A Thesis

For the Degree of Doctor of Philosophy

Few-Shot Image Generation and Novel Deep Learning Model for Enhancing Classification of Microscopic Single-Cell Images Obtained from Peripheral Blood Smears

Debapriya Hazra

Department of Computer Engineering
Graduate School
Jeju National University

June 2022

Few-Shot Image Generation and Novel Deep Learning Model for Enhancing Classification of Microscopic Single-Cell Images Obtained from Peripheral Blood Smears

Debapriya Hazra

(Supervised by Professor Yung-Cheol Byun)

A thesis submitted in partial fulfillment of the requirement for the degree of Doctor of Philosophy in Computer Engineering

2022.06.02

This thesis has been examined and approved.

[illegible]

Thesis Director

Sang-Joon Lee, Professor, Department of Computer Engineering

Dong Cheol, Lee

Dong-Cheol Lee, Professor, Department of Management Information

Namje Park

朴南濟印

Namje Park, Professor, Department of Computer Education

W. J. Kim

金眞佑印

Woo Jin Kim, Deputy Director, EONE Laboratories, Department of Laboratory Medicine

[illegible]

Thesis Supervisor

Yung-Cheol Byun, Professor, Department of Computer Engineering

June, 2022

Department of Computer Engineering

GRADUATE SCHOOL

JEJU NATIONAL UNIVERSITY

Acknowledgement

I wholeheartedly dedicate this milestone to my beloved father, Ramen Hazra, my mother, Sipra Hazra, and my brother Rohan Hazra. They have supported me unconditionally throughout my life in every way possible. I would like to thank Dr. Debnath Bhattacharyya, who has been my true source of inspiration to pursue my Ph.D. studies. I would use this opportunity to express my gratitude to all my relatives. I am lucky and blessed to have such a lovely family. I am grateful to Farda Abid, Dr. Wafa Shafqat, Dr. Sehrish Malik, Mudassar Liaq and Muhammad Sajjad for becoming my family abroad and being there for me through thick and thin.

I express my sincere appreciation to my supervisor, Dr. Yung-Cheol Byun, who convincingly encouraged me to be professional in achieving my goals and directed me to do the right thing even during my low times. Without his persistent help, the goal of completing this thesis wouldn't have been possible. I would like to acknowledge that he played a vital role in shaping my views towards maintaining a healthy balance between professional and personal life goals.

I would also like to thank Dr. Woo Jin Kim for supervising me on my Ph.D. thesis. He has constantly guided me to learn and improve my skills in the computational biology field. He has always been patient in answering every query in great detail.

I would like to thank Sana Salim, Madhuri Dey, and Subhajit Chatterjee for constantly encouraging me throughout this journey. They have been my constant support since the time we have known each other. Finally, I wish to appreciate the support of all my friends and colleagues here, who are not just important in my life but also crucial to the successful realization of this thesis; also expressing my apology that I could not mention them personally one by one.

I greatly appreciate the assistance and exposure provided by Jeju National University and its Computer Engineering Department in fulfilling my dreams.

Dedicated to

Maa, Baba and Dada
(Sipra Hazra, Ramen Hazra and Rohan Hazra)

Table of Contents

List of Figures

List of Tables

Abstract

The human body is composed of 1.2 to 1.5 gallons of blood which is approximately 7%-10% of the weight of an adult. The blood constitutes of plasma and blood cells, and the ratio is 55% and 45%. The plasma in the blood is responsible for carrying proteins, hormones and nutrients that help the blood clot and remove waste components from the human body. The blood cells in the body carry oxygen to the tissues, fight infections and also helps blood to clot. An excessive amount of blood cells or deficiency of blood cells can indicate various health problems like anaemia, leukaemia, thalassemia, etc. The most common form of malignancy in kids is leukaemia which represents 30% of all pediatric cancers. Classification of cells procured from bone marrow aspirate smears and cell differential count is important for hematologic disease diagnosis. But the process of classification and cell counting requires thorough manual intervention and is error-prone and tedious. Machine learning and deep learning have successfully generated accurate and excellent results in the medical domain, especially for automating the medical field's diagnosis process. The only drawback is the requirement of ample data and a balanced dataset to develop an efficient model. The structure and pattern of data also play an important role in enhancing the performance of a model.

In our research work, we take advantage of generative adversarial networks (GAN) and deep learning models to enhance the classification of microscopic single-cell images obtained from peripheral blood smears. To enhance classification, our primary goal was to balance and normalize the dataset. We use GAN for three reason in our proposed work. First, we utilize GAN for stain normalization. Second, we use GAN to generate synthetic images of blood cells that contained more than thousand images but less than six thousand images, per cell type in the dataset. Third, for cell types having less than hundred images we propose few-shot image generation based on

GAN architecture and meta-learning framework. We combine the original and the synthetic data to form a balanced dataset. After stain normalization and image generation, we obtain a balanced and normalized dataset which is used by the classification model. For classification, we proposed a novel deep learning model, SENet-154-GE which uses the original data and the balanced dataset individually to demonstrate how normalizing and balancing a dataset through proper models and algorithms can enhance the performance of a classification model and improve the classification accuracy.

For stain normalization, we have proposed a modified version of cycle consistency GAN (CycleGAN). We have incorporated Wasserstein GAN with gradient penalty (WGAN-GP) for the CycleGAN training and concentrated on the adversarial loss, cycle consistency loss and identity loss for building our model. Also, for the generator, instead of an encoder, transformer, and decoder network, we have used a U-Net architecture. For generating images of cell types with a moderate number of images, we have built a three-network GAN architecture consisting of a classifier, generator and discriminator. We named the model C-WGAN-GP since it's a classifier-based GAN architecture which incorporates the loss function of WGAN-GP. For cell types with very less images, we have proposed a meta learning based GAN framework that is trained on a larger dataset and, through learning the parameters, can generate images with fewer examples. We have incorporated the squeeze-and-excitation networks (SE) into the aggregated residual transformations (ResNeXt) architecture. We have implemented SENet-154 model and combined the gather excite model with SENet-154 for crucial and detailed feature extraction. We named the novel deep learning classification model SENet-154-GE.

For evaluation, we have used various evaluation metrics such as structural similarity index measurement (SSIM), Frechet inception distance (FID), and inception score (IS) for stain normalization. FID, precision, recall, F1-score, SSIM, L1 and L2 error, IS and learned perceptual image patch similarity (LPIPS) were used for evaluating C-WGAN-GP. We used FID, LPIPS and IS for evaluating few-shot image generation. We assessed the performance of the SENet-154-GE classification model through accuracy, specificity and sensitivity. We performed an ablation study to demonstrate the importance of each module in our proposed work, and the results show that our proposed approach can significantly enhance the performance of classifying microscopic single-cell images obtained from peripheral blood smears.

Chapter 1: Introduction

The peripheral blood smear is a procedure where count of various circulating blood cells is taken by viewing a blood sample under a microscope. This procedure helps in checking or detecting any abnormalities in the count or appearance of the blood cells. Blood is responsible for carrying nourishment, hormones, antibodies, electrolytes, vitamins, heat, immune cells and oxygen to the body tissues. The blood of the human body consists of plasma and blood cells. Blood cells can be subcategorized into red blood cells, white blood cells and platelets. Plasma is the liquid component of the blood that makes around 55% of the content of the blood [1]. The rest, 45%, constitutes the red blood cells, white blood cells and platelets [1]. Red blood cells are responsible for carrying oxygen from the lungs to the rest of the body parts. Red blood cell is also known as erythrocytes. White blood cells or leukocytes are responsible for fighting infection and help in building immunity against the infections. Platelets are responsible for blood clotting. These blood cells are prepared or made in the bone marrow of a human being. The bone marrow is known as the spongy tissue present inside our bone and contains stem cells that creates all types of blood cells. The creation, development and production of the new blood cells in the bone marrow is known as hematopoiesis.

Peripheral blood smear is a test to detect problems in the human body's red blood cells, white blood cells or platelets. Healthcare professionals take a sample of blood from the vein of the patient's arm. Special stains are then used on the blood sample, and the sample is examined under a microscope to observe the size, shape, number and if any abnormalities are present in the cells. First, a complete blood count (CBC) test is performed, and if there is any abnormality in the blood cell count, a peripheral blood smear test is needed to diagnose further. A peripheral blood smear test also helps determine the type of infection by identifying the types of white blood cells. There are various other reasons why a peripheral blood smear test is done. For example, it explains the reason for unintentional weight loss, can provide information about the cause for excess bleeding and bruising, to find the reason for low platelet count, to detect and diagnose blood disorders, to evaluate the cause of bone pain or enlargement of the lymph nodes, spleen or liver [2]. A blood smear can provide the information about count of each type of blood cell, the size of the cells, shape as well as differences in the sizes, etc. In Table 1, we have provided the names of some of the disorders related to each cell type, i.e. red blood cells, white blood cells and platelets.

Table 1: Examples of disorders related to each cell type

Red Blood Cells	**White Blood Cells**	**Platelets**
Iron-Deficiency Anemia	Acute or Chronic Leukemia	Myeloproliferative Disorders
Sickle Cell Anemia	Lymphoma	Thrombocytopenia
Hemolytic Uremic Syndrome	HIV	Bernard Soulier Disease
Acute Bleeding	Hepatitis C	Glanzmann's Thrombasthenia
Maturation Disorders	Parasitic Infections	Hermansky Pudlak Syndrome
Inflammation	Fungal Infections	Jacobsen Syndrome
Neoplasia	Lymphoproliferative Diseases	Lowe Syndrome
Chronic Disease	Idiopathic Thrombocytopenia	TAR Syndrome
Marrow Damage	Autoimmune Neutropenia	TTP

In Figure 1, we present how a sample of blood taken for peripheral blood smear test looks under the microscope. Eosinophil, basophil, neutrophil, lymphocyte and monocyte are the subtypes of white blood cells. Erythrocyte is red blood cells in Figure 1.

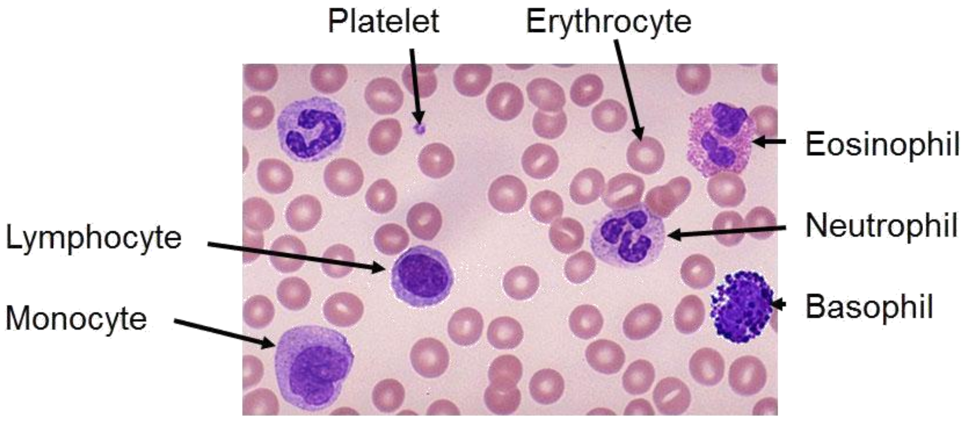

Figure 1: Sample of peripheral blood smear under a microscope [3]

Microscopic single-cell images refer to those images that contain an individual cell type, as shown in Figure 2. We have shown examples of a few cell types: basophil, eosinophil, erythroblast, immature granulocytes, lymphocyte, monoblast, monocyte and myeloblast, in Figure 2.

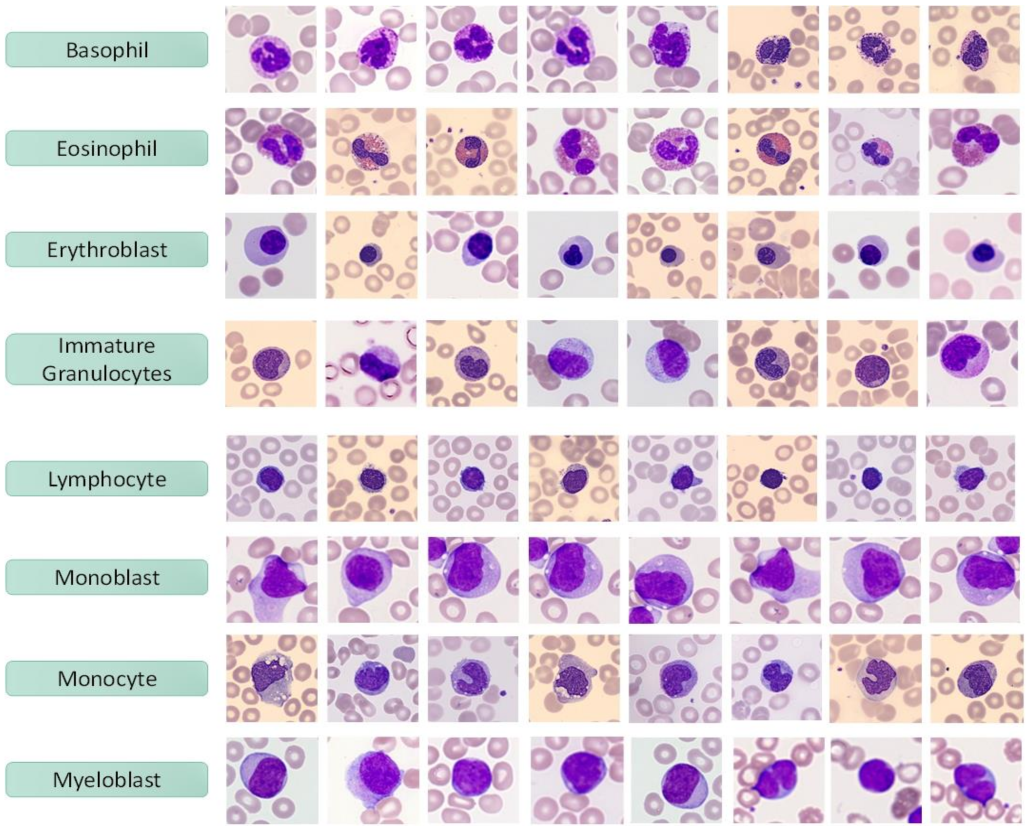

Figure 2: Examples of few cell types

Detection and diagnosis of blood related diseases or disorders require classification of these cell types, which is currently done manually, prone to error and requires a lot of time. Automatizing this process would help in the advancement of the medical domain and is a necessary requirement. Deep learning approaches have recently gained remarkable outcomes for medical image analysis, detection and classification. Machine learning and deep learning algorithms contributes in automatizing the process of diagnosis, accurate decision making and prognosis of a health problem which helps the doctors to find proper solutions. There has been rapid advancement in the field of medical imaging and analysis and also in the enhancement of classification outcomes. When handling medical data, there are certain constraints. Medical data are confidential and protected by patient data laws and these laws differ from country to country [4]. This makes the accessibility of medical data complicated and difficult. Machine learning or deep learning models require a large amount of data to train any model and to build an accurate and efficient model. But, in the medical domain, there is a problem of data scarcity. Several big research organizations and hospitals have made blood cell data anonymous and made them openly and freely available to the public, but they too are in less numbers [5]. Also, annotations of those data are done only once and show incorrectness and significant variations. Another issue with medical data is the data imbalance problem where data for one medical scenario exists in abundance, but there is very less data for the other medical scenario.

According to European Hematology Association (EHA), around 80 million people are currently affected by blood diseases or disorders in the European Union [6,7]. According to their reports, almost half of leukaemia, lymphoma or myeloma patients lose their lives due to inaccessible, incorrect or late diagnoses [6-7]. The average time for leukaemia diagnosis is estimated to be 14 days [6-7]. Machine learning has achieved significant outcomes in error reduction, fastening the diagnosis process, reducing the cost of care through reducing manual intervention, and providing contextual relevance to medical data and diagnosis. For data scarcity and imbalance problems, few-shot learning and generative adversarial networks (GAN) [8] have become famous for learning from small data and generating realistic synthetic data that would have no confidentiality problem and can be used for various research purposes. Data generated from GAN captures the distribution of the original data and can be used for developing real-time medical applications. Synthetic data generated by GAN has been fruitful in enhancing the classification of medical images and also in improving the process of automating medical image analysis.

In this thesis, we propose novel generative adversarial networks models and a novel deep learning model for enhancing the classification of microscopic single-cell images obtained from peripheral blood smears. We focus on four tasks in this thesis: data acquisition and preparation, stain normalization, data synthesis for solving data imbalance problem and classification. In the data acquisition and preparation phase, we collect peripheral blood cell images from different data sources and preprocess the data with the help of medical experts. Since we collected data from different data sources that used different stains for their experiments, we have performed stain normalization because different color intensities and multiple variations can affect the performance of any classifier. After stain normalization, we merge the dataset through expert intervention. Then we generate synthetic microscopic single-cell images using two generative adversarial networks model. We propose a three-network GAN architecture for generating images of the cell type containing more than thousand images. We propose a meta-learning based GAN architecture for few-shot image generation. We propose a few-shot image generation model for the cell types containing less than hundred images. We combine original and generated data to form a balanced and normalized dataset. We propose a novel deep learning model that is trained on the normalized and balanced dataset to classify microscopic single-cell images obtained from peripheral blood smears. The main contribution of this thesis is as follows:

- We collaborated with medical experts to form a new dataset. The new dataset was created by combining microscopic single-cell images obtained from two different sources. We preprocessed the data with expert intervention and also performed stain normalization before combining the datasets to form a single dataset.
- We have proposed a modified version of CycleGAN [9] for stain normalization.
- We proposed a classifier-based GAN architecture for generating images of the cell types containing more than thousand images but less than six thousand images.
- For cell types containing less than hundred images, we incorporated the meta-learning framework on the GAN architecture for few-shot image generation.
- We proposed a novel deep learning model, SENet-154-GE, for the classification of microscopic single-cell images obtained from peripheral blood smears.

The rest of the thesis is structured and organized in the following way: Chapter 2 presents the literature review of each module we proposed in our thesis. Chapter 3 gives details about the proposed methodology, its design and conceptualization. In chapter 4, we discuss the implementation results and performance analysis of each proposed module. Chapter 5 concludes the thesis along with a discussion.

Chapter 2: Literature Review

In this section, we present the literature review of relevant works on medical image classification, generative adversarial networks and few-shot image generation. The rapid technological advancement of artificial intelligence (AI) has achieved remarkable success in the medical domain. Machine learning brings new promises to clinical diagnosis and decision making through medical image analysis [10-12]. With millions of published research studies in the last decades, machine learning for medical image processing has become a trending subject of research [13]. In the healthcare domain, deep learning algorithms address a wide variety of problems, such as personalized treatment, disease monitoring, and cancer screening [14]. Deep learning in medical image analysis involves pattern recognition, image mining and computer vision such as segmentation, registration, detection and classification, etc. [15-16].

Enormous medical data reviewed by medical experts are now available on various data sources online, but the two most frequent problems with the medical datasets are imbalanced data collection and the lack of labeled or annotated data [17-19]. Not all labeled datasets are publicly available, and annotating data is a tedious task that requires manual intervention and is expensive as well as time-consuming. Supervised algorithms in machine learning require labeled datasets, but lately, the lack of labeled data has shifted the focus to semi-supervised, unsupervised or transfer learning models in machine learning for medical image analysis [20,21]. Machine learning consists of various types of learning that can be broadly categorized into four types and further subdivided into fourteen subtypes [22]. Figure 3 shows the types of learning in machine learning. We have classified the existing studies into five broad categories: generative adversarial networks, stain normalization, few-shot learning, few-shot image generation, and medical image analysis and classification.

