment using augmented data only, generated by GAN, achieved high accuracy in both training and testing for both CNN only (99.76% train acc and 83.33% test acc) and CNN+RNN (99.76% train acc and 87.22% test acc) models in comparison to the previous two experiments. This shows that adding synthetic images generated by GAN can significantly improve the performance of the models. On observing the accuracy vs. epochs curves in Fig 8 and Fig 12, we can notice that overfitting has significantly reduced in comparison to the above 2 experiments. This shows the robustness and improvement of our model performance by using GANs to make data balanced and larger and diverse.

The fourth experiment using augmented data with traditional augmentation techniques achieved the highest test accuracy for both CNN only (86.67% test acc) and CNN+RNN (87.77% test acc) models.

This shows that employing ensemble augmentation techniques as a combination of traditional ones with state-of-the-art GANs can prove to perform best. (See Fig 9 and Fig 12)

Fig 14-29 shows the confusion and classification matrices computed class-wise for all the experiments. On studying them all again, the precision, recall, and F1 scores, overall, the results suggest again that using GAN-generated synthetic images can significantly improve the performance of both CNN-only and CNN+RNN models. Additionally, traditional augmentation techniques can also be used to further improve the performance of the models, but the effect may be more pronounced in the CNN-only model than in the CNN+RNN model.

Furthermore, the CNN+RNN model outperformed the CNN model in all studies, indicating that the addition of the RNN layer improves the classification performance of the model. The addition of an RNN layer to the CNN model allows the model to consider the temporal sequence of the input images, which can be helpful in distinguishing between different classes of blood cells. For example, certain types of cells may appear differently over time, and the RNN layer can help capture these changes.

In summary, the ablation study demonstrated the effectiveness of our proposed method, which uses GANs to generate synthetic images and traditional augmentation techniques to augment the original dataset to address the problem of class imbalance in the BCCD dataset. The results showed that the augmented dataset significantly improves the performance of the model, and the addition of the RNN layer further improves the classification performance.

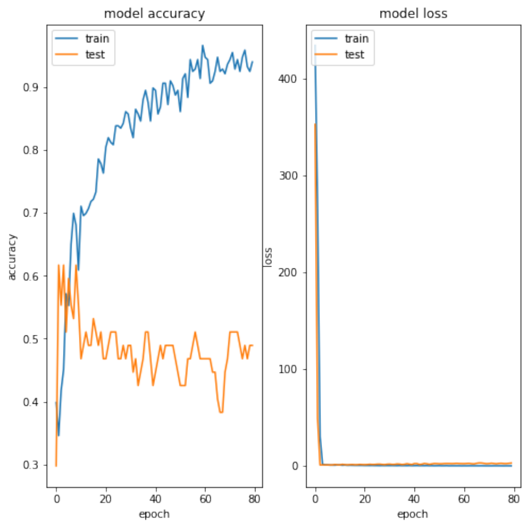


Fig 6: CNN (Original)

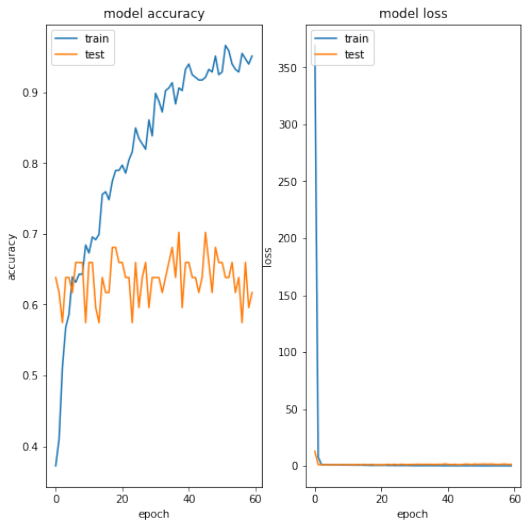


Fig 7: CNN (Original + Traditionally Augmented)