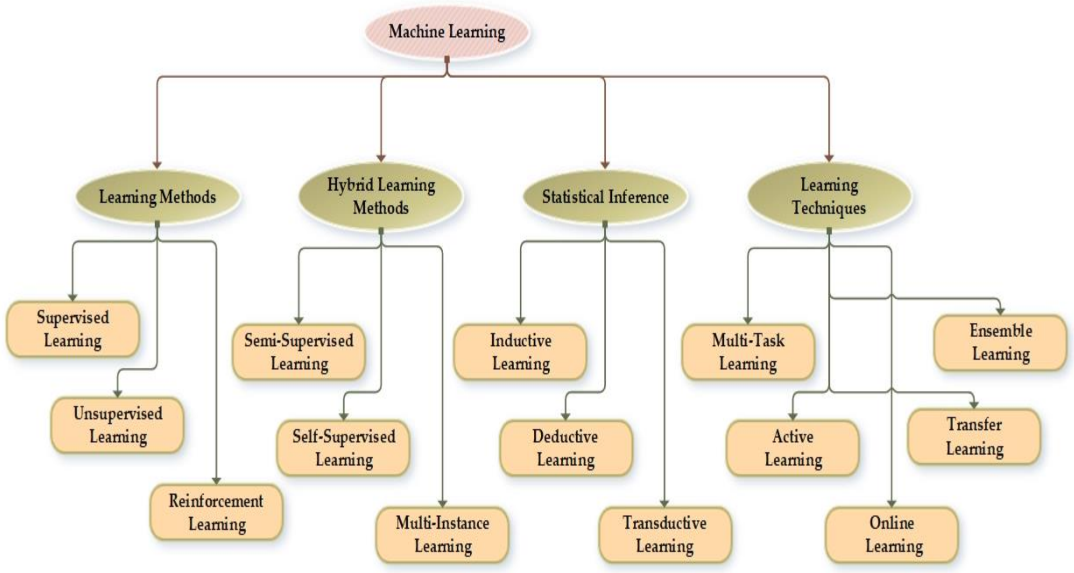

Figure 3: Types of learning in machine learning [22]

2.1 Generative Adversarial Networks (GAN)

Learning the hidden pattern or underlying structure of the data with no supervision is known as unsupervised learning. Examples of unsupervised learning are clustering, dimensionality reduction, feature reduction etc. Generative modeling is an unsupervised learning task that takes training samples as input from some distribution and learns to generate new samples that represent the same distribution. Generative modeling addresses density estimation which is a core problem in unsupervised learning. Figure 4. presents the taxonomy of generative models. Explicit density estimation assumes the prior distribution of the data and estimates the true probability density function over the sample space. Whereas implicit density estimation learns to model sample from the true distribution without explicitly defining it. Explicit density can be categorized into tractable and approximate density. Fully visible belief networks such as neural autoregressive distribution estimation (NADE) [23], masked autoencoder for distribution estimation (MADE) [24],

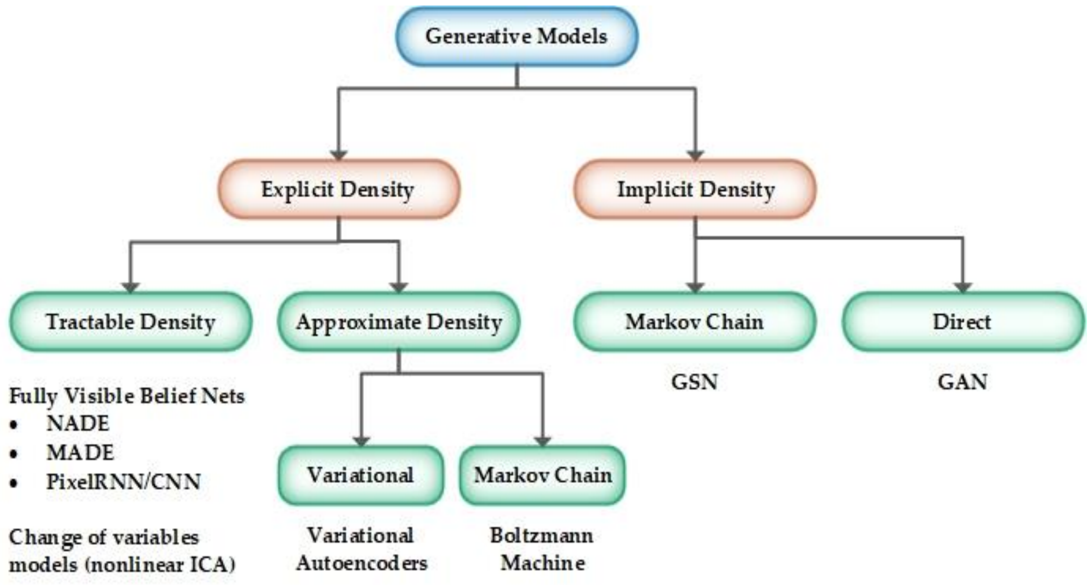

Figure 4: Taxonomy of generative models [25]

variational autoencoders, Boltzmann machine, PixelRNN [26] and PixelCNN [27] are some of the explicit density models. Generative stochastic networks and generative adversarial networks are forms of implicit density models. Generative Adversarial Networks (GAN) have demonstrated outstanding performance in image synthesis tasks; It is now possible to generate high-quality images with both fidelity and diversity because of the recent development of deep generative models. Although they often require many training examples to achieve a high-quality synthesis, to train a GAN from scratch for a new domain, we need many data to feed the generator and days of training time. Most deep generative models require many training images from a particular category to create new images for that category, which are sometimes prohibitively expensive to obtain. Given a limited amount of training data, the performance of generative adversarial networks (GAN) deteriorates dramatically. This is due to the discriminator having to memorize the exact training set. GANs are prone to overfitting without enough training data, resulting in mode collapse and training instabilities. GAN's application in domains where collecting a large dataset is not practicable is limited by its reliance on the availability of training data.

Pretrained GAN models are sometimes used to facilitate downstream tasks in a target domain with limited training data. This predicament usually develops as a result of costly data collection or privacy concerns in medical or biological applications [28]. Researchers concentrate on the difficult scenario of developing a GAN model with minimal training data. One of the important findings that inspired many research works is that a well-trained GAN may generate realistic images not seen in the training dataset [29-31], proving GANs' ability to generalize and capture the training data in a variety of ways. Likely arising from novel combinations of information/attributes/styles, StyleGAN [32] includes generalization of GANs for scenarios in which there are limited data. StyleGAN transfers style from one domain to another. GANs can be used to augment the training set via realistic data generation, alleviate overfitting or provide regularizations for classification [33,34], segmentation [35], or detection [36].

Recent efforts are centered on more stable objective functions [37-39], more advanced architectures [40-43] and better training strategy [44-46]. As a result, both the visual fidelity and diversity of generated images have increased significantly. For example, BigGAN [47] is able to synthesize natural images with a wide range of object classes at high resolution, and modified StyleGAN [48] can produce photorealistic face portraits with large varieties, often indistinguishable from natural images. However, the above work paid less attention to the data efficiency aspect. A recent attempt [49] leverages semi- and self-supervised learning to reduce the amount of human annotation required for training.

2.2 Stain Normalization

In computer-aided diagnosis, stain normalization of histopathology images is a promising technique. The effects of staining intensity and color difference are handled in various pathological imaging systems using this method. Deep learning has recently risen to prominence in the field of digital pathology image analysis. However, special pre-processing steps are required to train a valid deep learning model due to the present problems of digital pathology (color stain fluctuation, big pictures, etc.). Many of the medical phenomena now addressed in clinical medicine and pathology can be diagnosed via medical imaging. The quantitative analysis of histopathological images is influenced by color and intensity changes in stained histology slides. Stain normalization is a term used in biomedical image analysis to describe the process of transferring the color distribution from the source image to the target image. Stain normalization is typically accomplished using a pixel-by-pixel color mapping model that is dependent on a single reference image, and it is difficult to ensure precise style transformation between image datasets. Although deep learning-based algorithms can theoretically tackle this problem, their intricate structure leads to low computational efficiency and artefacts in the style transition, limiting its practical application.

To solve this well-defined problem of stain color fluctuations, stain normalizing methods have been developed. These approaches can be divided into two categories: traditional approaches, which attempts to match the image's color spectrum to that of the reference template image, and GAN based approach, which attempts to transfer style or color domain from one stain to another. Reinhard et al. [50] align the color channels to match the characteristics of the reference picture. However, because the same transformation is used across the photos and does not account for the individual contribution of stain dyes to the final color, this can result in incorrect color mapping. There are also stain-separation approaches that do normalization on each staining channel separately. Macenko et al. [51], for example, identify the stain vectors by converting RGB to Optical Density (OD) space. On the other hand, Khan et al. [52] provide a method for estimating the stain matrix that uses a color-based classifier to assign each pixel to the relevant stain component. According to Babak et al. [53], these approaches fail to account for the spatial aspects of the tissue structure, resulting in incorrect staining. Nonetheless, the majority of the methods in this class rely on a well-chosen reference template image, which has a significant impact on the methods' outcomes. The third type of solution is pure learning-based approaches, which treat stain normalization as a style-transfer problem. For example, BenTaieb et al. [54] use auxiliary Generative adversarial networks (auxGANs) with an auxiliary task on top, such as a classifier. They present a GANs-based method that not only eliminates the need for a reference image but also achieves high visual resemblance to the target domain, making it easier to remove stain variances and therefore enhancing the diagnosis process for both pathologists and CAD systems. Furthermore, unlike [55], stain type transfer does not necessitate being educated for a specific purpose.

Figure 5 [55] shows the framework, which is divided into two steps: StainGAN training, which is a generative adversarial networks model with two generators and two discriminators, and StainNet generation, which is a fully convolutional neural network. To learn the transition from the source color space to the target color space, StainNet requires matched source and target images. StainGAN is selected as the teacher network and StainNet as the student network because it is difficult to collect the paired photos and align them correctly in practice. That is, StainNet learns the output of StainGAN using the L1 loss. To achieve stain normalization, deep learning-based approaches primarily use generative adversarial networks (GANs) (55-60). To transfer the stain style, Shaban et al. (58) suggested StainGAN, as shown in Figure 6. It is an unsupervised stain normalizing approach based on CycleGAN. Cai et al. (56) developed a novel generator that would increase image quality and network speed. Cho et al. (60), Salehi et al. (57), and Tellez et al. (59), on the other hand, reconstructed original images from photos augmented with color, such as grayscale and Hue-Saturation-Value (HSV) transformation. They attempted to bring different color styles closer to the original. It is difficult to maintain all source information due to the intricacy of deep neural networks and the instability of GANs. It can sometimes introduce artefacts, which can have negative consequences for further analysis (60). At the same time, because the network of deep learning-based approaches typically comprises millions of parameters, it requires a lot of computer power and has a low computing efficiency (61).

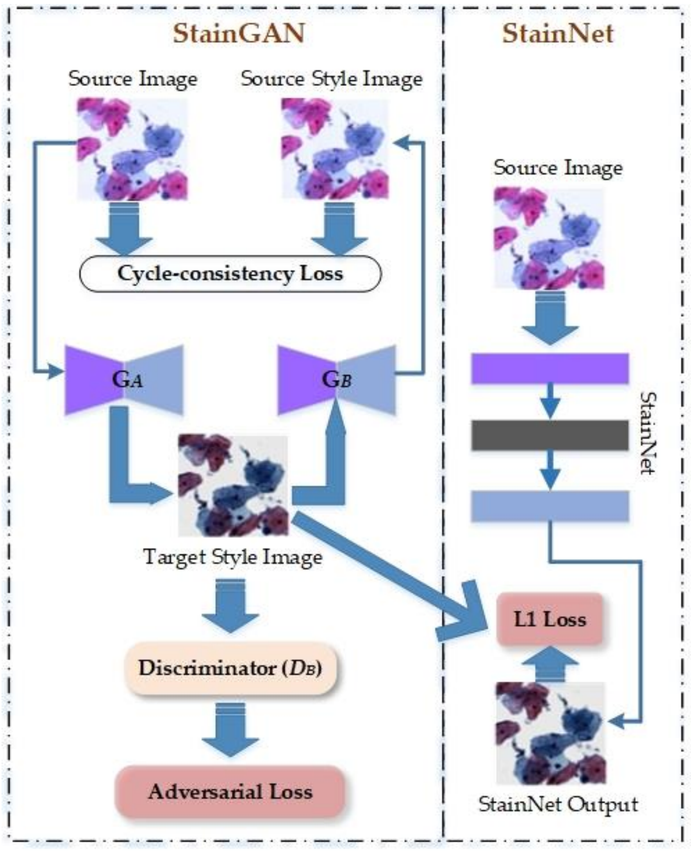

Figure 5: StainNet framework [55]

Color variations can be solved via stain normalization without the use of labeled data. Grayscale conversion methods primarily learn texture elements and discard color information. An unsupervised stain normalization approach was presented by Chen et al. [62]. Colorful photos were transformed to grayscale images using a stain removal module and a color encoding mask to preserve color information. A GAN model was used to process this mask and the grayscale image.

Gupta et al. [63] used color vector space geometry to conduct stain normalization. Hoque et al.

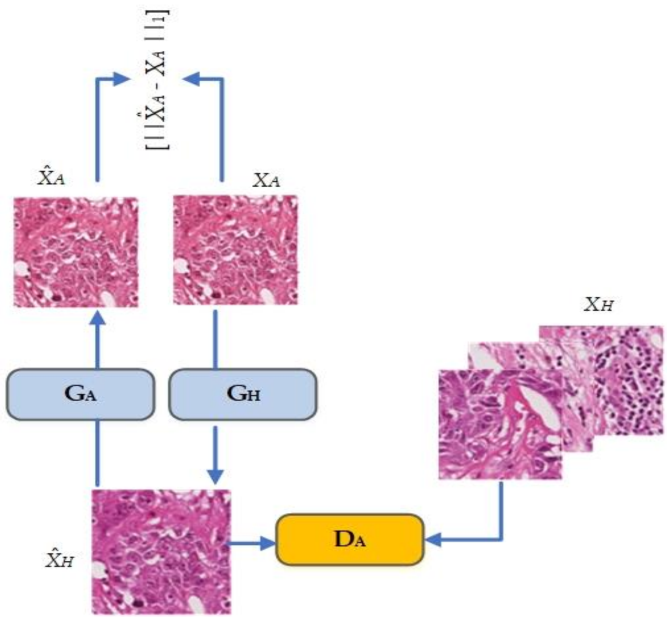

Figure 6: StainGAN framework [58]

[64] presented a Retinex model to normalize stain vectors in order to remove variability and improve analysis consistency. This approach had the disadvantage of necessitating time-consuming preprocessing to segment areas connected to each stain. Liang et al. [65] used GAN and the directional statistics-based color similarity index (DSCSI) as the loss for structure preservation in stain transformation. Using a pre-trained classifier network, a feature preserving loss was also computed to train the network. Hoque et al. [66] recommended a stain normalization technique based on the Retinex model in terms of area segmentation from stained tissue images to quantify the various stain components of the histochemical stains for the perfect reduction of variability. This method consistently had the minimum standard deviation, skewness value, and coefficient of variation in normalized median intensity measurements when compared to reference methods and tested on an organotypic carcinoma model based on myoma tissue. Janowczyk et al. [67] proposed a stain normalization method that used sparse autoencoders to cluster the pixels in the histogram. Mahapatra et al. [68] offered a self-supervised strategy to include semantic

guidance into a GAN-based stain normalization framework while still preserving precise structural information in their study. This method proved superior to existing methods since it does not require human segmentation maps. Between a pre-trained semantic network and the stain color normalization network, they integrate semantic information at different layers. The suggested system outperforms conventional color normalizing techniques, resulting in improved classification and segmentation results.

2.3 Few-Shot Learning

Few-shot learning is not the same as traditional supervised learning. The goal of few-shot learning isn't to have the model recognize the photos in the training set before moving on to the test set. Machine learning has a sub-area called few-shot learning. It's about classifying new data when you only have a few supervised training samples. Few-shot learning, as the name implies, is the method of feeding a learning model with a limited quantity of training data rather than the more common practice of using a big number of data. This method is most commonly used in the field of computer vision to use an object categorization model without multiple training samples and still yield acceptable results. If we have a problem categorizing bird species from photos, for example, some uncommon bird species may not have enough pictures to use in the training images. As a result, if we have a classifier for bird photos but not enough data, we'll regard the problem as a few-shot or low-shot machine learning problem.

Few-shot learning is simply a more adaptable variation of one-shot learning in which we have multiple training examples (usually two to five images). Sun et al. [69] presented few-shot learning at the 2019 Conference on Computer Vision and Pattern Recognition. This model established the bar for future study by providing cutting-edge results and paving the way for more advanced meta-transfer learning methods. To achieve spectacular results, many of the meta-learning and reinforcement-learning algorithms are integrated with traditional deep learning methods. One of the most prominent deep learning methods, prototypical networks, is widely utilized for this task. Human cell classification and analysis is a time-consuming process that frequently requires the assistance of a skilled professional. An active area of research includes automating cell classification using deep learning-based algorithms in an attempt to speed up this process. Walsh et al. [70] evaluate the viability of employing few-shot learning-based strategies to reduce the amount of data required for accurate training in this study. First, human cell classification is used to evaluate current state-of-the-art few-shot learning algorithms. The chosen algorithms are trained on a non-medical dataset before being tested on two human cell datasets that are out-of-domain.

Few-shot learning [71] is a machine learning method for learning a task from a limited number of images with supervision, addressing the abovementioned concerns. Few-shot learning-based segmentation is a new research area. Many existing few-shot segmentation models have a support branch [72-76] or a prototype learner [77], [78], and a query branch, similar to the original work [72]. Figure 7 depicts the basic concept of these models: Support images are sent to the support branch (or prototype learner), and query images are sent to the query branch; the connections between these two branches provide support information to help feature extraction and segmentation of the query images. As a result, researchers often followed a certain procedure when utilizing such models for few-shot segmentation tasks [72]. They apply the mode of receiving annotated support-query picture pairs to train a model on numerous tasks, then use it on an unseen target task for query segmentation with very few annotated samples. Another work based on few-shot medical image segmentation with few labeled examples is [80]. In this work Tang et al. [80] proposed a context relation encoder (CRE) that enhances context relationship information around the object boundary by using the correlation between foreground and background.

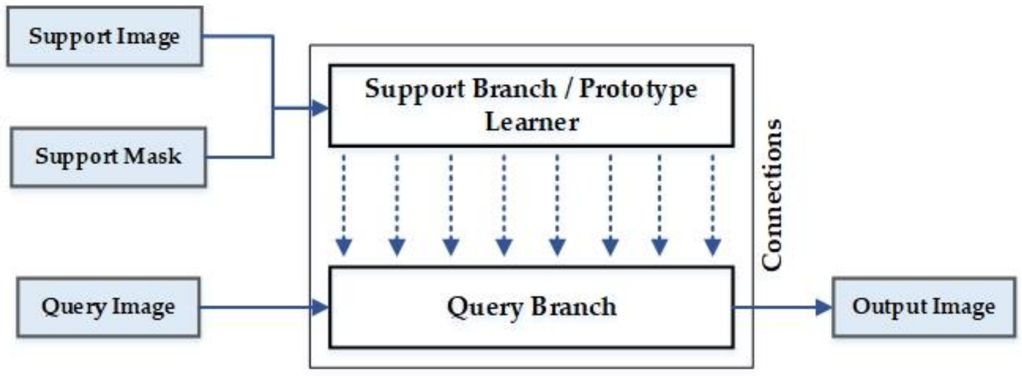

Figure 7: Overview of few-shot image segmentation models. [79]

A new approach for few-shot medical image segmentation is employing a recurrent module with CRE and prototype networks to modify the prediction mask repeatedly. Figure 8 shows the three primary components of the proposed model RP-Net: (1) A feature encoder that extracts features from both support and query images; (2) a context relation encoder (CRE) that uses correlation to improve local context relationship features; (3) a recurrent mask refinement module that iteratively uses CRE and a prototypical network to recapture and refine local context features.

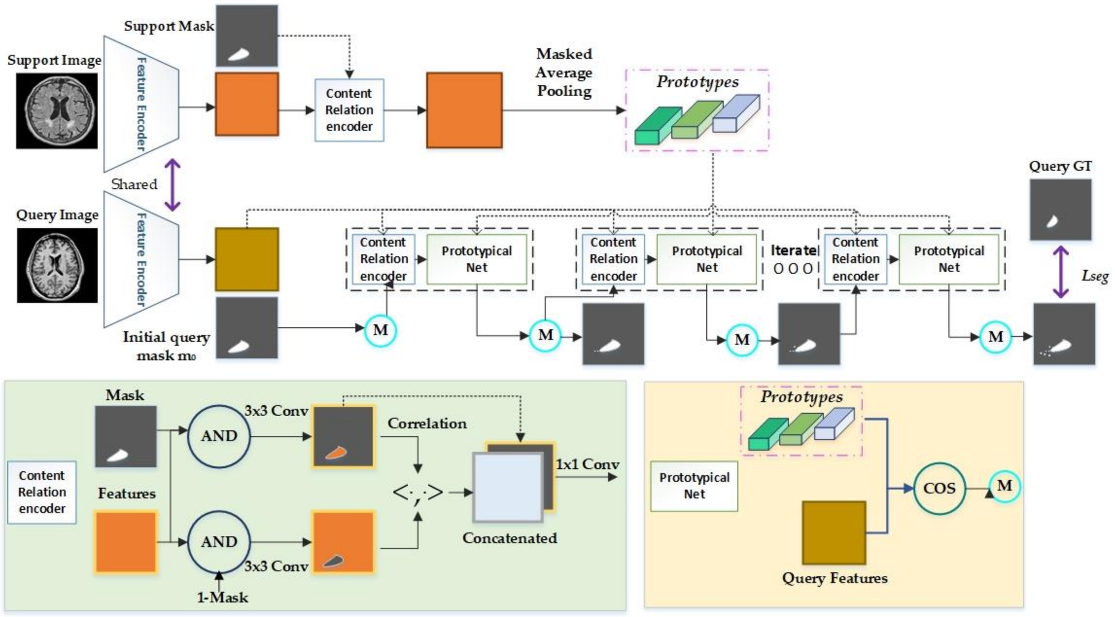

Figure 8: The architecture of RP-Net. [80]

2.4 Few-Shot Image Generation

It is good to build a model for image generation that generalize to new domains from a few examples. It is challenging when there is limited data available for training in GAN. Given a limited amount of training data, the performance of generative adversarial networks (GANs) heavily deteriorates. This is due to the discriminator having to memorize the exact training set. Due to the likelihood-free nature of GANs, obtaining gradients for the generator from assessing it on development sets is a serious difficulty when using meta-learning on them. Similarly, recent improvements in meta-learning have paved the way for new few-shot learning applications. Some improved work has been done to overcome this training with small data situations.

The study by Robb et al. [82] proposed few-shot GAN where they improved GAN training with a few images. Component analysisis approaches are repurposed in Few-Shot GAN (FSGAN), which learns to modify the singular values of pre-trained weights while freezing the corresponding singular vectors. This creates an ample parameter space for adaptation while keeping changes to the pre-trained weights limited. The primary goal is to fine-tune the GAN on small image domains by identifying a more adequate and limited parameter space for adjusting the pre-trained weights. Li et al. [83] proposed a few-shot image generation model with adaption technique to produce data with few samples. They adopt a pre-trained model to use less than 10 target domain samples without adding more parameters to regularize the weights during the adaption. Ojha et al. [84] introduced a unique GAN adaptation framework for few-shot image creation that ensures cross-domain correlation. They show that their model automatically discovers correspondences between relevant source and target domains to generate diversified and realistic images using substantial qualitative and quantitative data. The authors provide an anchor-based technique to encourage varying levels of realism over distinct regions in the latent space to further prevent overfitting.

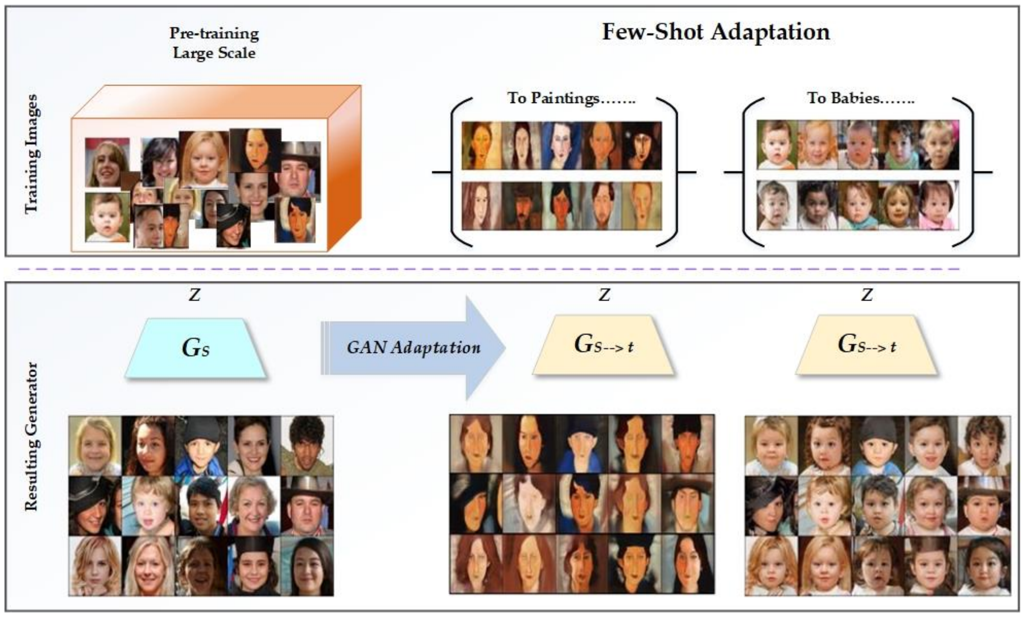

Figure 9: Overview of few-shot adaptation

Figure 9 depicts their work, which shows how the author has proposed to adapt a model trained on a large source dataset (Gs) to random picture domains, so that the final model (Gs→t) captures these target distributions with very few training samples. Their method reveals a one-to-one relationship between the distributions in the process, with noise vectors mapping to corresponding images in the source and target. Training a GAN model with few numbers of samples leads to a model overfitting issue. Xiao et al. [85] suggested a relaxed spatial structural alignment (RSSA) method for calibrating the target generative models during adaptation to address this issue. To address the generative model's identity degradation problem, it uses richer spatial structure priors of images from a source domain. They created a cross-domain spatial structural consistency loss that combines self-correlation and disturbance correlation consistency losses. They compress the original latent space of generative models to a subspace to relax cross-domain alignment.

The goal of few-shot image generation is to create a diverse set of high-quality photographs in a new domain using minimal training data. The most straightforward method is to fine-tune a GAN that has already been trained [86-88]. However, fine-tuning the network weights as a whole frequently produces unsatisfactory results. Researchers recommended changing a portion of the network weights [89], and Noguchi et al. [90] offered a new strategy focused on batch statistics, size, and shift of the hidden layers in the generator. They achieved stable generator training by supervised training only these parameters, and their method can generate higher quality images than earlier methods without collapsing, even when the dataset is small. FIGR stands for Few-shot Image Generation with Reptile, a GAN meta-trained with Reptile. With as few as four images from an unknown class, their [91] suggested model successfully creates novel images on MNIST and Omniglot. They also contribute FIGR-8, a new dataset for few-shot picture generation that has 1,548,944 icons divided into 18,409 categories. Lake et al. [92] is the first successful attempt at one-shot or few-shot image production that we are aware of. The images and stroke data are utilized to train a Bayesian model using Bayesian Program Learning on the Omniglot dataset introduced in the same paper. It expresses notions like a pen stroke as simple probabilistic programs and mixes them hierarchically to create visuals. This results in a model that can be trained on a single image of an unknown character and then used to create new samples of that letter.

2.5 Medical Image Analysis and Classification

Human professionals in the clinic, such as radiologists and physicians, does the majority of medical image interpretations. However, because of the complexity of medical imaging and the wide range of human expertise required, it is extremely difficult for radiologists and physicians to consistently provide an efficient and accurate diagnosis. As a broad area, medical image analysis is challenging for novices, even from computer vision or the clinical community, as it generally requires background knowledge from both sides. It is especially analyzing multiple datasets with different modalities as the datasets usually are nonstandard. On the other hand, although deep learning has dominated the research and application in medical image analysis [92, 93], it consumes large amounts of engineering labor to tune the deep learning models. As a result, automatic machine learning (AutoML) [94] has become increasingly essential. However, there exist few benchmarks for comparing AutoML in medical image classification.

Transfer learning from natural images is extensively employed in medical image analysis despite variances in image statistics, scale, and task-relevant properties [95-98]. Several empirical investigations [99-101] suggest that this enhances performance. However, extensive tests of this method by Raghu et al. [102] show that it does not always increase performance in medical imaging scenarios. They do, however, indicate that ImageNet transfer learning can speed up convergence and is especially useful when medical image training material is scarce. The study used tiny architectures and found considerable benefits with small amounts of data, notably when using ResNet-50 [103], which was their most significant architecture. Domain mismatch can be mitigated by transfer learning from in-domain data. Azizi at el. [104] suggested a method depicted in Figure 10, consisting of three steps: (1) SimCLR-based self-supervised pretraining on unlabeled ImageNet. (2) Additional unlabeled medical photos for self-supervised pretraining. If many images of each medical condition are available, a new Multi-Instance Contrastive Learning (MICLe) algorithm is utilized to create more informative positive pairs based on the diverse photographs. (3) Fine-tuning on tagged medical pictures under supervision. Steps (2) and (3), unlike step (1), are task and dataset-specific. A deep network was designed by Saba et al. [105] for segmentation and classification of leukemia. The author used fusion of transfer learning models. The authors used pretrained models such as DarkNet-53 and ShuffleNet. Also, principal component analysis (PCA) was used for informative feature selection. Datasets that were used for performance evaluation are ALL-IDB and LISC dataset.

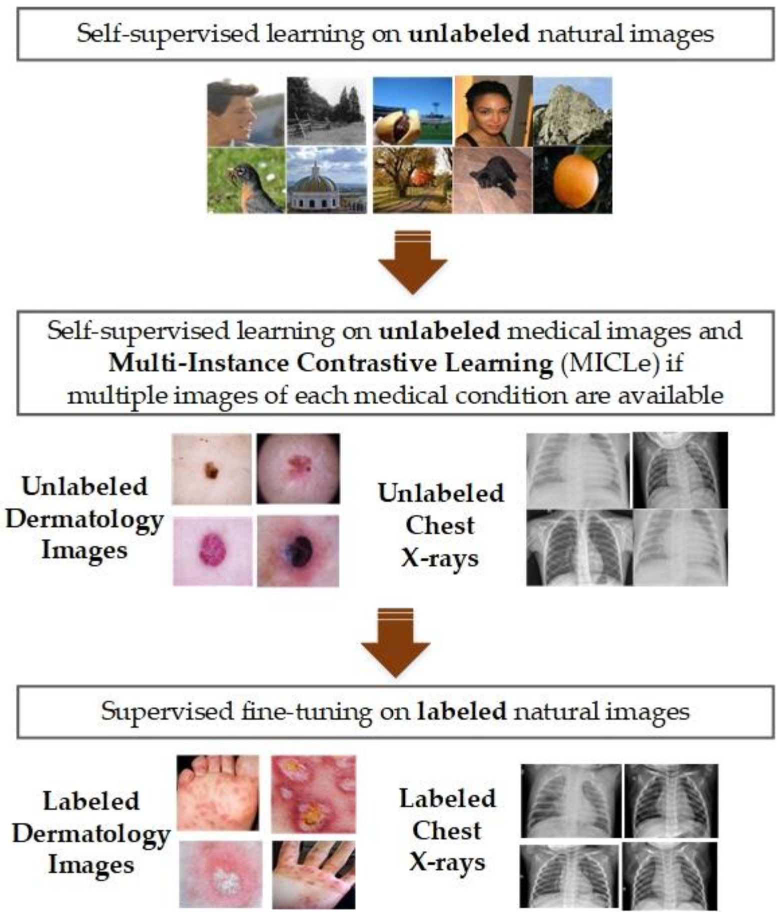
Self-supervised learning on **unlabeled** natural images
Self-supervised learning on **unlabeled** medical images and **Multi-Instance Contrastive Learning** (MICLe) if multiple images of each medical condition are available
Unlabeled Dermatology Images
Unlabeled Chest X-rays
Supervised fine-tuning on **labeled** natural images
Labeled Dermatology Images
Labeled Chest X-rays

Figure 10: Proposed approach by Azizi et al. [104] for medical image classification

Chapter 3: Proposed Methodology for Enhancing Classification of Microscopic Single-Cell Images

In this chapter, we present a comprehensive study of our proposed methodology for enhancing the classification of microscopic single-cell images procured from peripheral blood smears. Our proposed model can be conceptually categorized as a three-step process: data collection and preparation, data synthesis and classification. In the data collection stage, we collect data from two open-access data sources and perform preprocessing steps. To obtain better classification results, we implement stain normalization since the stain used in the two datasets are different. After normalization, we combine images belonging to the same class from the two data sources and form a single final dataset.

For image synthesis, we propose two generative adversarial networks (GAN) architecture. For cell types containing more than 1000 images, we propose a classifier-based GAN model; for cell types containing less than 100 images, we propose a few-shot image generation GAN model. After image synthesis, we obtain a normalized and balanced dataset which is further used for classification. We propose a novel deep learning model that uses the balanced dataset for training and classifying microscopic single-cell images procured from peripheral blood smears. The proposed framework is able to handle variations in datasets and complicated datasets. Our proposed model was trained on twelve cell types and could efficiently improve the classification accuracy for each cell type.

Figure 11 shows the conceptual view of the proposed approach for enhancing classification of microscopic single-cell images. Expert from EONE Laboratories helped us to form the final dataset. Preprocessing the data and combining the datasets or images of the cell types belonging to the same group was done with the help of expert intervention. The total number of cell types from the two data sources before preprocessing was nineteen, and after combining the datasets, we formed the final datasets with twelve cell types. We divide the cell types according to the number of images they contain and implement different synthesis techniques to achieve a balanced dataset.

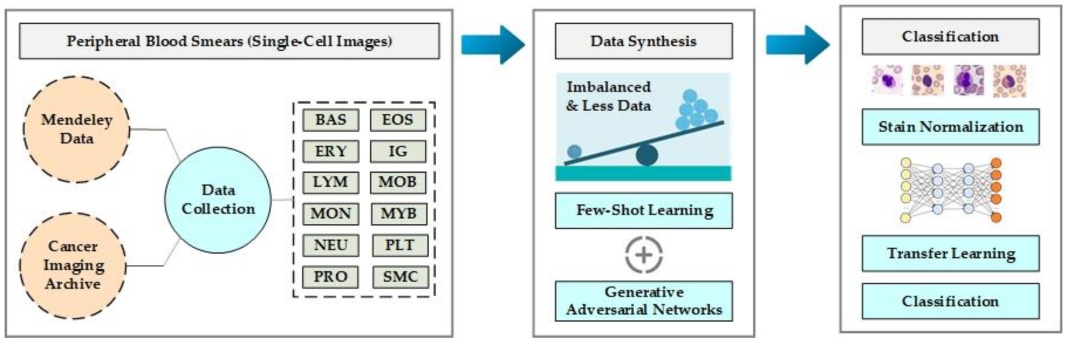

Figure 11: Conceptual view of the proposed methodology

In section 3.1 we present the overview of the proposed methodology. Section 3.2 presents the details of the dataset used and the data preprocessing techniques. In section 3.3 we present explanation of the foundations for our proposed methodology. Section 3.4 presents the model for stain normalization. Section 3.5 presents the classifier based generative adversarial networks architecture; section 3.6 presents the architecture for few-shot image generation and section 3.7 presents the novel deep learning model for classification.

3.1 Overview of the Proposed Methodology

The overview of the proposed methodology is shown in Figure 12. Our proposed approach as can be seen in Figure 12 has four layers. The first layer is the data acquisition and data preparation layer, where we collect data from two sources and with expert intervention, we first perform stain normalization. After normalizing and procuring images of the same intensity, we merge the images of two datasets belonging to the same class or cell types. We combine the datasets to form a new

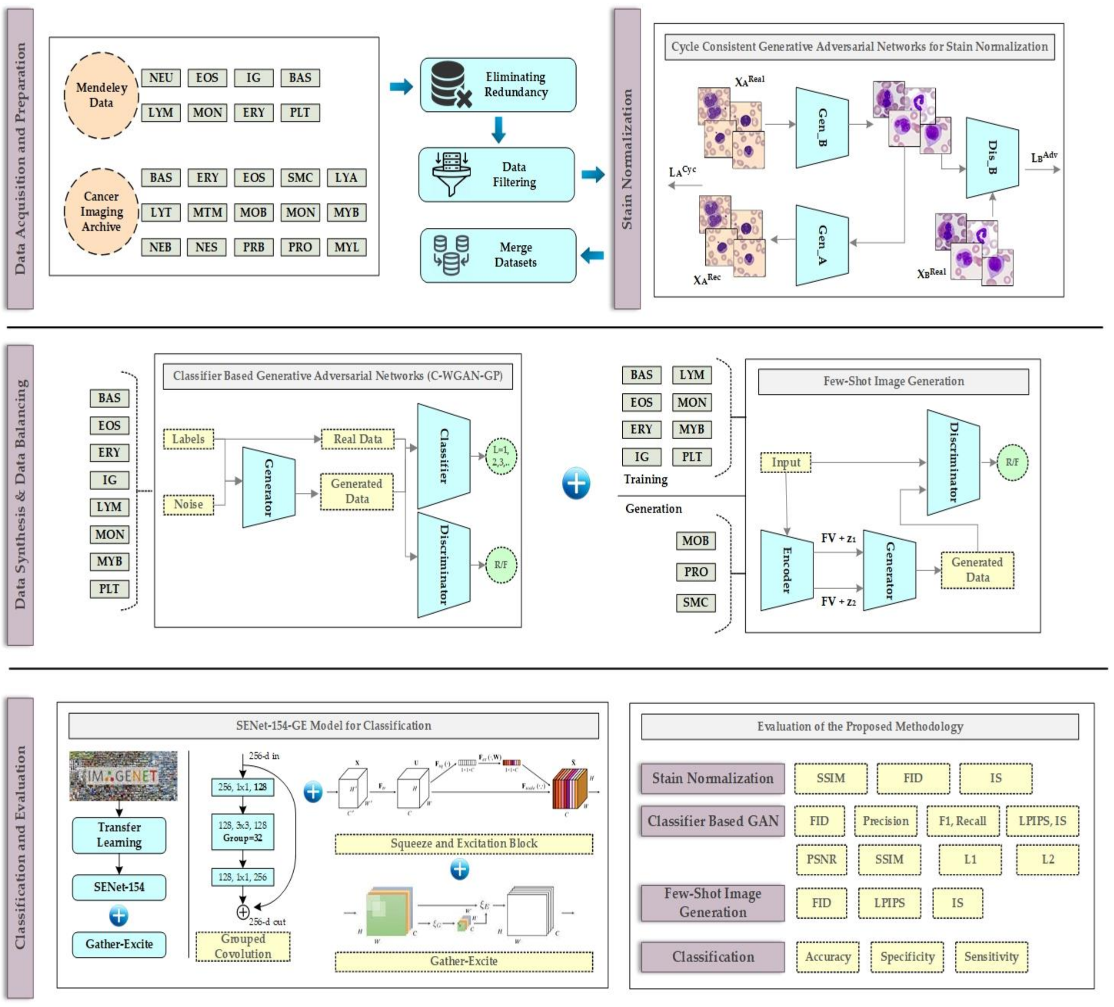

Figure 12: Layout of the proposed methodology

dataset. We perform stain normalization through cycle consistency generative adversarial networks (CycleGAN).

The third layer is the data synthesis and data balancing layer. Individual, as well as the newly-formed combined dataset, suffered from data imbalance problem. The number of images for each cell type varied to a great extent in few cases. For enhancing classification accuracy, it is important to have a uniform distribution of images for each cell type. In the data synthesis and data balancing layer, we first classify the cells in the dataset into two categories: cell types having more than thousand images and cell types containing less than hundred images. For cell types having more than thousand images, we perform data synthesis through classifier based generative adversarial networks. For cell types containing less than hundred images we implement few-shot image generation where we use the images of the cell types with more than thousand images for the training phase. In the generation phase of the few-shot image generation, we generate images for only the cell types having less than hundred images. For few-shot image generation we developed an encoder based generative adversarial networks in few-shot learning settings.

The next layer is the classification and evaluation layer. After data balancing, we propose a novel deep learning model that uses squeeze and excitation networks (SENet), ResNeXt-101 and gather-excite (GE) module to classify the microscopic single-cell images obtained from the peripheral blood smears. For evaluating our proposed methodology, we divide the evaluation phase into four phases. First, we evaluate the performance of stain normalization using three evaluation metrics as shown in Figure 12. In the second phase, we measure the performance of classifier-based GAN for synthetic blood cell image generation. We use six evaluation metrics and also present visual result for the image generation. In the third phase, we assess the implementation of few-shot image generation using three evaluation metrics and finally, we quantify the performance of the classification model using the original dataset and the dataset prepared through the proposed methodology. For each module, we compare our proposed model with existing models. We perform ablation studies to justify the proposed contribution and to investigate the performance of our proposed architectures.

3.2 Overview of the Datasets and Data Preprocessing Techniques

We collected data from two open access data sources that were merged to form a new dataset for our work after preprocessing and stain normalization. Microscopic single-cell images from peripheral blood smears were collected from Mendeley Data Source [106] and Cancer Imaging Archive [107]. The Mendeley dataset contains 17,092 blood cell images that has no irregularities and were acquired from the Hospital Clinic of Barcelona in the Core Laboratory through CellaVision DM96 analyzer. The dataset contains eight cell types: immature granulocytes that has promyelocytes, metamyelocytes and myelocytes, neutrophils, basophils, lymphocytes, eosinophils, monocytes, erythroblasts and platelets. Table 2. shows the count of images for cell type in the first dataset, i.e. Mendeley Dataset.