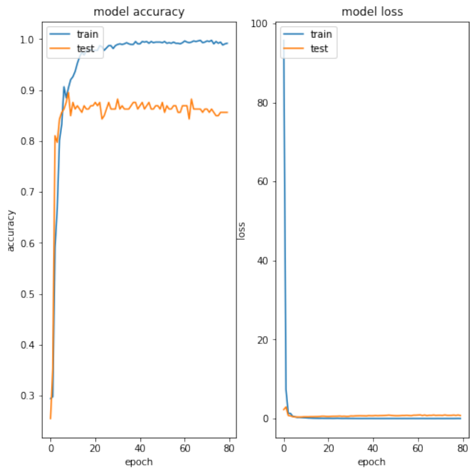


Fig 8: CNN (Original + GAN Augmented)

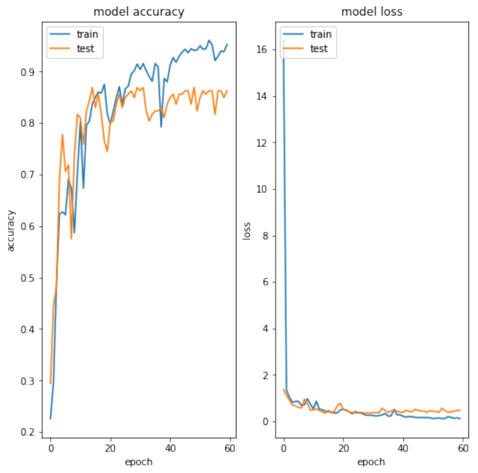


Fig 9: CNN (Original + Traditional + GAN Augmented)

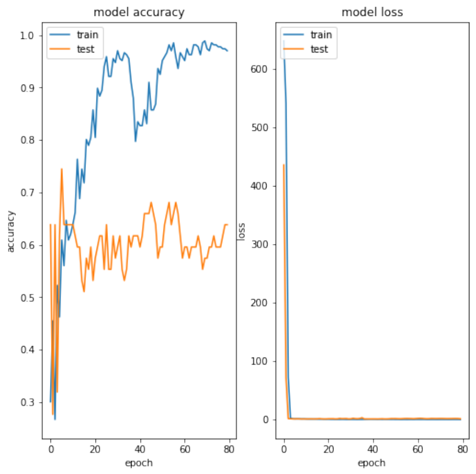


Fig 10: CNN-RNN (Original)

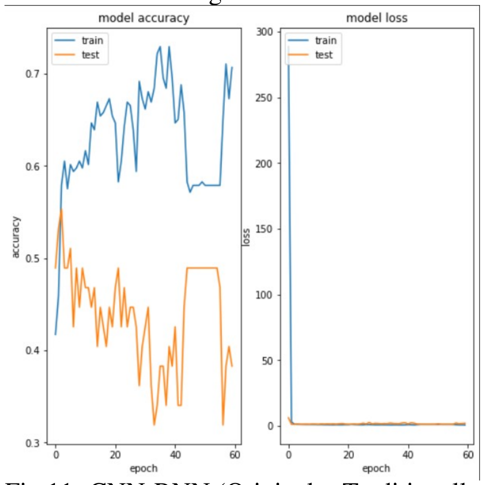


Fig 11: CNN-RNN (Original + Traditionally Augmented)

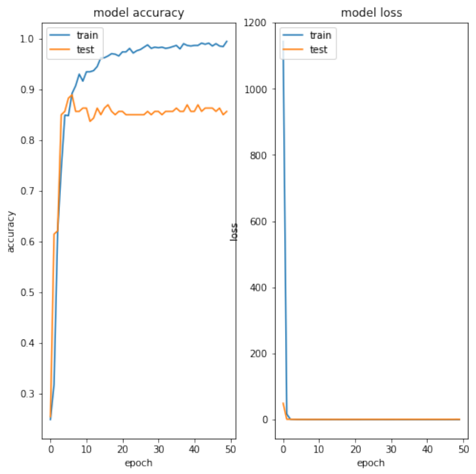


Fig 12: CNN-RNN (Original + GAN Augmented)