Table 2: Count of images for each types of cells in the Mendeley Dataset

Types of Cells	**Count of Images**
Neutrophil (NEU)	3329
Eosinophil (EOS)	3117
Basophil (BAS)	1218
Lymphocyte (LYM)	1214
Monocyte (MON)	1420
Immature Granulocyte (Metamyelocyte, Myelocyte) (IG)	2895
Erythroblast (ERY)	1551
Platelet (Thrombocytes) (PLT)	2348
Total	**17092**

The images are present in JPG format. The images were taken from individuals who did not have any form of infection, hematologic disease or oncologic disease and were also free from drug consumption during blood collection. The blood cell count in the Mendeley dataset were obtained by analyzing the blood sample in the Advia 2120 instrument. They prepared the smear using the Sysmex SP1000i slide maker-stainer with May Grünwald-Giemsa staining.

Table 3: Count of images for each types of cells in the Cancer Imaging Archive

Types of Cells	**Count of Images**
Basophil (BAS)	79
Erythroblast (EOS)	78
Eosinophil (EOS)	424
Smudge Cell (SMC)	15
Lymphocyte (Atypical) (LYA)	11
Lymphocyte (Typical) (LYT)	3937
Monoblast (MOB)	26
Monocyte (MON)	1789
Myelocyte (MYL)	42
Myeloblast (MYB)	3268
Neutrophil (Band) (NEB)	109
Neutrophil (Segmented) (NES)	8484
Promyelocyte (Bilobed) (PRB)	18
Promyelocyte (PRO)	70
Metamyelocyte (MTM)	15
Total	**18365**

The Cancer Imaging Archive consists of 18,365 expert annotated single-cell images procured from the peripheral blood smears at the Munich University Hospital between the year 2014 and 2017. Table 3 presents the number of images for each type of cell existing in Cancer Imaging Archive. The data were gathered from hundred patients who did not have signs of haematological malignancy and hundred patients who were diagnosed with Acute Myeloid Leukemia (AML). The dataset contains images of fifteen cell types: basophil, monoblast, eosinophil, promyelocyte, lymphocyte (atypical), erythroblast, metamyelocyte, lymphocyte (typical), monocyte, myelocyte, myeloblast, neutrophil (segmented), promyelocyte (bilobed), neutrophil (band), and smudge cells. M8 (Precipoint GmbH, Freising, Germany) digital scanner was used for image acquisition. Figure 13 presents examples of images of each cell types and how does it look in microscopic images.

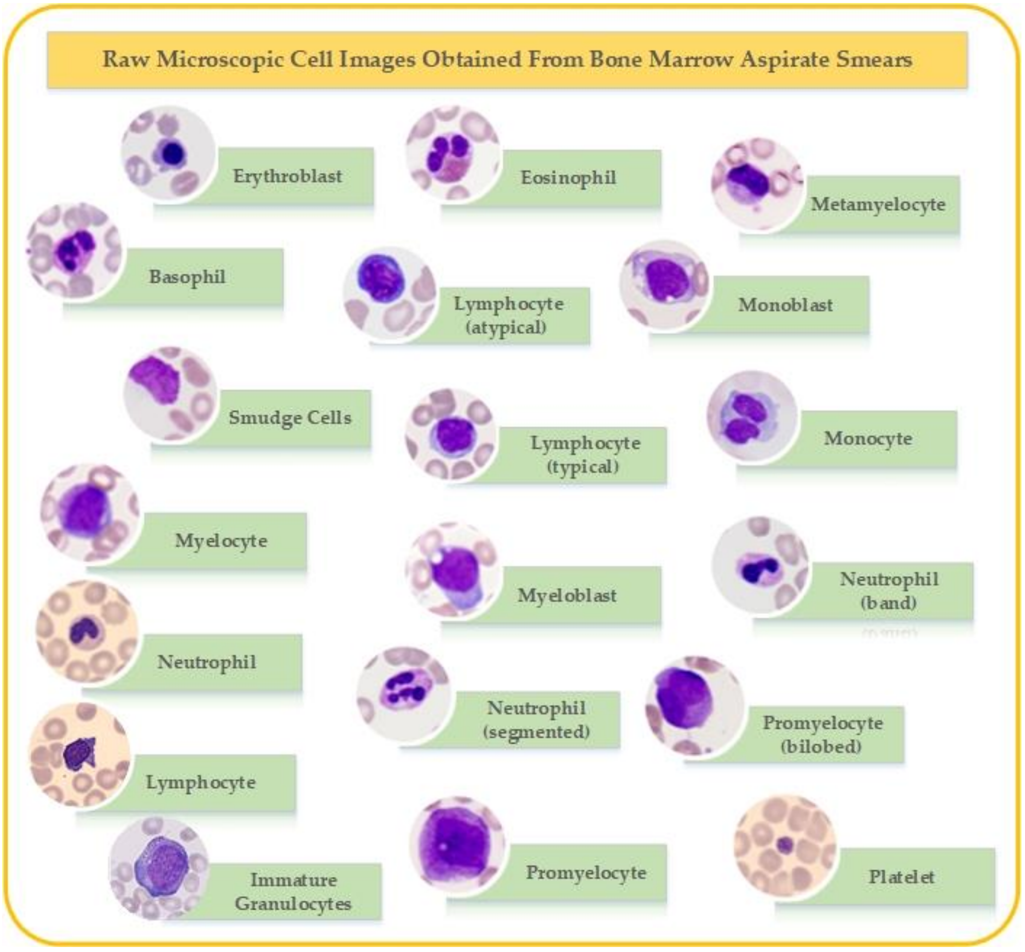

Figure 13: Sample of microscopic cell images

3.2.1 Data Preprocessing Techniques

In the data preprocessing section, we remove redundant or duplicate images. We filter the images with the help of a medical expert from EONE Laboratories. After elimination, we perform stain normalization and then we combine images from two sources belonging to the same cell type.

Table 4. And Table 5. shows the number of images removed from each cell type present in the Cancer Imaging Archive and the Mendeley Dataset.

Table 4: Count of images eliminated for each types of cells present in Cancer Imaging Archive

Types of Cells	Count of Images
Basophil (BAS)	17
Erythroblast (ERY)	1
Eosinophil (EOS)	3
Smudge Cell (SMC)	0
Lymphocyte (Atypical) (LYA)	4
Lymphocyte (Typical) (LYT)	119
Monoblast (MOB)	0
Monocyte (MON)	624
Myelocyte (MYL)	3
Myeloblast (MYB)	164
Neutrophil (Band) (NEB)	27
Neutrophil (Segmented) (NES)	1138
Promyelocyte (Bilobed) (PRB)	0
Promyelocyte (PRO)	1
Metamyelocyte (MTM)	2
Total	**2103**

Table 5: Count of images eliminated for each types of cells in the Mendeley Dataset

Types of Cells	Count of Images
Neutrophil (NEU)	13
Eosinophil (EOS)	0
Basophil (BAS)	56
Lymphocyte (LYM)	1
Monocyte (MON)	2
Immature Granulocyte (Metamyelocyte, Myelocyte) (IG)	14
Erythroblast (ERY)	81
Platelet (Thrombocytes) (PLT)	9
Total	**176**

After data elimination, we merge images of cell types belonging to the same cell class or family and form a single cell class. Subgrouping of the cell types were done by medical expert based on maturation sequence of human hematopoietic cells. Figure 14. shows the subgrouping of the cell types in our proposed work.

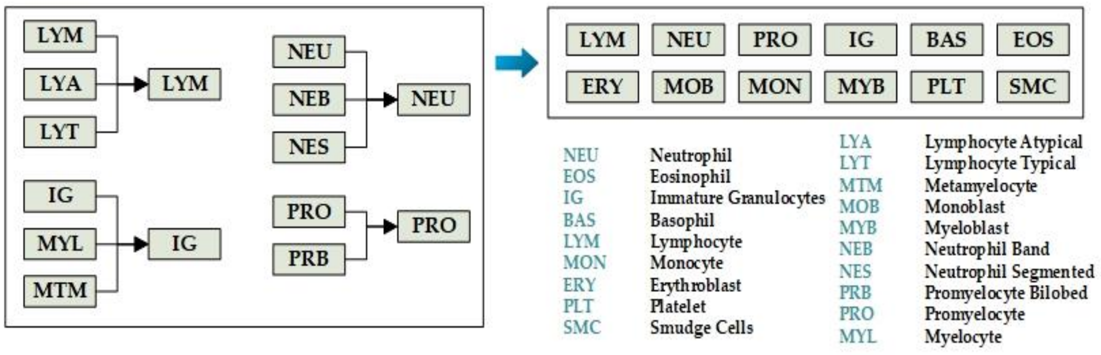

Figure 14: Subgrouping of cell types

As shown in Figure 14, lymphocyte typical, lymphocyte and lymphocyte atypical were integrated to form a single group, i.e. lymphocyte. Also, immature granulocyte, metamyelocyte and myelocyte were combined to form immature granulocyte group. Images from neutrophil, neutrophil band and neutrophil segmented were combined to form a single cell type, i.e. neutrophil. The fourth group promyelocyte was created by combining cell types promyelocyte and promyelocyte bilobed.

Combining similar cell types images from the two data sources, we created a new dataset with a total of nineteen cell types. The final dataset contains twelve cell types after subgrouping. Table 6. presents the total number of cell types and the total number of images for each cell type in the final dataset.

Table 6: Total number of cell types and images for each types of cells in the final dataset.

Sl. No.	Types of Cells	Count of Images
1	Basophil (BAS)	1224
2	Eosinophil (EOS)	3538
3	Erythroblast (ERY)	1547
4	Immature Granulocyte (IG)	2933
5	Lymphocyte (LYM)	5038
6	Monoblast (MOB)	26
7	Monocyte (MON)	2583
8	Myeloblast (MYB)	3104
9	Neutrophil (NEU)	10744
10	Platelet (PLT)	2339
11	Promyelocyte (PRO)	87
12	Smudge Cells (SMC)	15
	Total	**33178**

3.3 Foundations of the Proposed Methodology

The two basic building blocks of our proposed methodology are generative adversarial networks (GAN) and few-shot learning. In this section, we will explain the architecture of generative adversarial networks and the concept of meta learning and few-shot learning.

3.3.1 Generative Adversarial Networks (GAN)

In machine learning when we use a model to make predictions, it is known as predictive modeling. When we train this predictive model with a training dataset comprising an input variable and an output class label associated with it, it is known as supervised learning. In supervised learning, for every input a corresponding label is provided and the goal is to learn the mapping from the input to the output. Supervised learning approaches include classification and regression. Another kind of learning is where the model is only provided with the input variable, and no output variable or labels exist. The model has to extract and learn from the patterns of the input data. This form of learning is referred to as unsupervised learning. Unsupervised learning includes generative modeling and clustering. Discriminative models discriminate between various data instances and draw boundaries in the data space, whereas generative model captures the distribution of the input data and generates new data instances or data points. Discriminative models are mainly utilized for supervised machine learning and are known as conditional models. They are not capable of generating new data points. The ultimate goal is to separate one class from the other class. Generative models focus on modeling how the data was generated and how it is placed throughout the space. Generative models are used in unsupervised machine learning problems and applies probability estimates and likelihood for modeling data points and to differentiate between various classes or labels present in the dataset.

Generative adversarial networks (GAN) as the name suggests, is a generative model that was introduced in the year 2014 by Ian Goodfellow. GANs are capable of generating realistic data or synthetic data by using two neural networks. GANs are a deep learning based generative model that consists of two models: Generator (Gen) and Discriminator (Dis). The generator is trained to generate fake data, and the discriminator is trained to distinguish between the generated and the real data. The term adversarial in the generative adversarial networks model refers to the minmax game concept between the two models in the GAN framework. The objective of the generator is to fool the discriminator by generating realistic samples of images, and the objective of the discriminator is to be able to discriminate between real and the generated images and prove the generator wrong. This is known as a zero-sum game. The GAN architecture is shown in Figure 15.

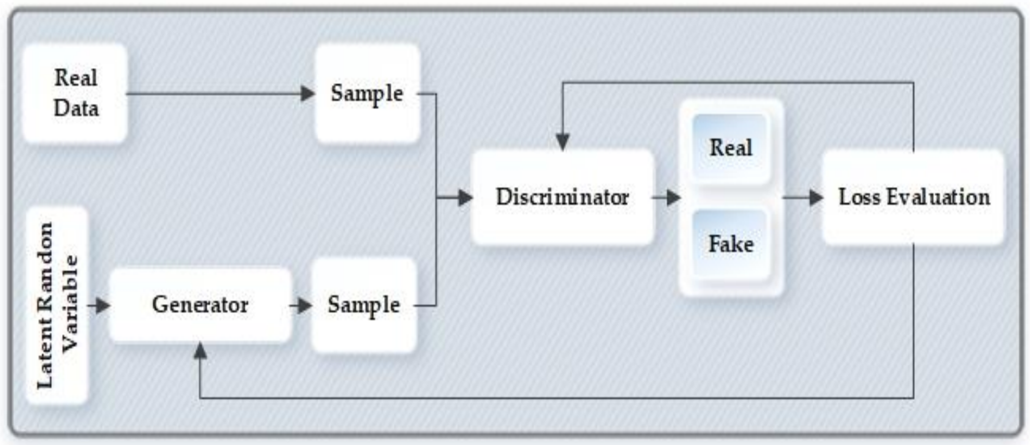

Figure 15: Generative adversarial networks (GAN) architecture

The two networks continuously try to outsmart each other. The more the generator gets better at generating new realistic data, the better the discriminator has to perform in distinguishing the real samples from the generated samples. In Table 7, we describe the input, output and goal of the generator and the discriminator in the GAN architecture and in Table 8, we describe some parameters and variables we will use to show the derivation of the loss functions of the generator and the discriminator.

Table 7: Input, output and goal of generator and discriminator networks

Function	Generator	Discriminator
Input	Random noise vector	Two Sources: - Real Samples from training dataset - Generated fake samples from the generator
Output	Fake samples that looks like original data	Predicted probability of whether the sample is real
Goal	Generate fake samples which are indistinguishable from the real data	Distinguish whether data is coming from real distribution or belongs to the generated distribution

Table 8: Definition of variables used for mathematical expressions

Variable	Definition
x	Real or original data
z	Latent random vector
Dis	Discriminator
Gen	Generator
$P_z(z)$	Distribution of input noise
$P_\mathrm{d}\ (x)$	Distribution of real data
$P_r(x)$	Generated distribution
Gen(z)	Generated data
Dis(x)	Discriminator's (real data)
Dis(Gen(z))	Discriminator's (fake data)

In Figure 15, real data refers to the original dataset or samples that we want the generator to learn to synthesize as realistic as possible. Random noise vector is a vector of random numbers that is fed to the generator, which utilizes it as a starting point for generating fake samples. For every prediction of the discriminator, just like a classifier, we determine how good it is. For every prediction, the loss is evaluated, and the result is used to iteratively tune the trainable parameters of the generator and the discriminator networks through backpropagation. The discriminator maximizes the objective to correctly predict real and fake and the generator tries to minimize the objective that the discriminator correctly identifies the samples. From binary cross-entropy loss, we can derive the original loss function of GANs, which can be written as in Equation 1:

$$\mathrm{Loss}(\widehat{\mathrm{w}},\mathrm{w}) = [\mathrm{w} \cdot \log \widehat{\mathrm{w}} + (1-(\mathrm{w}) \cdot \log(1-\widehat{\mathrm{w}})] \qquad (1)$$

Where w is the original data and $\widehat{\mathrm{w}}$ is the reconstructed data. When we train the discriminator, the label for the data coming from the real distribution i.e. $P_{\mathrm{d}}(x)$ is w=1 (real data) and the reconstructed data $\widehat{\mathrm{w}}$ = Dis(x). So, we can write the above loss function as in Equation 2:

$$\mathrm{Loss}(\mathrm{Dis}(\mathrm{x}),1) = \log\big(\mathrm{Dis}(\mathrm{x})\big) \qquad (2)$$

and for the generated data, the label is w=0 (generated data), $\widehat{\mathrm{w}}$ = Dis(Gen(z)) and the loss function from Equation 1 can be written as in Equation 3:

$$\mathrm{Loss}\big(\mathrm{Dis}\big(\mathrm{Gen}(\mathrm{z})\big),0\big) = \log\Big(1-\mathrm{Dis}\big(\mathrm{Gen}(\mathrm{z})\big)\Big) \qquad (3)$$

The goal of the discriminator is to correctly distinguish real and generated data. So, Equation 2 and Equation 3 should be maximized. The final loss function of the discriminator can be defined as shown in Equation 4:

$$\mathrm{Loss}^{(\mathrm{Dis})} = \max\Big[\log\big(\mathrm{Dis}(\mathrm{x})\big) + \log\Big(1-\mathrm{Dis}\big(\mathrm{Gen}(\mathrm{z})\big)\Big)\Big] \qquad (4)$$

The role of the generator is to compete against the discriminator and minimize Equation 4. So, the loss function can be defined as in below Equation 5:

$$\mathrm{Loss}^{(\mathrm{Gen})} = \min\Big[\log\big(\mathrm{Dis}(\mathrm{x})\big) + \log\Big(1-\mathrm{Dis}\big(\mathrm{Gen}(\mathrm{z})\big)\Big)\Big] \qquad (5)$$

We can define the combined loss function for a single data point as shown in Equation 6:

$$\mathrm{Loss} = \min_{\mathrm{Gen}}\max_{\mathrm{Dis}}\Big[\log\big(\mathrm{Dis}(\mathrm{x})\big) + \log\Big(1-\mathrm{Dis}\big(\mathrm{Gen}(\mathrm{z})\big)\Big)\Big] \qquad (6)$$

For the entire dataset, the minmax function can be defined as Equation 7:

$$\min_{\mathrm{Gen}} \max_{\mathrm{Dis}} \mathrm{F}(\mathrm{Dis}, \mathrm{Gen}) = \mathbb{E}_{x\sim P_d(x)}[\log \mathrm{Dis}(x)] + \mathbb{E}_{z\sim P_z(z)}\left[\log\left(1-\mathrm{Dis}\big(\mathrm{Gen}(z)\big)\right)\right]$$
$$= \mathbb{E}_{x\sim P_d(x)}[\log \mathrm{Dis}(x)] + \mathbb{E}_{x\sim P_r(x)}[\log(1-\mathrm{Dis}(x)]] \tag{7}$$

Kullback-Leibler divergence, known as KL divergence and Jensen-Shannon divergence, or JS divergence, are the metrics that measure the similarity between two probability distributions. KL divergence quantifies how one probability distribution (PA) differs from another probability distribution (PB), and JS divergence quantifies the similarity between two probability distributions that are bounded by [0,1]. KL and JS can be defined as Equation 8 and Equation 9:

$$\mathrm{KLD}(P_A \parallel P_B) = E_{x\sim P_A} \log\frac{P_A}{P_B} \tag{8}$$

$$\mathrm{JSD}(P_A \parallel P_B) = \frac{1}{2}\mathrm{KLD}\left(P_A \parallel \frac{P_A+P_B}{2}\right) + \frac{1}{2}\mathrm{KLD}\left(P_B \parallel \frac{P_A+P_B}{2}\right) \tag{9}$$

For any generator, the optimal discriminator can be obtained by maximizing the value function in Equation 7 through a partial derivative of V(Gen,Dis) with respect to Dis(x). The optimal discriminator is denoted by Dis$^+$(x) and it occurs when the condition as shown in Equation 10 occurs.

$$\frac{P_d(x)}{\mathrm{Dis}(x)} - \frac{P_r(x)}{1-\mathrm{Dis}(x)} = 0 \tag{10}$$

Rearranging Equation 10, we get Equation 11:

$$\mathrm{Dis}^+(x) = \frac{P_d(x)}{P_d(x)+P_r(x)} \tag{11}$$

So, if a sample x is realistic, we expect $P_d(x)$ to be near about one and $P_r(x)$ to converge to zero and in that case the optimal discriminator would assign one to the realistic sample. But, for a generated sample x=Gen(z), the optimal discriminator would assign zero to the sample.

For training the generator, we assume that the discriminator is fixed and we put Equation 9 in the value function as shown in Equation 12:

$$\begin{aligned} F(\mathrm{Gen}, \mathrm{Dis}^{+}) &= \mathbb{E}_{x\sim P_d}\left[\log\left(\mathrm{Dis}^{+}(x)\right)\right] + \mathbb{E}_{x\sim P_r}\left[\log\left(1-\mathrm{Dis}^{+}(x)\right)\right] \\ &= \mathbb{E}_{x\sim P_d}\left[\log\frac{P_d(x)}{P_d(x)+P_r(x)}\right] + \mathbb{E}_{x\sim p_r}\left[\log\frac{P_r(x)}{P_d(x)+P_r(x)}\right] \end{aligned} \tag{12}$$

In Equation 12 we exploit the properties of logarithms and take out a -log 4 and change the whole equation accordingly so that we can interpret it in KL divergence equation as shown in Equation 13 and Equation 14:

$$\begin{aligned} F(\mathrm{Gen}, \mathrm{Dis}^{+}) &= \mathbb{E}_{x\sim P_{data}}\left[\log\frac{P_d(x)}{P_d(x)+P_r(x)}\right] + \mathbb{E}_{x\sim p_r}\left[\log\frac{P_r(x)}{P_d(x)+P_r(x)}\right] \\ &= -\log 4 + \mathbb{E}_{x\sim P_d}\left[\log P_d(x) - \log\frac{P_d(x)+P_r(x))}{2}\right] \\ &\quad + \mathbb{E}_{x\sim P_r}\left[\log P_r(x) - \log\frac{P_d(x)+P_r(x))}{2}\right] \end{aligned} \tag{13}$$

$$F(\mathrm{Gen}, \mathrm{Dis}^{+}) = -\log 4 + \mathrm{KLD}\left(P_d \parallel \frac{P_d+P_r}{2}\right) + \mathrm{KL}\left(P_r \parallel \frac{P_r+P_r}{2}\right) \tag{14}$$

and, now we can incorporate the JS divergence shown in Equation 9 from Equation 14 as below in Equation 15.

$$F(\mathrm{Gen}, \mathrm{Dis}^{+}) = -\log 4 + 2 \cdot \mathrm{JSD}(P_d \parallel P_r) \tag{15}$$

The objective of training the generator is to minimize the value function, so we want the JSD to be as small as possible between the distribution of the real data and the generated data. Therefore, for a generator to be optimal, Pr should be as close to Pd as possible, which means the generator should produce realistic samples that cannot be distinguished from the real data by the discriminator.

According to game theory, every zero-sum game has a Nash Equilibrium. A GAN model is said to have reached Nash Equilibrium when both the generator and the discriminator stops learning, and both of them cannot improve their performance anymore. When the Nash Equilibrium is achieved, the GAN model is considered to have converged. At this point, the GAN training is stopped since the generator and the discriminator cannot learn further.

3.3.2 Few-Shot Learning and Meta Learning

For machine learning models to perform exceptionally, we require enormous data. But, in many scenarios, abundant data is not present. Few-shot learning helps in building models that can perform accurately even with a small dataset or, as the name denotes, few examples of data. The advantage of few-shot learning is not only that we can build a model with limited information, but also it reduces the computational cost, requires less time and reduces resource cost, e.g. lowering the cost of labeling extensive data. Few-shot learning is an application of meta learning that differs from conventional supervised learning in a way that in supervised learning the model is trained on some training data and then evaluated on some unseen data but in meta learning the model learns to learn. So, while the meta learning is about learning how to learn, few-shot learning is about learning from few examples of data. Meta learning and few-shot learning both has the goal of generalizing the learning process.

Few-shot learning is defined through a function that is trained to predict the similarity between samples. In few-shot learning paradigm, the training data is known as the Support set, and the data to be evaluated or the test data is known as the Query set. The support set is different from the training set of traditional machine learning algorithms in a way that the traditional training data consists of a large volume of data but the Support set only consists few labeled examples or data from each class. Unlike conventional supervised learning, the Query set of few-shot learning contains data of classes that are not present in the support set and are never seen before. The terminologies used in few-shot learning are:

- **N-way:** N represents the number of classes. So, if there are 3 classes, then it is denoted as 3-way.
- **K-shot:** K denotes the number of examples present for each class. So, if there are 3 classes having 4 examples each then it would be called 3-way 4-shot learning.

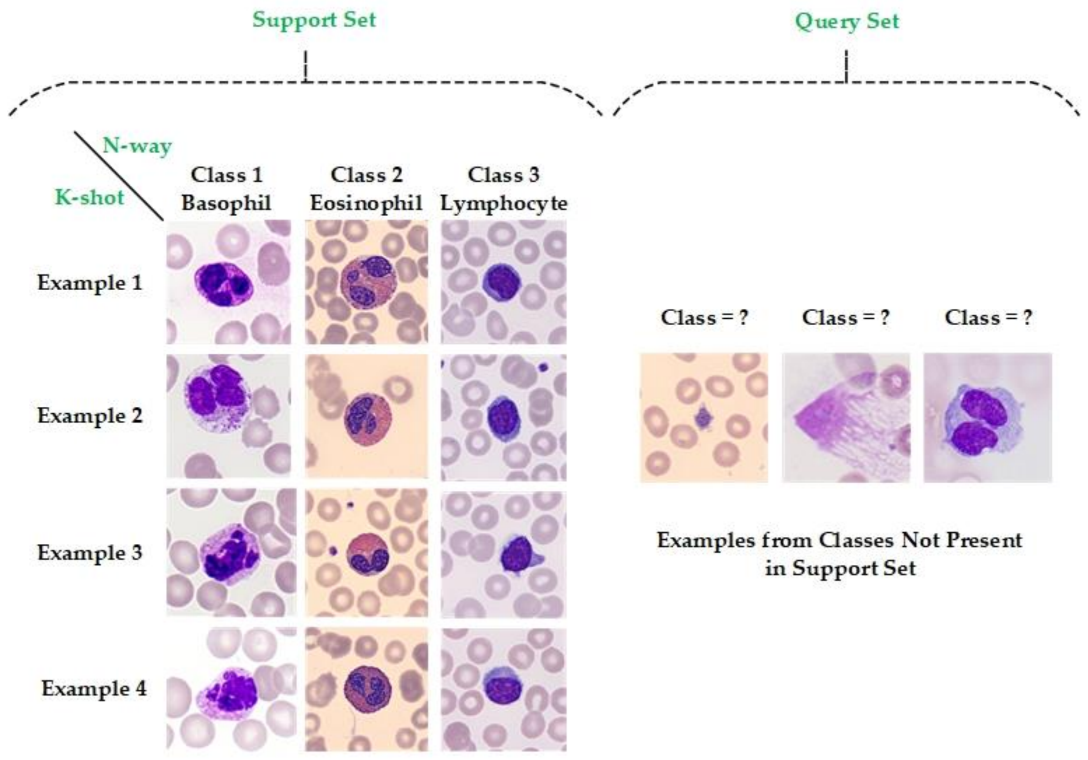

Figure 16: Few-shot terminologies and dataset structure

Figure 16 shows the representation of few-shot learning terminologies and the dataset structure for few-shot learning.

Few-shot learning follows certain approaches when building a model. The approaches are defined based on three categories: previous knowledge of similarity, knowledge based on learning and existing knowledge about the data. Previous knowledge about similarity helps the model to distinguish different classes that are unseen and was not used in the training process. The model learns the patterns and the embeddings in the training dataset. Some examples are Siamese networks [108] and triplet networks [109] which can differentiate between two classes that are unseen. Matching networks [110], prototypical networks [111] and relational networks [112] are models that can learn the patterns from training data and distinguish between multiple or more than two unseen classes. Knowledge based on learning refers to the scenario where models limits the learning model on hyperparameters that can generalize prior learning on unseen data and perform better for few-shot data. Model agnostic meta learning (MAML) [113] and meta learning LSTMs [114] are examples of few-shot learning based approach that depends on existing knowledge of rules and hyperparameters. Previous knowledge of data is another approach on building a few-shot learning model. These models have existing knowledge about the structure, patterns and variables of the dataset. It is also useful for generating new synthetic data. Examples are the Penn-stroke model [115] and analogies [116].

In meta learning, there are a set of training and test tasks. Meta learning is about learning to learn, and a model is said to be learning if, with every task experience, the model improves its performance on solving the training tasks and if the experience can be used for solving the test tasks.

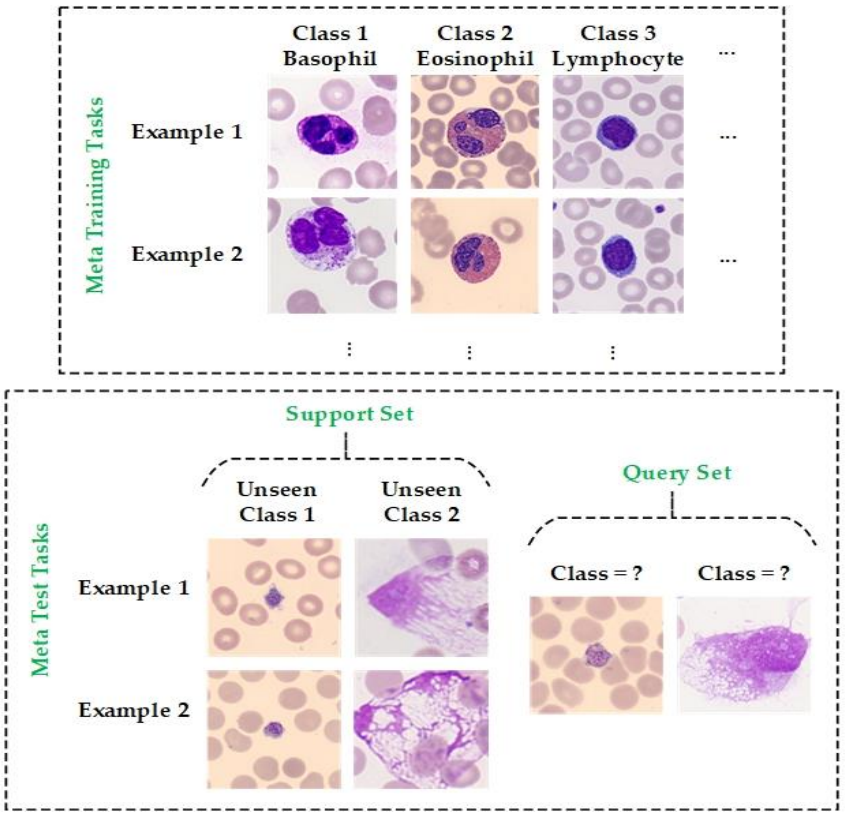

Figure 17: Meta learning training and test tasks layout

As shown in Figure 17 in meta learning first the model learns from the meta training dataset or tasks so that it can perform well on the meta test dataset. The meta training learning is adapted on the support set of meta test dataset. The model learns about the meta test support set through learning adaptation from the meta training tasks and then the model is evaluated on the query set of the meta test dataset. The meta test dataset has unseen classes on the support and the query set. Every task has an associated dataset defined as data=(a,y) where a is considered as the data samples and y is the label associated with it. So, a model with trainable parameter φ can be represented as learning the Pφ(y|a) where φ is defined as in Equation 16

$$\varphi = \underset{\varphi}{\operatorname{argmax}}\, E_{(a,y)\in data}\, [P_{\varphi}(y \mid a)]. \qquad (16)$$

$$\varphi = \underset{\varphi}{\operatorname{argmax}}\, \mathbb{E}_{Y\in\mathcal{Y}} \left[\mathbb{E}_{Sup^{Y}\subset data, Que^{Y}\subset data}\left[\textstyle\sum_{(a,y)\in Que^{Y}} P_{\varphi}\left(a,y,Sup^{Y}\right)\right]\right] \qquad (17)$$

Referring to Equation 17 we will start with the discussion about task sampling in meta learning. The support and query set in a task or dataset is defined as data=(Sup,Que). We then take subset of the labels $Y \in \mathcal{Y}$ so that $Sup^{Y}, Que^{Y} \in data$.

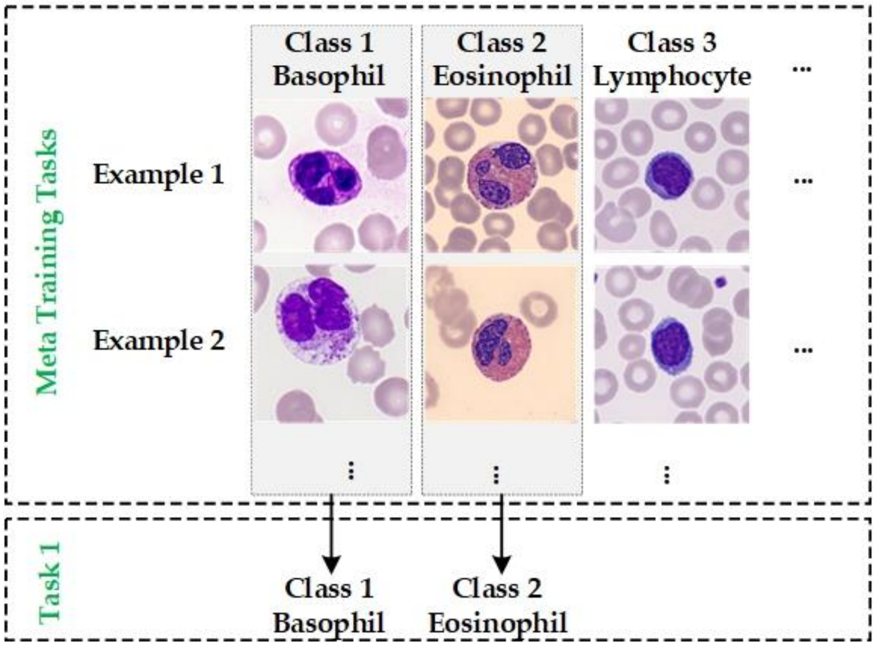

Figure 18: Task sampling in meta learning

The training is performed in episode wise fashion and we would explain the training for one episode with example. So, first we sample N classes in 1 task from the meta training dataset. We can create multiple tasks. As shown in Figure 18, in task 1 we take two classes from the training dataset.

After sampling the task, we define the support set for the meta training dataset for each task. We sample images from classes defined in the task as shown in Figure 19.

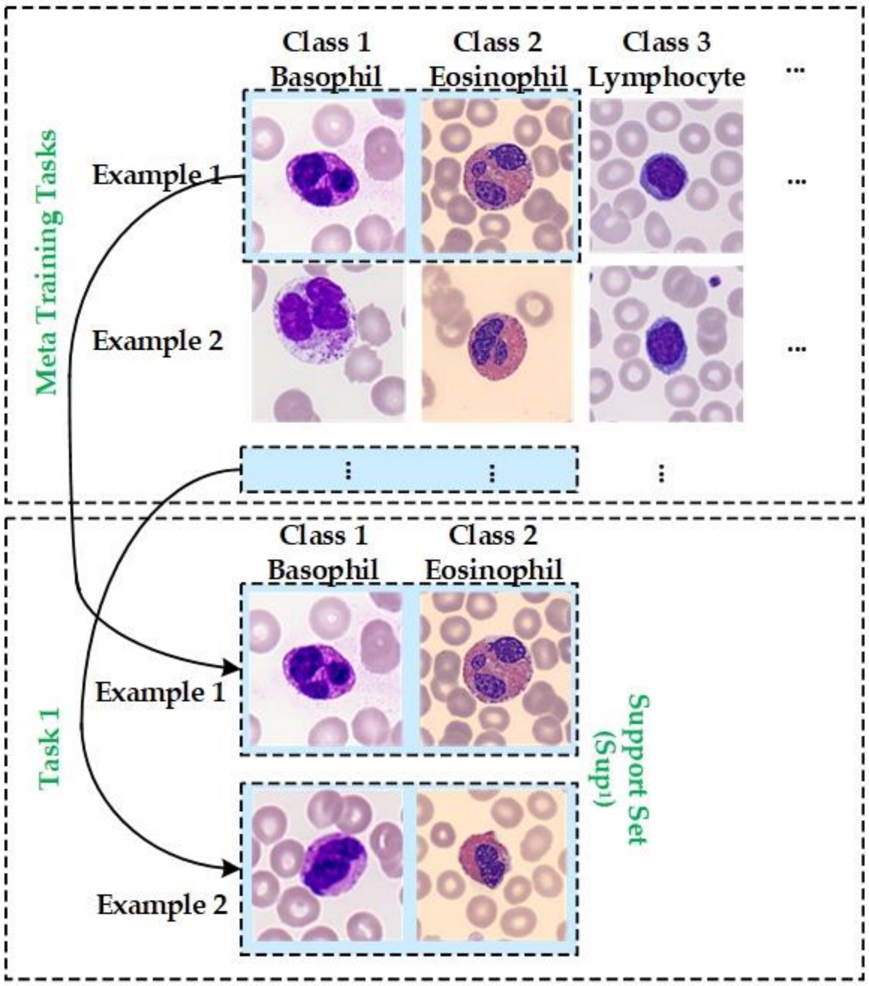

Figure 19: Sampling support set for meta training

Next, for each task we perform sampling of the query set from the meta training dataset.

Images from every class defined in the task are sampled to prepare the query set. The query set contains images that are not present in the support set. Figure 20 shows the example of a query set.

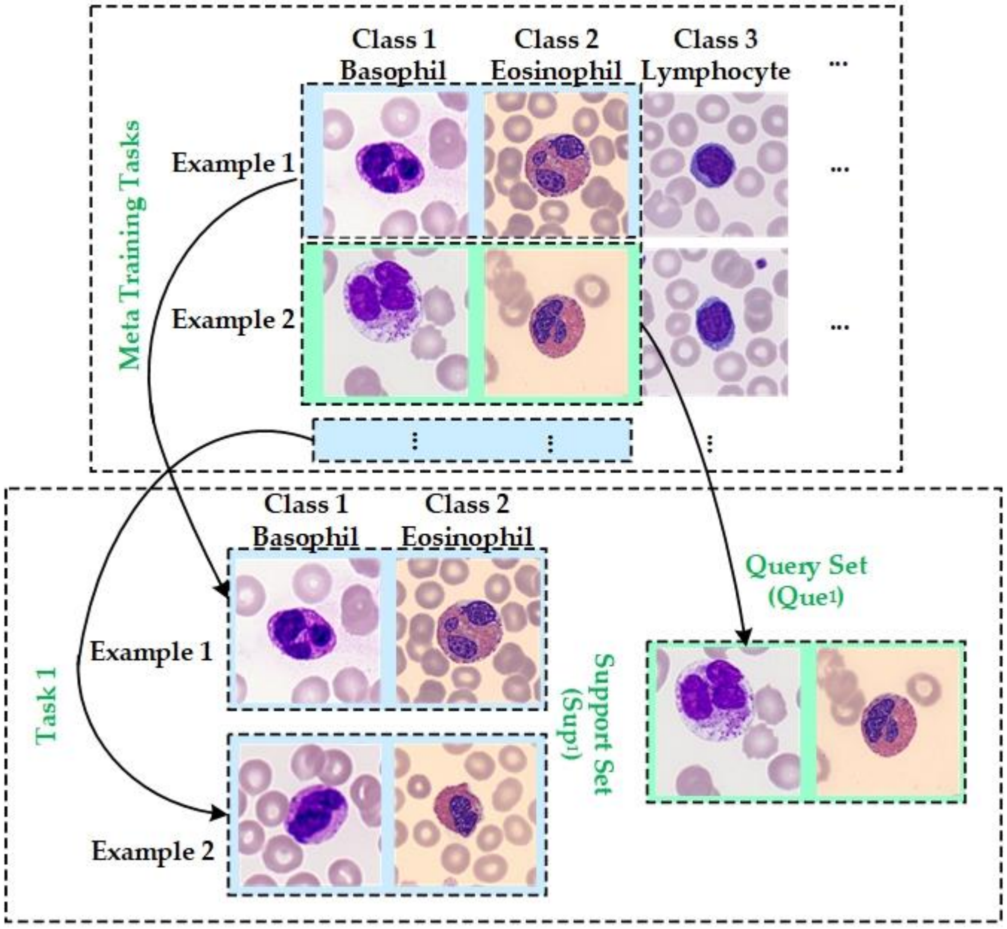

Figure 20: Sampling query set for meta training

The meta training tasks shown in Figure 20 now has the same structure as meta test tasks shown in Figure 17. We train the model on the support set and evaluate the model on query set. After several episodes the model learns to learn the pattern and structure of the few-shot dataset. Through loss calculation, the model can be optimized with the objective as defined in Equation 17. The learnings from the meta training dataset can now be adapted on the meta test dataset to for solving the meta test tasks.

3.4 Stain Normalization

Stains are a necessity for cytological and haematological research studies. Stains are useful for detecting haematological disorders and also irregularities in chromosomes. They are important for differentiating cells of blood and bone marrow samples. Romanowsky stains are used for haematological studies. The name was termed after Dmitri Leonidovich Romanowsky who was the first to invent an eponymous histological stain. There are multiple Romanowsky staining types, and every laboratory might not use the same stain for experiments. Therefore, the microscopic blood cell images obtained from different laboratories vary in color, staining quality and illumination. Figure 21 shows examples from our dataset of how images for the same cell types have different intensities when different stains are used.