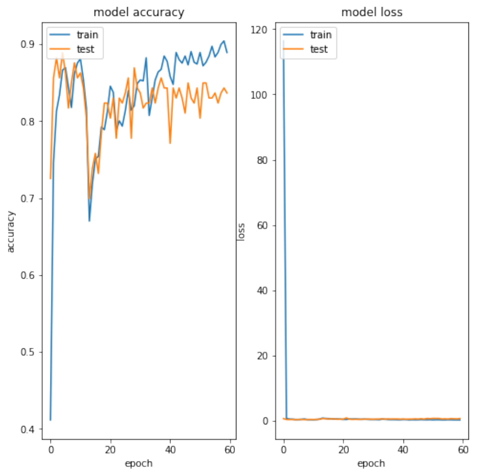


Fig 13: CNN-RNN (Original + Traditional + GAN Augmented)

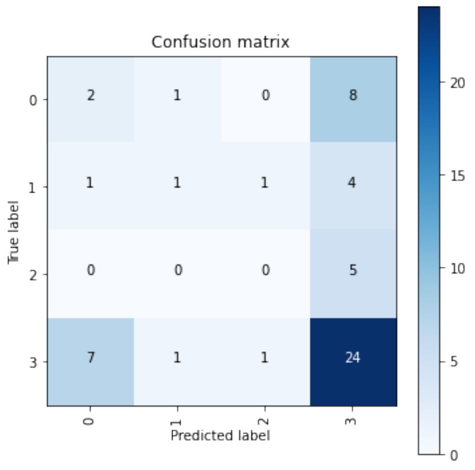


Fig 14: CNN (Original)

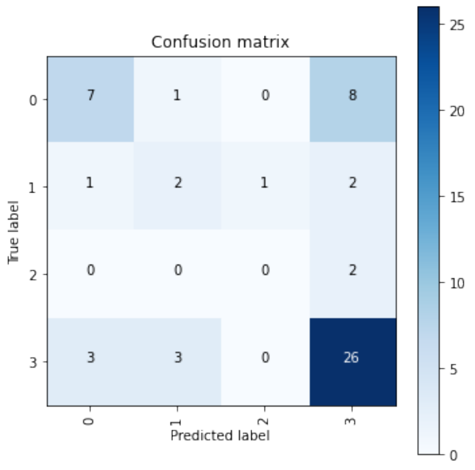


Fig 15: CNN (Original + Traditionally Augmented)

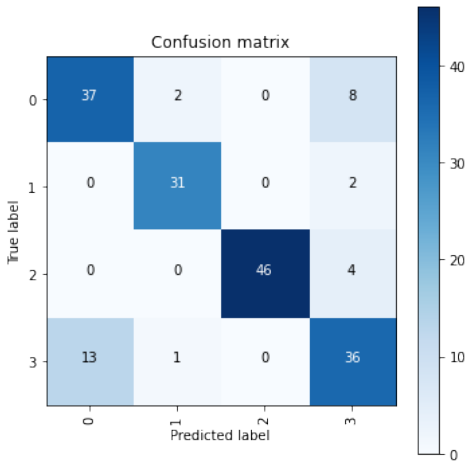


Fig 16: CNN (Original + GAN Augmented)

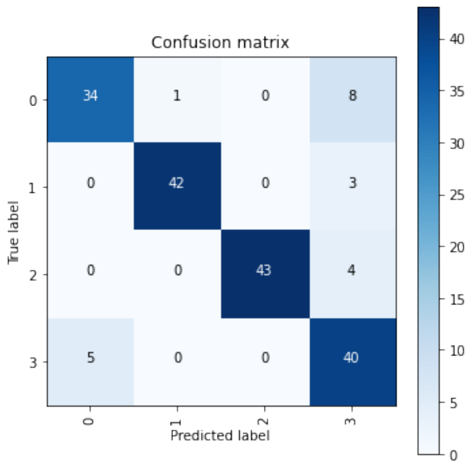


Fig 17: CNN (Original + Traditional + GAN Augmented)