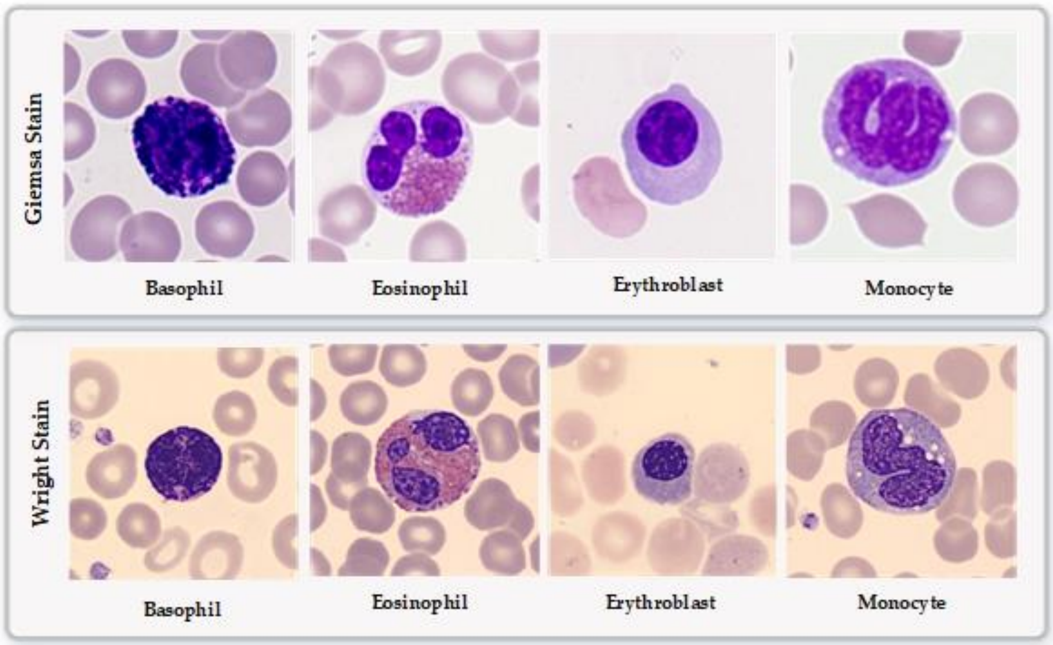

Figure 21: Variation in blood cell images due to application of different stains

In our work, we use the cycle consistency GAN (CycleGAN) architecture for stain normalization. Stain normalization is one of the main steps for enhancing classification of blood cell images obtained from peripheral blood smears. In the evaluation section, we also present an ablation study to show how accuracy is affected when the stain normalization module is removed from our proposed methodology. CycleGAN has become tremendously popular for image-to-image translation. Image-to-image translation is the mapping of an image from one source domain to a target domain. It is also useful for style transfer. Through CycleGAN, we can translate a semantic label to a photorealistic image, change color of images from one color domain to another, synthesize items from outlines, etc. The advantage of CycleGAN is that we do not need paired inputs for training GAN. It can be used for unpaired image-to-image translation. Also, traditional approaches like ensemble models [117] with different reference slides, color matching [118], normal coding, stain separation [119-121] suffers from a larger problem, i.e. it does not consider spatial features. In this scenario where it does not consider the spatial features, the tissue structure is not preserved which leads to an inefficient model output which is not appropriate for real-life usage in the medical field.

CycleGAN uses two generators and two discriminators. One generator G_{AB} takes input of domain A and converts it to domain B, whereas another generator G_{BA} converts images from B to A. A being the input or source image distribution and B is the target output distribution. The two discriminators are defined as D_A and D_B and each generator has a corresponding discriminator. D_A tries to distinguish generated output from real ones for domain A and D_B tries to distinguish generated output from real ones for domain B. Figure 22, presents the architecture of CycleGAN for our proposed work. There are two flows in the Figure 22. The first one being translation from domain $A \rightarrow B \rightarrow A$ and the second one being a translation from domain $B \rightarrow A \rightarrow B$.

In our work, for each cell type, we compare the count in both the datasets. We perform stain normalization for the dataset having a smaller number of images for a particular cell type. For e.g. in Mendeley dataset, basophil has 1218 images and in the Cancer Imaging Archive dataset there are only 79 images for basophil. So, we consider the stain in Cancer Imaging Archive dataset as the source and we perform stain normalization to the target stain i.e. stain in the Mendeley dataset. Count of images for neutrophil cell type in Cancer Imaging Archive dataset is more than the images for neutrophil in the Mendeley Dataset, so we consider the stain used in Mendeley Dataset as the source and the stain in the Cancer Imaging Archive Dataset as the target. For our work, we

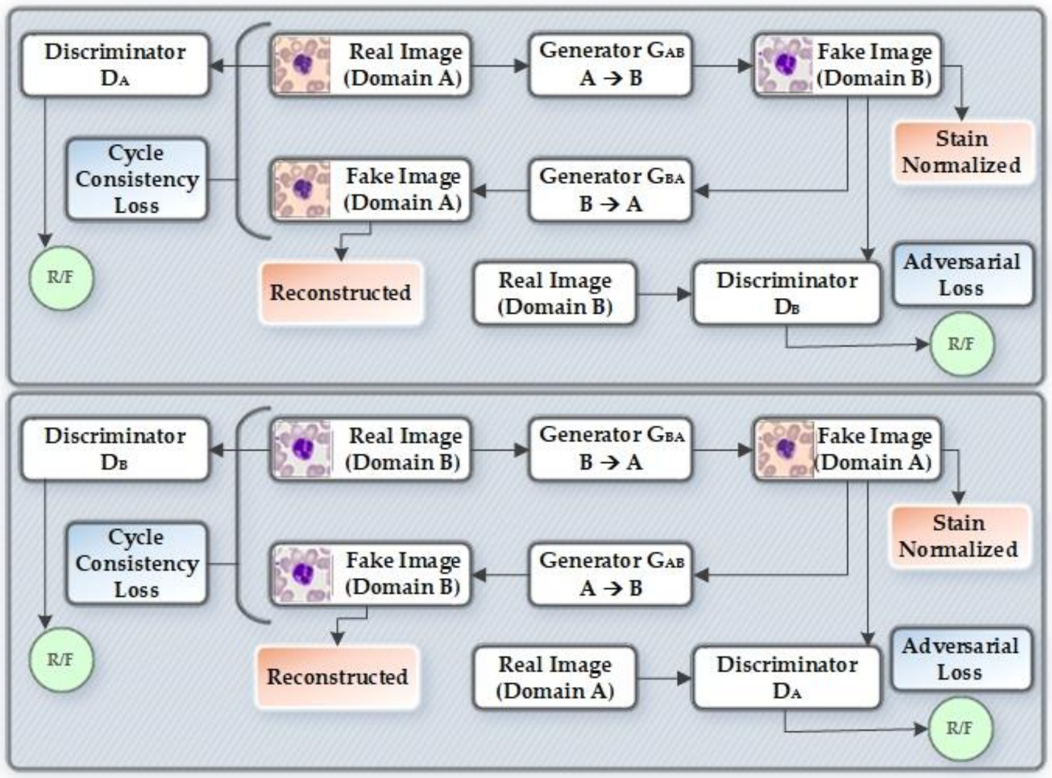

Figure 22: CycleGAN architecture for stain normalization

consider Mendeley dataset as domain A and Cancer Imaging Archive Dataset as domain B.

As can be seen from Figure 22, the image follows two paths. First real image of domain A is fed to the generator GAB which maps the image and normalizes the stain from domain A to domain B. The generated image is then passed to the discriminator DB which evaluates whether the data seems real or not and then the image is again translated back to domain A through generator GBA. The real image of domain A and the fake translated image of domain A is then used to measure cycle consistency loss. In this flow the source is considered as domain A and the target as domain B. In the second flow, same steps are followed but the domain translation or stain normalization is from source domain B to target domain A. Table 9 summarizes the input, output and objective of the two generators and the two discriminators.

Table 9: Summary of objective, input and output of four networks in CycleGAN

Network	Input	Output	Objective
G_{AB}	Images from Mendeley Dataset (Domain A) or Generated images from generator G_{BA}	Translation to Domain B (stains of Cancer Imaging Archive)	Stain Normalization- Realistic images of Domain A with stains of Domain B (Generate image in domain B)
G_{BA}	Images from Cancer Imaging Archive Dataset (Domain B) or Generated images from generator G_{AB}	Translation to Domain A (stains of Mendeley Dataset)	Stain Normalization- Realistic images of Domain B with stains of Domain A (Generate image in domain A)
D_A	Real image of Domain A or Translated image of Domain A	Probability of the image being real	Correctly identify generated samples by G_{BA}
D_B	Real image of Domain B or Translated image of Domain B	Probability of the image being real	Correctly identify generated samples by G_{AB}

For our work we concentrate on three losses of CycleGAN: adversarial loss, cycle consistency loss and identity loss. Adversarial loss is responsible for measuring the probability of an image being from the original data distribution rather than from the generator. It also matches the distribution of the source domain or generated images to the distribution of the second or targeted domain. Adversarial losses are applied to both the mappings i.e. 1) $G_{AB} : A \rightarrow B$ and its discriminator D_B and 2) $G_{BA} : B \rightarrow A$ and its discriminator D_B. The adversarial loss can be formulated as shown in Equation 18 and Equation 19:

For $G_{AB} : A \rightarrow B$

$$\mathrm{Loss}_{\mathrm{Adv}}(G_{AB}, D_B, A, B) = \mathbb{E}_{b \sim p_d(b)}[\log D_B(b)] + \mathbb{E}_{a \sim p_d(a)}\left[\log\left(1 - D_B\left(G_{AB}(a)\right)\right)\right] \quad (18)$$

For $G_{BA} : B \rightarrow A$

$$\mathrm{Loss}_{\mathrm{Adv}}(G_{BA}, D_A, B, A) = \mathbb{E}_{a \sim p_d(a)}[\log D_A(a)] + \mathbb{E}_{b \sim p_d(b)}\left[\log\left(1 - D_A\left(G_{BA}(b)\right)\right)\right] \quad (19)$$

Adversarial loss can help to learn the mapping from source domain to target domain but when the data is large, the input image from the source domain can be mapped to any combination of images in the target domain that follows the distribution of the target domain. So adversarial loss only by itself cannot confirm the mapping of an image from the source domain to a particular desired output in the target domain. Therefore, we require cycle consistency loss that guarantees that an image a from domain A translated to domain B should be able to translate a back to the real image, i.e. $a \rightarrow G_{AB}(a) \rightarrow \mathbf{G_{BA}}(G_{AB}(a)) \approx a$. This is known as the forward cycle consistency. Also, an image b from domain B translated to domain A should be able to translate b back to the real image, i.e. $b \rightarrow G_{BA}(b) \rightarrow G_{AB}(G_{BA}(b)) \approx b$. This is known as the backward cycle consistency. We measure the performance through the cycle consistency loss defined as in Equation 20:

$$\mathrm{Loss}_{\mathrm{cyc}}(G_{AB}, G_{BA}) = \mathbb{E}_{a \sim p_d(a)}\left[\| G_{BA}\left(G_{AB}(a)\right) - a \|_1\right] + \mathbb{E}_{b \sim p_d(b)}\left[\| G_{AB}\left(G_{BA}(b)\right) - b \|_1\right] \quad (20)$$

Therefore, the objective function of a CycleGAN can be defined by combining the adversarial loss and the cycle consistency loss as shown in Equation 21:

$$\mathrm{Loss}(G_{AB}, G_{BA}, D_A, D_B) = \mathrm{Loss}_{Adv}(G_{AB}, D_B, A, B) + \mathrm{Loss}_{Adv}(G_{BA}, D_A, B, A) + \lambda \mathrm{Loss}_{cyc}(G_{AB}, G_{BA}) \qquad (21)$$

We have two objectives as defined in Equation 22 and λ denotes the relative importance of it.

$$G_{AB}{}^{*}, G_{BA}{}^{*} = \arg \min_{G_{AB}, G_{BA}} \max_{D_A, D_B} \mathrm{Loss}(G_{AB}, G_{BA}, D_A, D_B). \qquad (22)$$

In our work, we also use the identity mapping or the identity loss along with the overall loss to preserve the color intensities between the source and the target stains. The identity loss is defined as in Equation 23:

$$\mathrm{Loss}_{identity}(G_{AB}, G_{BA}) = \mathbb{E}_{b \sim p_d(b)}[\| G_{AB}(b) - b \|_1] + \mathbb{E}_{x \sim p_d(a)}[\| G_{BA}(a) - a \|_1] \qquad (23)$$

CycleGAN uses least-squares [30] loss instead of the negative log likelihood objective defined in Equation 18 and 19. For our proposed work we improve the training of CycleGAN by introducing two changes: 1) We introduce WGAN-GP to the loss function of CycleGAN to improve the stability of the training, 2) We implement a U-Net architecture for the generator instead of a vanilla ResNet architecture.

The training in the Vanilla GAN led to various problems like mode collapse, vanishing gradient, difficulty in convergence (Nash Equilibrium), low dimensional support etc. Also, KL and JS divergence value failed to produce any meaning for disjoint distributions. Wasserstein distance also known as the Earth Mover's distance is also a measure that provides the distance between two probability distribution. The Wasserstein distance performs better than KL divergence or JS divergence because it can produce significant representation of the distance between two probability distributions even if they are not overlapping in the lower dimensional manifold. The WGAN model takes advantage of the Wasserstein distance which comprises of continuity properties and differentiability properties, and introduces Wasserstein Loss function. The Wasserstein distance between the distribution of real image R and the generated image S can be defined as in Equation 24:

$$\mathrm{WD}(\mathrm{R},\mathrm{S}) = \sup_{|f|_{L\leq 1}} \mathbb{E}_{y\sim R}[f(y)] - \mathbb{E}_{\tilde{y}\sim S}[f(\tilde{y})] \quad (24)$$

Where sup represents supremum considered over all the functions f which should be 1-Lipschitz continuous. Y in the function f: Y→ℝ is a compact metric space. In WGAN the discriminator is represented through the function f and it does not work as a classifier or a direct critic. The discriminator helps to estimate the Wasserstein distance between the original and the generated data distribution. The objective function of the WGAN can be defined as in Equation 25:

$$\min_{G} \max_{D\in L} \mathbb{E}_{y\sim R}[D(y)] - \mathbb{E}_{\tilde{y}\sim S}[D(\tilde{y})] \quad (25)$$

Where L represents 1- Lipschitz functions. But, to add the Lipschitz constraint to the discriminator, a weight clipping method is used, where the weights of the discriminator are clipped in a compact space $[c, -c]$ and are limited to it. In WGAN few changes were made with respect to the original GAN training. The changes were: logarithm was not used anymore for loss function in WGAN, weight clipping for the discriminator in WGAN, sigmoid was removed from the last layer of the discriminator and RMSProp was used instead of Adam optimizer. Although WGAN eliminated the problem of Vanilla GAN, the weight clipping method led to optimization difficulties. Therefore, to solve this issue, WGAN with gradient penalty (WGAN-GP) was introduced by [122]. WGAN-GP replaces the weight clipping by complying with the condition of 1-Lipschitz and by enforcing a penalty on the gradient. The objective of WGAN-GP can be defined as the combination of original critic loss and gradient penalty as shown in below Equation 26:

$$\begin{aligned}\mathrm{Loss}(G,D,Y) = {} & \underbrace{\mathbb{E}_{G(z)\sim S}\left[D\big(G(z)\big)\right] - \mathbb{E}_{y\sim R}[D(y)]}_{\text{Original critic loss}} \\ & + \underbrace{\partial \mathbb{E}_{\hat{y}\sim \mathbb{P}_{\hat{y}}}\left[\left(\left\|\nabla_{\hat{y}} D(\hat{y})\right\|_2 - 1\right)^2\right]}_{\text{Gradient penalty}}\end{aligned} \quad (26)$$

Where ∂ is the penalty coefficient, $\hat{\mathbf{y}} \sim \mathbb{P}_{\hat{y}}$ are random samples and $\mathbb{P}_{\hat{y}}$ is responsible for uniformly sampling between pairs of points sampled from real and generated data distribution along a straight line. WGAN-GP has tremendously improved the training of GANs.

For incorporating WGAN-GP in CycleGAN we have made the following changes: removed sigmoid function from the last layer of the discriminator, instead of RMSProp we have used Adam optimizer, eliminated logarithm for generator and discriminator loss function, applied gradient penalty on the loss function of the discriminator and removed batch normalization.

So, the adversarial losses of CycleGAN for our proposed work can be defined as in Equation 27 and Equation 28:

$$\mathrm{AdvLoss_{WGAN}}(\mathrm{G_{AB}},\mathrm{D_B},\mathrm{A},\mathrm{B}) = \mathbb{E}_{\mathrm{b}\sim \mathrm{p_d(b)}}[\mathrm{D_B}(\mathrm{b})] - \mathbb{E}_{\mathrm{a}\sim \mathrm{p_d(a)}}[\mathrm{D_B}(\mathrm{G_{AB}}(\mathrm{a})] \quad (27)$$

$$\mathrm{AdvLoss_{WGAN}}(\mathrm{G_{BA}},\mathrm{D_A},\mathrm{B},\mathrm{A}) = \mathbb{E}_{\mathrm{a}\sim \mathrm{p_d(a)}}[\mathrm{D_A}(\mathrm{a})] - \mathbb{E}_{\mathrm{b}\sim \mathrm{p_d(b)}}[\mathrm{D_A}(\mathrm{G_{BA}}(\mathrm{b})] \quad (28)$$

And after combining the cycle consistency loss, the W-CycleGAN objective can be defined as shown in Equation 29:

$$\begin{aligned}\mathrm{Loss}(\mathrm{G_{AB}},\mathrm{G_{BA}},\mathrm{D_A},\mathrm{D_B}) &= \mathrm{AdvLoss_{WGAN}}(\mathrm{G_{AB}},\mathrm{D_B},\mathrm{A},\mathrm{B}) + \mathrm{AdvLoss_{WGAN}}(\mathrm{G_{BA}},\mathrm{D_A},\mathrm{B},\mathrm{A}) \\ &+ \lambda \mathrm{Loss_{cyc}}(\mathrm{G_{AB}},\mathrm{G_{BA}}) \qquad (29)\end{aligned}$$

We improve the performance of W-CycleGAN by introducing gradient penalty. The objective of W-CycleGAN-GP can be defined as shown in Equation 30:

$$\begin{aligned}\mathrm{Loss}(\mathrm{G_{AB}},\mathrm{G_{BA}},\mathrm{D_A},\mathrm{D_B}) &= \mathrm{AdvLoss_{WGAN}}(\mathrm{G_{AB}},\mathrm{D_B},\mathrm{A},\mathrm{B}) + \mathrm{AdvLoss_{WGAN}}(\mathrm{G_{BA}},\mathrm{D_A},\mathrm{B},\mathrm{A}) \\ &+ \lambda \mathrm{Loss_{cyc}}(\mathrm{G_{AB}},\mathrm{G_{BA}}) + \partial_1 \mathop{\mathbb{E}}_{\hat{\mathrm{a}}_1\sim \mathrm{p}_{\mathrm{d}_{\hat{\mathrm{a}}_1}}}\left[\left(\left\|\nabla_{\hat{\mathrm{a}}_1}\mathrm{D_B}(\hat{\mathrm{a}}_1)\right\|_2 - 1\right)^2\right] \\ &+ \partial_2 \mathop{\mathbb{E}}_{\hat{\mathrm{a}}_2\sim \mathrm{p}_{\mathrm{d}_{\hat{\mathrm{a}}_2}}}\left[\left(\left\|\nabla_{\hat{\mathrm{a}}_2}\mathrm{D_A}(\hat{\mathrm{a}}_2)\right\|_2 - 1\right)^2\right] \qquad (30)\end{aligned}$$

Where λ is the relative importance for cycle consistency loss, $\hat{a}_1 \sim p_{d_{\hat{a}_1}}$ and $\hat{a}_2 \sim p_{d_{\hat{a}_2}}$ are random samples, ∂_1 and ∂_2 are the penalty coefficient and $p_{d_{\hat{a}_1}}$and $p_{d_{\hat{a}_2}}$ are responsible for uniformly sampling between pairs of points sampled from real and generated data distribution along a straight line.

In general, the CycleGAN generator consists of an encoder, a transformer and a decoder as shown in Figure 23. The encoder is built with three convolutional layers which downsamples the input image and increases the number of channels. The transformer uses few residual blocks (ResNet) to transform the image and the decoder uses convolutional blocks for upsampling and generating the output image.