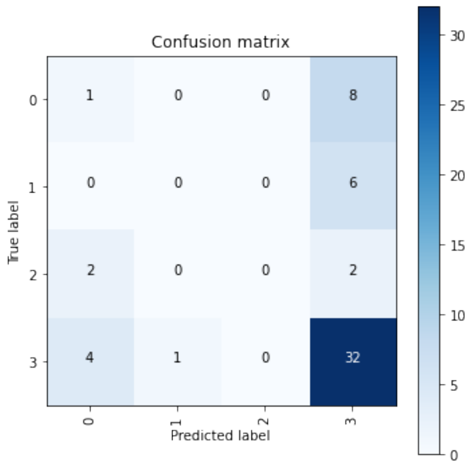


Fig 18: CNN-RNN (Original)

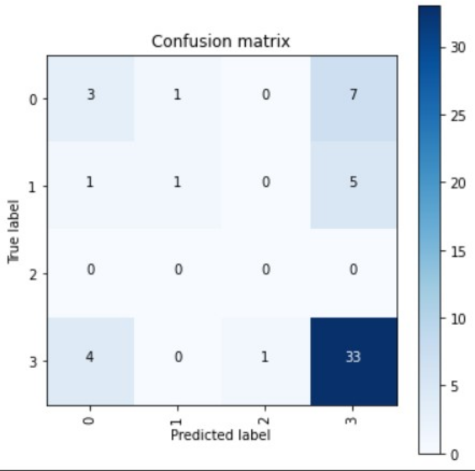


Fig 19: CNN-RNN (Original + Traditionally Augmented)

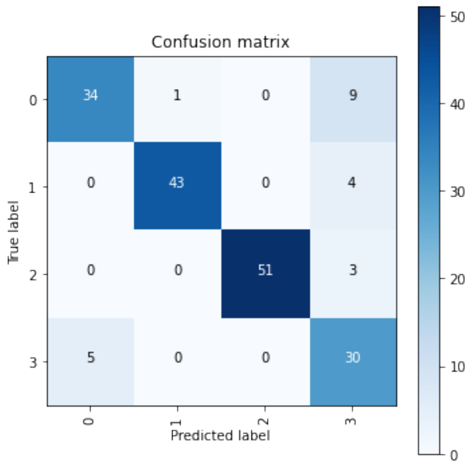


Fig 20: CNN-RNN (Original + GAN Augmented)

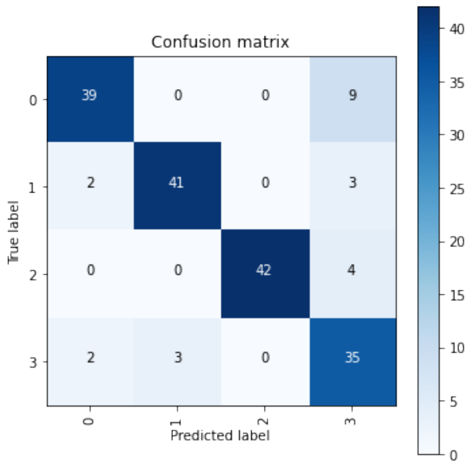
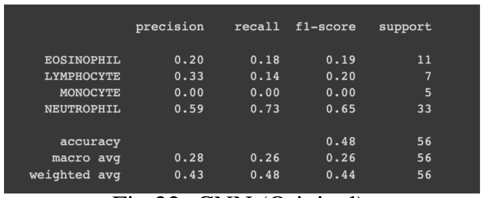
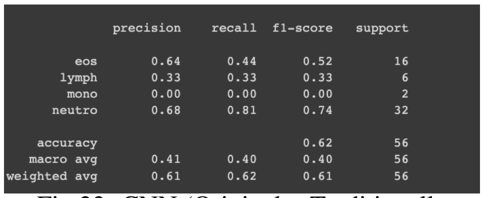


Fig 21: CNN-RNN (Original + Traditional + GAN Augmented)



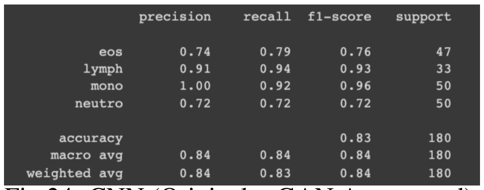
```
              precision    recall  f1-score   support

  EOSINOPHIL       0.20      0.18      0.19        11
  LYMPHOCYTE       0.33      0.14      0.20         7
    MONOCYTE       0.00      0.00      0.00         5
  NEUTROPHIL       0.59      0.73      0.65        33

    accuracy                           0.48        56
   macro avg       0.28      0.26      0.26        56
weighted avg       0.43      0.48      0.44        56
```