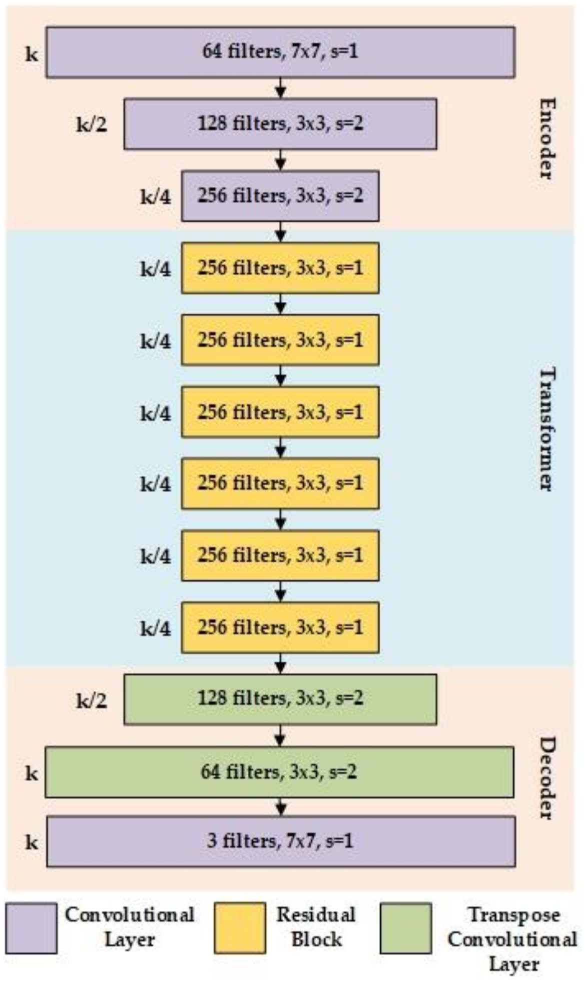

Figure 23: CycleGAN generator architecture

In our work, we have used a U-Net architecture that uses convolutional layers to design the encoders. We have skip connections in between convolutional blocks so that information can traverse easily and quickly. Then for the decoder we have used deconvolutional blocks with one convolutional layer for upsampling the image and for producing the output image of the same size as the original image. Figure 24 presents the U-Net architecture for the generator network of W-CycleGAN-GP. For U-Net we have used standard 2D convolutional layers, LeakyReLU activation and instance normalization instead of batch normalization, transposed convolution and created skip connections between the downsample and the upsample blocks as shown in Figure 24.

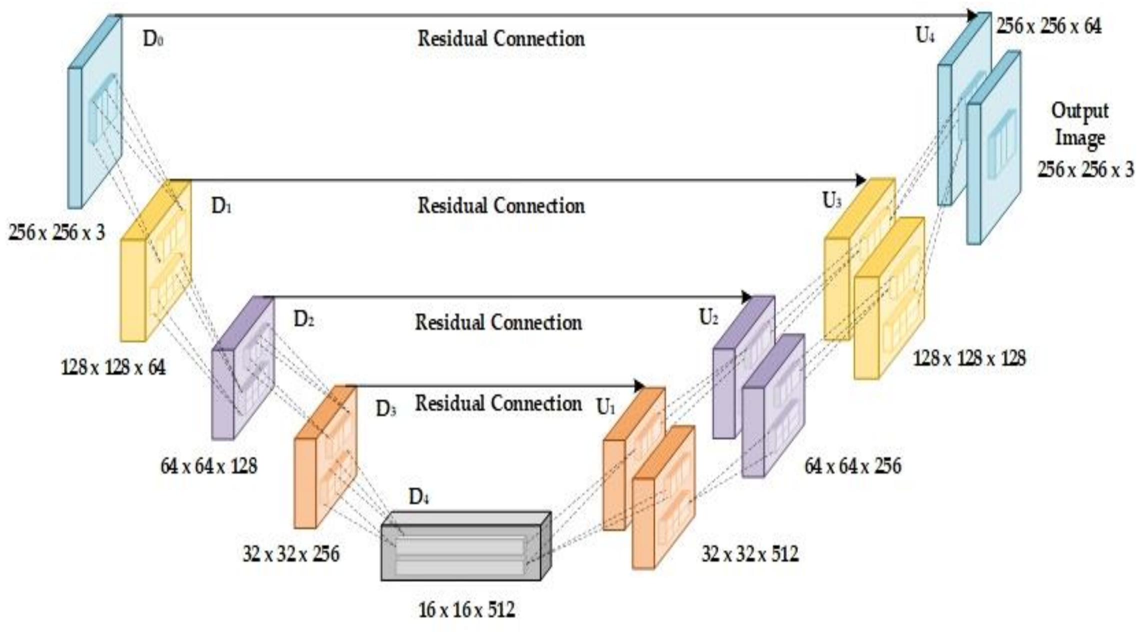

Figure 24: Proposed U-Net generator architecture for CycleGAN

We have used same discriminator architecture as CycleGAN which is based on the PatchGAN [123] architecture. It uses fully convolutional neural networks that examines a patch of the input image and produces the probability of the patch being original or real. This approach is considered to be more efficient computationally as compared to evaluating an entire image. This allows the discriminator to concentrate more on intricate details like texture that is mostly modified during translation of images. PatchGAN discriminator does not output a probability like conventional discriminators. PatchGAN discriminator predicts for each patch of the whole image whether it is real or not. The predictions are then averaged to generate the output or is tallied to a matrix of expected values i.e. zero or one. The discriminator architecture uses 2D convolutional blocks, LeakyReLU activation and instance normalization. The PatchGAN based discriminator model in our work (W-CycleGAN-GP) is presented in Figure 25.

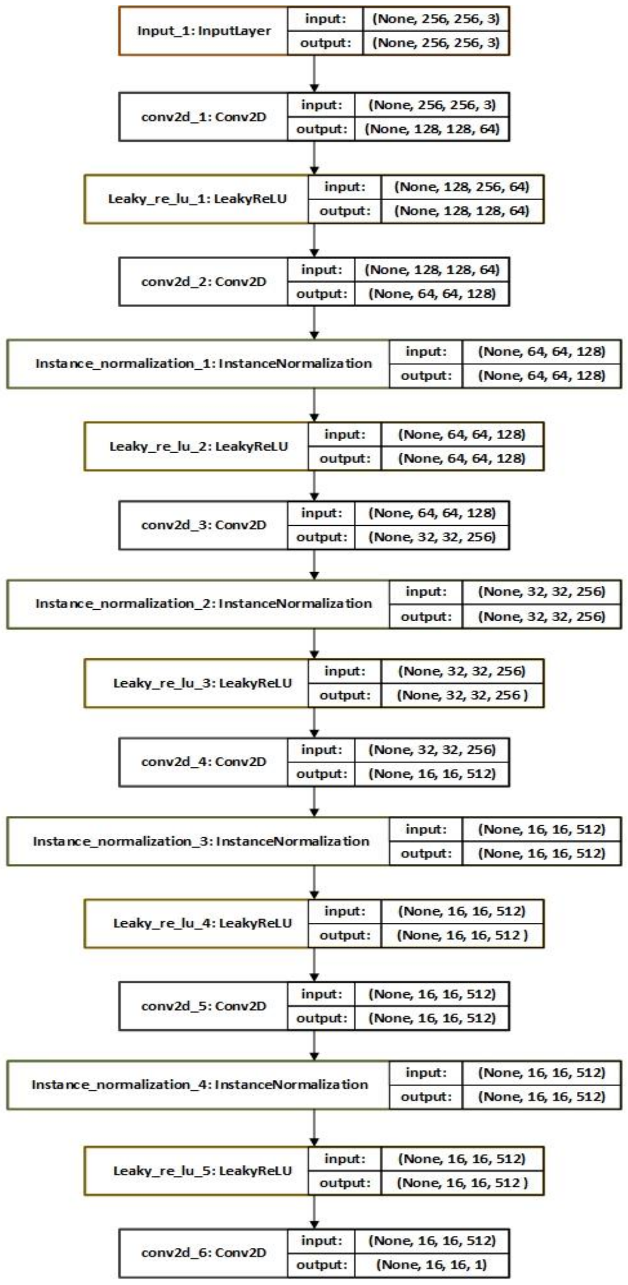
Input_1: InputLayer | input: (None, 256, 256, 3) | output: (None, 256, 256, 3)
conv2d_1: Conv2D | input: (None, 256, 256, 3) | output: (None, 128, 128, 64)
Leaky_re_lu_1: LeakyReLU | input: (None, 128, 256, 64) | output: (None, 128, 128, 64)
conv2d_2: Conv2D | input: (None, 128, 128, 64) | output: (None, 64, 64, 128)
Instance_normalization_1: InstanceNormalization | input: (None, 64, 64, 128) | output: (None, 64, 64, 128)
Leaky_re_lu_2: LeakyReLU | input: (None, 64, 64, 128) | output: (None, 64, 64, 128)
conv2d_3: Conv2D | input: (None, 64, 64, 128) | output: (None, 32, 32, 256)
Instance_normalization_2: InstanceNormalization | input: (None, 32, 32, 256) | output: (None, 32, 32, 256)
Leaky_re_lu_3: LeakyReLU | input: (None, 32, 32, 256) | output: (None, 32, 32, 256)
conv2d_4: Conv2D | input: (None, 32, 32, 256) | output: (None, 16, 16, 512)
Instance_normalization_3: InstanceNormalization | input: (None, 16, 16, 512) | output: (None, 16, 16, 512)
Leaky_re_lu_4: LeakyReLU | input: (None, 16, 16, 512) | output: (None, 16, 16, 512)
conv2d_5: Conv2D | input: (None, 16, 16, 512) | output: (None, 16, 16, 512)
Instance_normalization_4: InstanceNormalization | input: (None, 16, 16, 512) | output: (None, 16, 16, 512)
Leaky_re_lu_5: LeakyReLU | input: (None, 16, 16, 512) | output: (None, 16, 16, 512)
conv2d_6: Conv2D | input: (None, 16, 16, 512) | output: (None, 16, 16, 1)

Figure 25: Plot of discriminator model of CycleGAN

3.5 Classifier Based Generative Adversarial Networks

Conditional GAN introduced in [124] is a variation of GAN which conditions the generator and the discriminator of a GAN model on class labels or specific conditions. We can provide any auxiliary information like labels of data or data of other modalities as the extra information C in the GAN architecture. The condition C is forwarded as input to both the generator network and the discriminator network. The structure of conditional GAN model is shown in Figure 26.

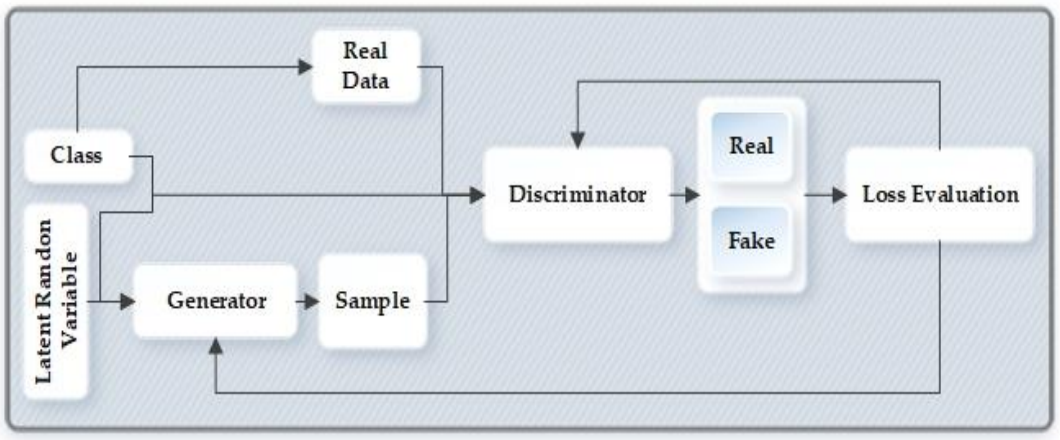

Figure 26: Conditional generative adversarial networks

The generator receives the combination of an input noise vector P(z) and the condition C. The input to the generator is in the form of a hidden representation. The discriminator receives the real image y and the generated image as input. The discriminator is also conditioned on the extra information. The objective function of the conditional GAN can be described as in Equation 31:

$$\min_G \max_D V(D,G) = \mathbb{E}_{y\sim p_d(y)}[\log D(y \mid C)] + \mathbb{E}_{z\sim P(z)}\left[\log\left(1 - D\left(G(z \mid C)\right)\right)\right] \quad (31)$$

In conditional GAN the discriminator cannot output the category or labels of data directly. To get the data category, the data should be provided as input by mentioning each category one by one. Also, conditional GAN fails to provide intricate features or images with detailed and specific labels. In auxiliary classifier GAN (AC-GAN) [125], the generator receives an input noise vector along with a condition. The generator produces a fake sample which is forwarded to the discriminator. The discriminator receives the real image and the fake image. The discriminator in AC-GAN performs two tasks. The discriminator has to produce the probability distribution of the data being real and also output the probability of the image belonging to a specific class. AC-GAN defines a classifier on top of the discriminator, so the discriminator has two tasks. One is to distinguish generated image from a real image, and the second is to predict the class labels of the image. Figure 27 shows the structure of the auxiliary classifier GAN.

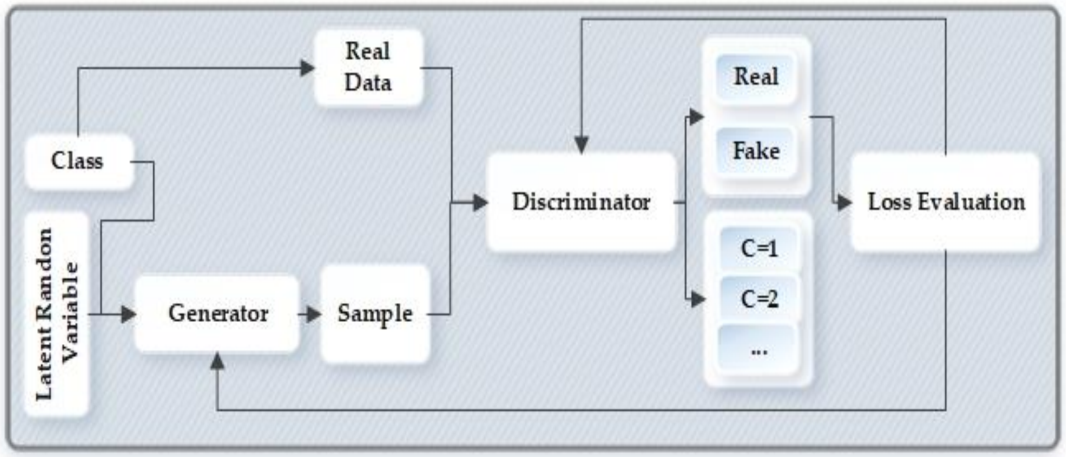

Figure 27: Auxiliary classifier generative adversarial networks (AC-GAN)

The objective function of AC-GAN can be represented with the log-likelihood of the image being from the real (R) distribution (L_{Source}) and the log-likelihood of the image being from correct class label (L_{Class}). They can be described with Equation 32 and Equation 33.

$$L_{Source} = E[\log P(Source = R \mid X_R)] + E[\log P(Source = F \mid X_F)] \quad (32)$$

$$L_{Class} = E\log P(Class = C \mid X_R)] + E[\log P(Class = C \mid X_F)] \quad (33)$$

Where F denotes the generated data. The discriminator in AC-GAN is not provided with the class label. But due to the sharing of weight parameters in the discriminator architecture AC-GAN has a complex and limited training process. Following conditional generation from conditional GAN and classifier-based discriminator in AC-GAN, in our work we have proposed a classifier-based GAN architecture C-WGAN-GP which is a three-player GAN architecture consisting of a generator, discriminator and a classifier. We have used WGAN-GP loss for our model training. We have separated the classifier from the discriminator, and the generator is simultaneously trained from the feedback of the classifier and the discriminator. The generator tries to fool the discriminator and simultaneously targets to generate samples that can be classified correctly by the classifier. Figure 28 presents our proposed classifier-based GAN architecture.

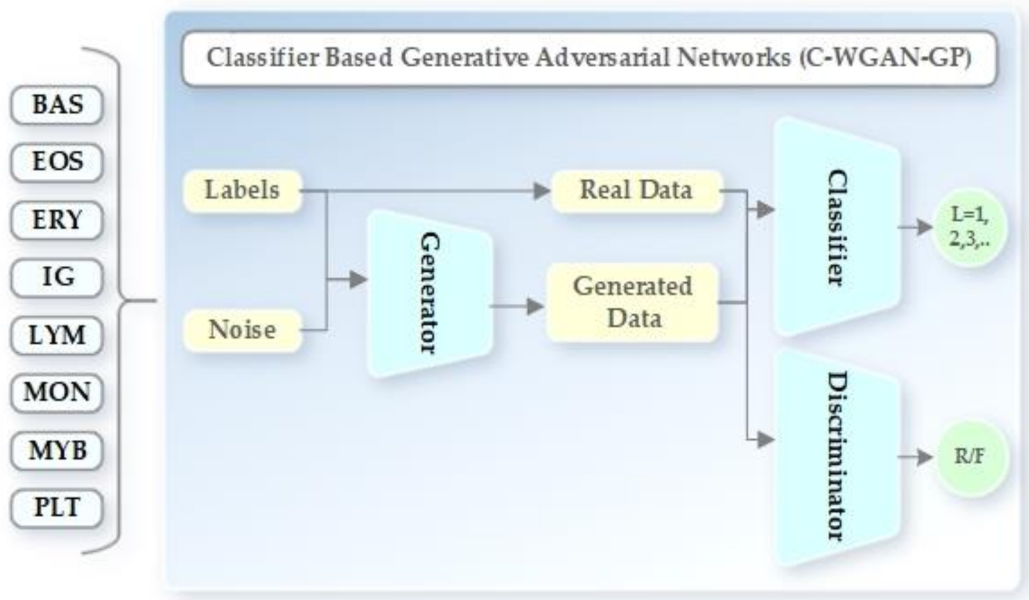

Figure 28: Classifier-based generative adversarial networks (C-WGAN-GP)

In our work, we use the classifier-based GAN model to generate images of cell types consisting more than thousand images each. There are eight cell types with more than thousand images and they are: basophil, eosinophil, erythroblast, immature granulocytes, lymphocyte, monocyte, myeloblast and platelet. We have enough data for neutrophil, so we did not consider neutrophil cell type for image generation. The structure of the generator, discriminator and the classifier are presented in Table 10, Table 11 and Table 12. We use two hidden deconvolutional layer for each residual block. We have used LeakyReLU for the activation function.

Table 10: Network architecture details of the generator

Blocks	Size of Kernel	Shape of Output
Input Concatenate(z,C)		
Fully Connected	-	64 x 32 x 32
Residual Block (1,2)	3 x 3	64 x 32 x 32
Deconvolution (s=2)	5 x 5	64 x 64 x 64
Residual Block (3-6)	3 x 3	64 x 64 x 64
Deconvolution (s=2)	5 x 5	64 x 128 x 128
Residual Block (7,8)	3 x 3	64 x 128 x 128
Deconvolution (s=1)	5 x 5	64 x 128 x 128

Table 11: Network architecture details of the discriminator

Blocks	Size of Kernel	Shape of Output
Input (X, G(z,C))		
Convolution (s=2)	5 x 5	64 x 64 x 64
Residual Block (1,2)	3 x 3	64 x 64 x 64
Average Pooling	2 x 2	64 x 32 x 32
Residual Block (3-6)	3 x 3	64 x 32 x 32
Average Pooling	2 x 2	64 x 16 x 16
Residual Block (7-10)	3 x 3	64 x 16 x 16
Average Pooling	2 x 2	64 x 8 x 8
Fully Connected	-	128
Fully Connected	-	1

Table 12: Network architecture details of the classifier

Blocks	Size of Kernel	Shape of Output
Input (X, G(z,C))		
Convolution (s=2)	5 x 5	64 x 64 x 64
Residual Block (1,2)	3 x 3	64 x 64 x 64
Average Pooling	2 x 2	64 x 32 x 32
Residual Block (3-6)	3 x 3	64 x 32 x 32
Average Pooling	2 x 2	64 x 16 x 16
Residual Block (7-10)	3 x 3	64 x 16 x 16
Average Pooling	2 x 2	64 x 8 x 8
Fully Connected	-	128
Fully Connected	-	8

3.6 Few-Shot Image Generation

In this section, we generate images for the cell types containing less than hundred images through meta learning based few-shot image generation. The cell types for which we have introduced few-shot image generation are monoblast, promyelocyte and smudge cells. We have followed the meta learning architecture of [126], but modified the generator architecture for better performance, high resolution images and to build a light weight model. The few-shot image generation model is an encoder-based GAN model which has a training phase and a generation phase. In the training phase we take the dataset we prepared from classifier-based GAN model as input to the encoder. In the training phase we have training tasks denoted as Task I_n and in the generating phase we have testing tasks denoted as Task J_n. The testing tasks are unseen and are not present in the training tasks just like the concept of meta learning. Both the training and generating phase has two loops: inner and outer. First the encoder En_1 in the inner loop of the training phase receives the input image a from the training tasks. The encoder downsamples the input into a feature vector FV, FV= $En_1(a)$. The feature vector is combined with two random noise vectors denoted as z_1 and z_2. There are two random noise vectors to eliminate mode collapse problem in GAN as was introduced in mode seeking GAN [36]. Then the two combined random noise with feature vector is passed to the generator (Gen) as input. The generator outputs synthetic images $b_1=Gen_1(z_1, r)$ and $b_2=Gen_1(z_2, r)$. The generated images are then passed to the discriminator. The discriminator Dis_1 tries to distinguish between the real image (a) and generated image (b). Once all the iteration of the inner loop is completed the outer loop is activated. The outer loop consists of the second encoder (En_2), generator (Gen_2) and the discriminator (Dis_2). By setting the gradient $\theta = \theta - Wei_{Task}$, the gradients are updated for the encoder, generator and discriminator in the outer loop. Figure 29 presents the layout of the training phase for few-shot image generation.