Fig 22: CNN (Original)



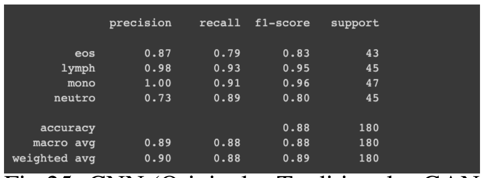
```
              precision    recall  f1-score   support

         eos       0.64      0.44      0.52        16
       lymph       0.33      0.33      0.33         6
        mono       0.00      0.00      0.00         2
      neutro       0.68      0.81      0.74        32

    accuracy                           0.62        56
   macro avg       0.41      0.40      0.40        56
weighted avg       0.61      0.62      0.61        56
```

Fig 23: CNN (Original + Traditionally Augmented)



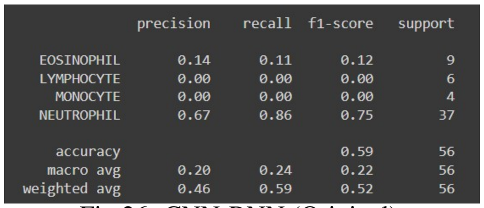
```
              precision    recall  f1-score   support

         eos       0.74      0.79      0.76        47
       lymph       0.91      0.94      0.93        33
        mono       1.00      0.92      0.96        50
      neutro       0.72      0.72      0.72        50

    accuracy                           0.83       180
   macro avg       0.84      0.84      0.84       180
weighted avg       0.84      0.83      0.84       180
```

Fig 24: CNN (Original + GAN Augmented)



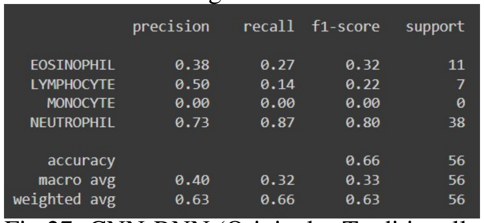
```
              precision    recall  f1-score   support

         eos       0.87      0.79      0.83        43
       lymph       0.98      0.93      0.95        45
        mono       1.00      0.91      0.96        47
      neutro       0.73      0.89      0.80        45

    accuracy                           0.88       180
   macro avg       0.89      0.88      0.88       180
weighted avg       0.90      0.88      0.89       180
```

Fig 25: CNN (Original + Traditional + GAN Augmented



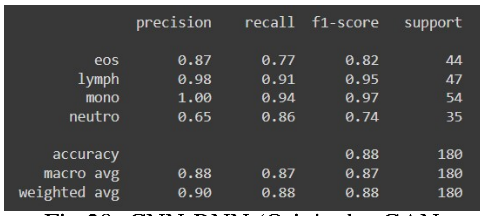
```
              precision    recall  f1-score   support

  EOSINOPHIL       0.14      0.11      0.12         9
  LYMPHOCYTE       0.00      0.00      0.00         6
    MONOCYTE       0.00      0.00      0.00         4
  NEUTROPHIL       0.67      0.86      0.75        37

    accuracy                           0.59        56
   macro avg       0.20      0.24      0.22        56
weighted avg       0.46      0.59      0.52        56
```

Fig 26: CNN-RNN (Original)



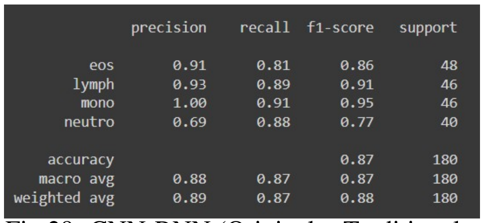
```
              precision    recall  f1-score   support

  EOSINOPHIL       0.38      0.27      0.32        11
  LYMPHOCYTE       0.50      0.14      0.22         7
    MONOCYTE       0.00      0.00      0.00         0
  NEUTROPHIL       0.73      0.87      0.80        38

    accuracy                           0.66        56
   macro avg       0.40      0.32      0.33        56
weighted avg       0.63      0.66      0.63        56
```

Fig 27: CNN-RNN (Original + Traditionally Augmented)



```
              precision    recall  f1-score   support

         eos       0.87      0.77      0.82        44
       lymph       0.98      0.91      0.95        47
        mono       1.00      0.94      0.97        54
      neutro       0.65      0.86      0.74        35

    accuracy                           0.88       180
   macro avg       0.88      0.87      0.87       180
weighted avg       0.90      0.88      0.88       180
```

Fig 28: CNN-RNN (Original + GAN Augmented)



```
              precision    recall  f1-score   support

         eos       0.91      0.81      0.86        48
       lymph       0.93      0.89      0.91        46
        mono       1.00      0.91      0.95        46
      neutro       0.69      0.88      0.77        40

    accuracy                           0.87       180
   macro avg       0.88      0.87      0.87       180
weighted avg       0.89      0.87      0.88       180
```

Fig 29: CNN-RNN (Original + Traditional + GAN Augmented)

5 Supplementary Material

Project Video: https://drive.google.com/file/d/1geho0TM-5LnXk6cMoR0FVPvuSo_UZUs1/view?usp=sharing

Presentation:https://docs.google.com/presentation/d/11vFe_HhKdY-ZML66eHmfnd9PCy-FuSGDT26qpfAMTRQ/edit?usp=sharing

References

[1] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, 42, 60-88.

[2] Shen, D., Wu, G., Suk, H. I. (2017). Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19, 221-248.

[3] Liang, G., Hong, H., Xie, W., Zheng, L. (2018). Combining Convolutional Neural Network With Recursive Neural Network for Blood Cell Image Classification. *IEEE Access*, 6, 36188-36197. doi: 10.1109/ACCESS.2018.2846685.

[4] Ghosh, I., Kundu, S. (2021). Combining Neural Network Models for Blood Cell Classification. arXiv preprint arXiv:2101.03604.

[5] Habibzadeh, M., Jannesari, M., Rezaei, Z., Baharvand, H., Totonchi, M. (2018). Automatic white blood cell classification using pre-trained deep learning models: ResNet and Inception. In Proceedings of the Tenth International Conference on Machine Vision (Vol. 10696, p. 1069612). SPIE. doi: 10.1117/12.2311282.

[6] Khamparia, A., Garg, N., Gupta, A. (2020). White blood cell classification using deep learning. *Journal of Ambient Intelligence and Humanized Computing*, 11(3), 1201-1213.

[7] Antun, V., Kuzmanić, Š., Šegvić, S. (2018). Data augmentation for skin lesion analysis. In *Proceedings of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics* (pp. 275-280). IEEE.

[8] Li, X., Chen, H., Qi, X., Dou, Q., Fu, C. W., Heng, P. A. (2020). Deep learning for understanding and deconvoluting data augmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9), 3268-3283.

[9] Gopinath, K., Krishnan, M., Babu, R. V. (2020). A novel GAN-based data augmentation for retinal vessel segmentation. *Neural Computing and Applications*, 32(23), 17017-17030.

[10] Wu, W., Gao, Y., Qi, J., Chen, X. (2019). GAN-based data augmentation for pulmonary nodule detection from CT images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 388-396). Springer, Cham.

[11] Rana, P., Sowmya, A., Meijering, E., et al. (2022). Data augmentation with improved regularization and sampling for imbalanced blood cell image classification. *Scientific Reports*, 12, 18101. doi: 10

[12] Bailo, O., Ham, D., & Shin, Y. M. (2019). Red blood cell image generation for data augmentation using conditional generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (pp. 0-0).

[13] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).

[14] Mooney, P. (2021). Blood Cells. Kaggle Datasets. https://www.kaggle.com/datasets/paultimothymooney/blood-cells

[15] Shenggan. (n.d.). BCCD_Dataset. https://github.com/Shenggan/BCCD_Dataset