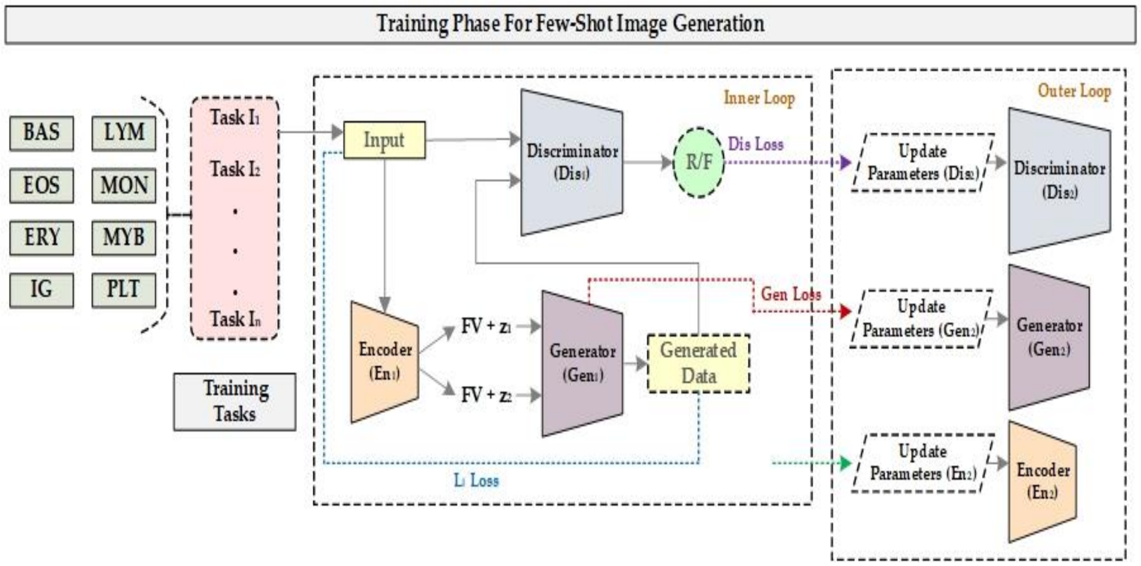

Figure 29: Training phase for few-shot image generation

In the generating phase, we work with the testing task Task Jn. The parameters of the encoder, generator and the discriminator from the training phase are adapted for training the testing task in the generating phase. In the generating phase, we use the cell types having less than hundred images. We get the generated images for monoblast, promyelocyte and smudge cells from the generating phase. Figure 30 presents the layout of the generating phase for few-shot image generation.

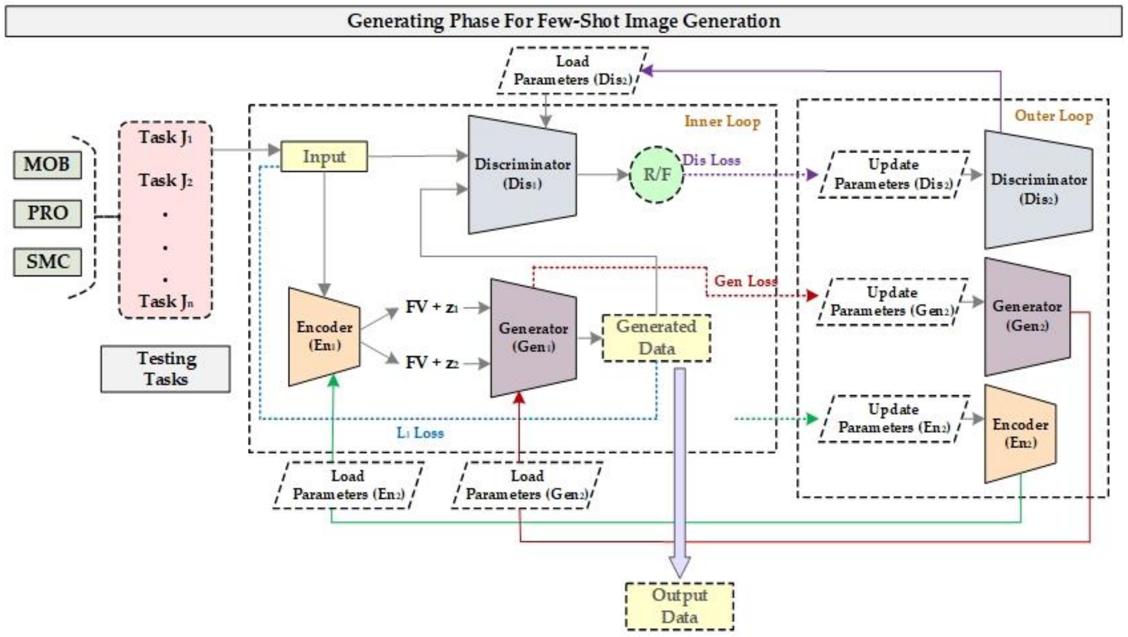

Figure 30: Generating phase for few-shot image generation

We follow the algorithms of [126] to train the overall encoder-based GAN network for few-shot image generation. Since we have considered two random variable z_1 and z_2, we define the objective function of the discriminator as shown in Equation 34, Equation 35 and Equation 36.

$$\mathrm{Loss}_{\mathrm{Dis}} = \frac{\mathrm{Loss}_{\mathrm{Dis}_1} + \mathrm{Loss}_{\mathrm{Dis}_2}}{2} \tag{34}$$

Where,

$$\mathrm{Loss}_{\mathrm{Dis}_1} = -\mathbb{E}_{a_i \sim a}\left[\log(\mathrm{Dis}_1(a_i)\right] - \mathbb{E}_{b_{1i} \sim b_1}[\log(1 - \mathrm{Dis}_1(b_{1i})] \tag{35}$$

$$\mathrm{Loss}_{\mathrm{Dis}_2} = -\mathbb{E}_{a_i \sim a}\left[\log(\mathrm{Dis}_2(a_i)\right] - \mathbb{E}_{b_{2i} \sim b_2}[\log(1 - \mathrm{Dis}_2(b_{2i})] \tag{36}$$

Where i is the index of the image i.e. from the input a and the output b of every task, we sample

I images and $i \in [0,I]$. Similarly, the objective function of the generator can be defined as shown in Equation 37, Equation 38 and Equation 39.

$$\mathrm{Loss_{Gen}} = \frac{\mathrm{Loss_{Gen_1}} + \mathrm{Loss_{Gen_2}}}{2} \qquad (37)$$

Where,

$$\mathrm{Loss_{Gen_1}} = -\mathbb{E}_{b_{1i}\sim b}[\log(\mathrm{Dis_1}(b_{1i})] - \mathbb{E}_{a_i\sim a}[\log(1-\mathrm{Dis_1}(a_i)] \qquad (38)$$

$$\mathrm{Loss_{Gen_2}} = -\mathbb{E}_{b_{2i}\sim b}[\log(\mathrm{Dis_1}(b_{2i})] - \mathbb{E}_{a_i\sim a}[\log(1-\mathrm{Dis_1}(a_i)] \qquad (39)$$

Also, in the same way we can define the objective function of the encoder through average of two encoders as shown in Equation 40, Equation 41 and Equation 42.

$$\mathrm{Loss_{En}} = \frac{\mathrm{Loss_{En_1}} + \mathrm{Loss_{En_2}}}{2} \qquad (40)$$

Where,

$$\mathrm{Loss_{En_1}} = \sum_{i=1}^{I} \|a_i - b_{1i}\| \qquad (41)$$

$$\mathrm{Loss_{En_2}} = \sum_{i=1}^{I} \|a_i - b_{2i}\| \qquad (42)$$

The weights of the generator WeiGen, discriminator WeiDis and the encoder WeiEn for every sampled training task Task In and generating task Task Jn are initialized and minimized in the inner loop of the model through the objective functions $\mathrm{Loss_{Gen}}$, $\mathrm{Loss_{Dis}}$ and $\mathrm{Loss_{En}}$. In the outer loop we update the global parameters θ_{Gen}, θ_{Dis} and θ_{En} of the generator, discriminator and the encoder by minimizing or reducing the distance between the optimized weights and the initial parameters of the generator, discriminator and the encoder (θ_{Gen}, θ_{Dis} and θ_{En}) from the inner loop. We denote it through the following Equation 43:

$$\mathrm{minimise} \sum(\theta_{\mathrm{Dis}} - \mathrm{Wei_{DisTask}}) + (\theta_{\mathrm{Gen}} - \mathrm{Wei_{GenTask}}) + (\theta_{\mathrm{En}} - \mathrm{Wei_{EnTask}}) \qquad (43)$$

Therefore, the overall loss function can be defined as shown in Equation 44:

$$\mathrm{Loss} = \mathrm{Loss_{Dis}} + \mathrm{Loss_{Gen}} + \lambda_{\mathrm{En}}\mathrm{Loss_{En}} + \lambda_{\mathrm{MSR}}\mathcal{L}_{\mathrm{MSR}} \qquad (44)$$

Where, $\mathcal{L}_{\mathrm{MSR}}$ denotes the regularization term defined in mode seeking GAN and the hyperparameter λ_{En} and λ_{MSR} are set to one following [127].

We have built the generator of the meta learning model following [128]. The generator consists of channel-wise attention along with skip layer excitation that performs larger skip connections than ResNet. Figure 31 represents the architecture of the generator for few-shot image generation. The combination of the encoder's feature vector and the two random noise is fed to the fully connected layer, which is then passed through the transposed convolution with batch normalization and gated linear unit [129]. The grey boxes in Figure 31 represents the feature maps with the spatial size and as can be seen the boxes follows up-sampling of the image. The green boxes represent the skip layer excitation module which is described in Figure 32. Instead of element-wise addition of skip connection between the activations of same special dimension as in residual blocks, we use channel-wise multiplication for the skip connection. Also, it is not required that the spatial dimension should be same. As we can see from Figure 31, the skip layer excitation

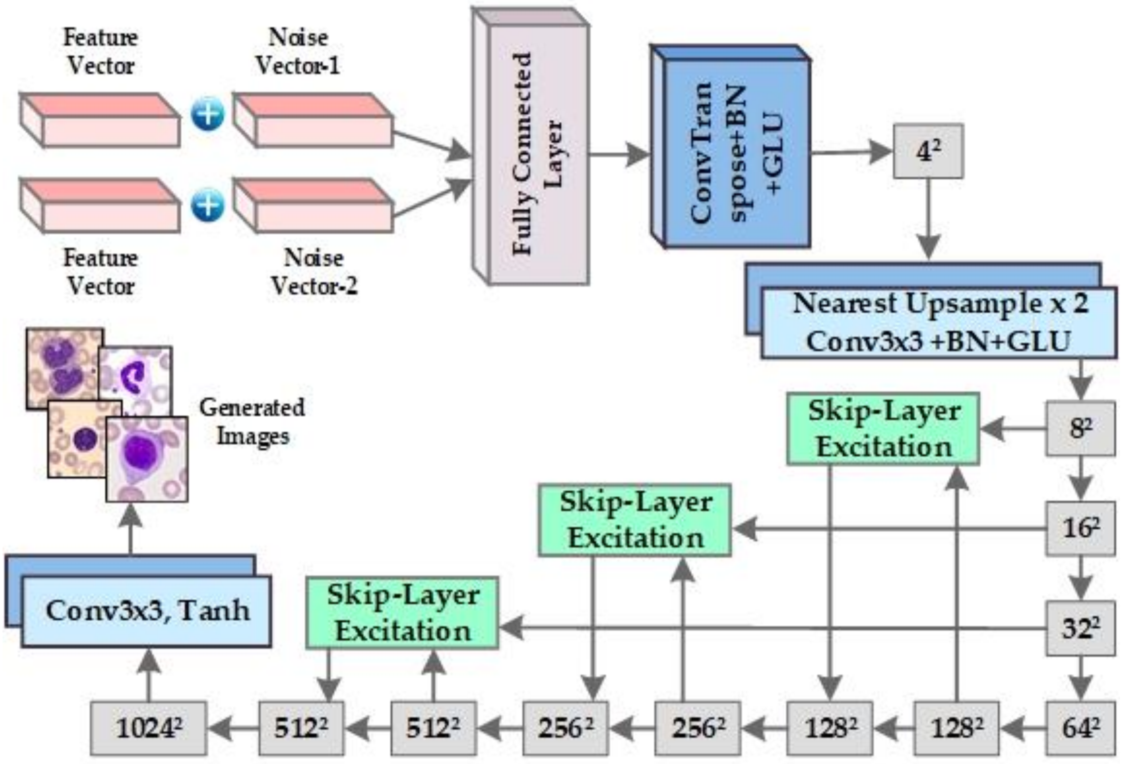

Figure 31: Generator architecture for few-shot image generation

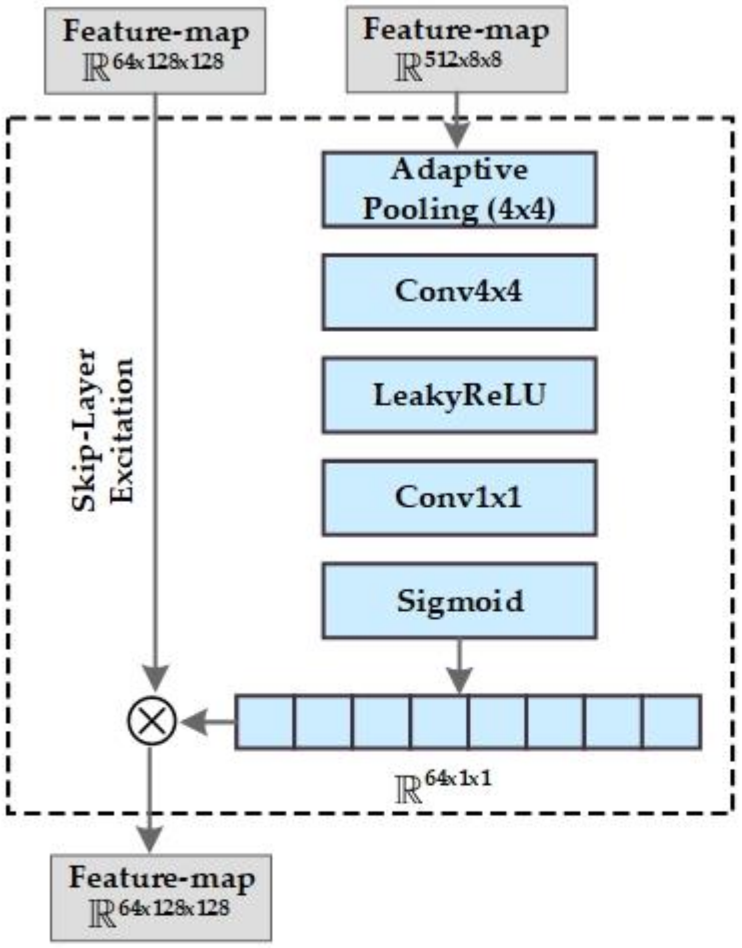
Feature-map $\mathbb{R}^{64\times128\times128}$
Feature-map $\mathbb{R}^{512\times8\times8}$
Adaptive Pooling (4x4)
Conv4x4
LeakyReLU
Conv1x1
Sigmoid
Skip-Layer Excitation
$\mathbb{R}^{64\times1\times1}$
Feature-map $\mathbb{R}^{64\times128\times128}$

Figure 32: Skip layer excitation module of the generator network

module is between different resolution size like 8^2 and 128^2, 16^2 and 256^2, etc. The skip layer excitation module can be described as shown in Equation 45:

$$\mathrm{SLE} = \mathcal{F}(\mathrm{I}_{\mathrm{low}}, \{\mathrm{Wei}\}) \cdot \mathrm{I}_{\mathrm{high}} \qquad (45)$$

Where I represent input feature map, SLE represents output feature maps of the skip layer excitation module. Wei is the weight of the module and $\mathcal{F}$ denotes the operations on the input feature maps of low resolution $\mathrm{I}_{\mathrm{low}}$. In Figure 32, the skip layer excitation module is shown where $\mathrm{I}_{\mathrm{low}}$ is 8x8 resolution and $\mathrm{I}_{\mathrm{high}}$ is 128x128 resolution. First $\mathrm{I}_{\mathrm{low}}$ is downsampled through adaptive average pooling to 4x4, then a convolutional layer is used to downsample it further. The nonlinearity is modeled through LeakyReLU, after a 1x1 convolution, a sigmoid function is used and the output from the $\mathcal{F}$ operation is multiplied with I_{high} along the dimension of the channel to obtain SLE that has the same resolution as I_{high}.

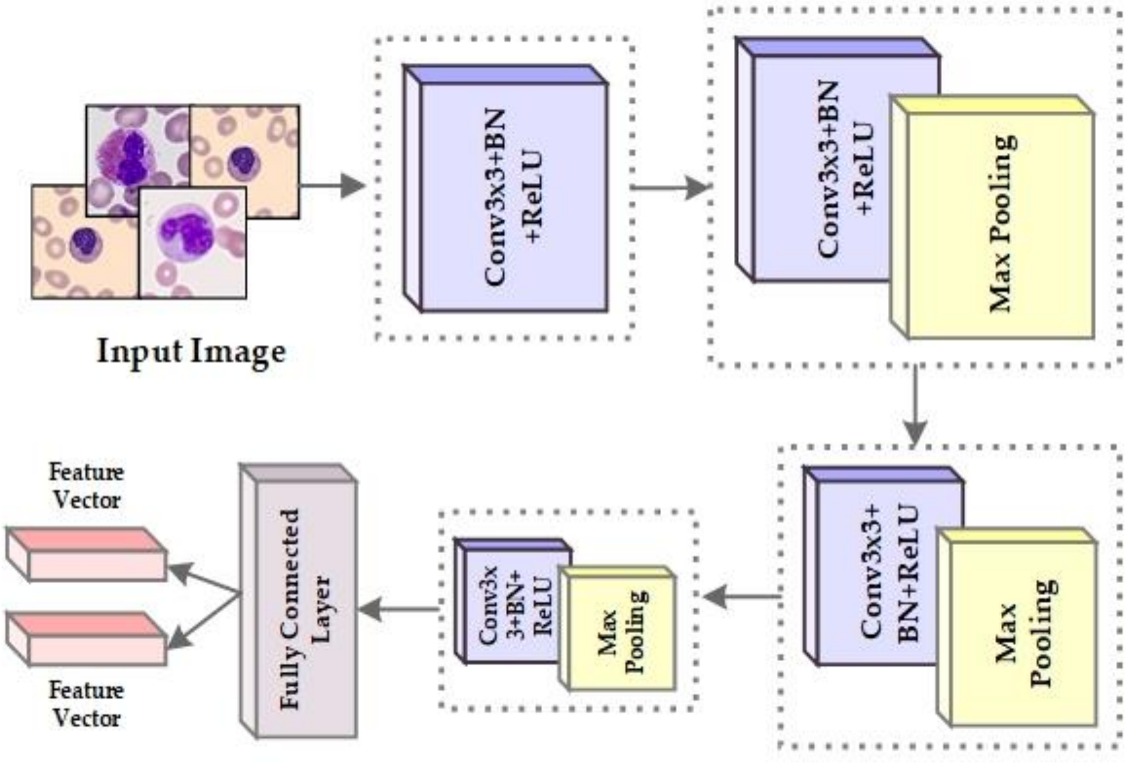

Figure 33: Architecture of the encoder network of the few-shot image generation

The encoder network is shown in Figure 33 which consists of four residual blocks with convolutional layer, ReLU activation and batch normalization. Three out of four boxes are followed by a max pooling layer and at the end there is a fully connected layer and feature vectors are the output of the encoder network. Figure 34 presents the discriminator architecture of the few-shot image generation which uses three 3x3 convolutions with spectral normalization and LeakyReLU followed by 4x4 convolutions with spectral normalization and LeakyReLU. The model is followed by a single 3x3 convolution and a fully connected later. The discriminator outputs the probability of an image belonging to the real distribution.

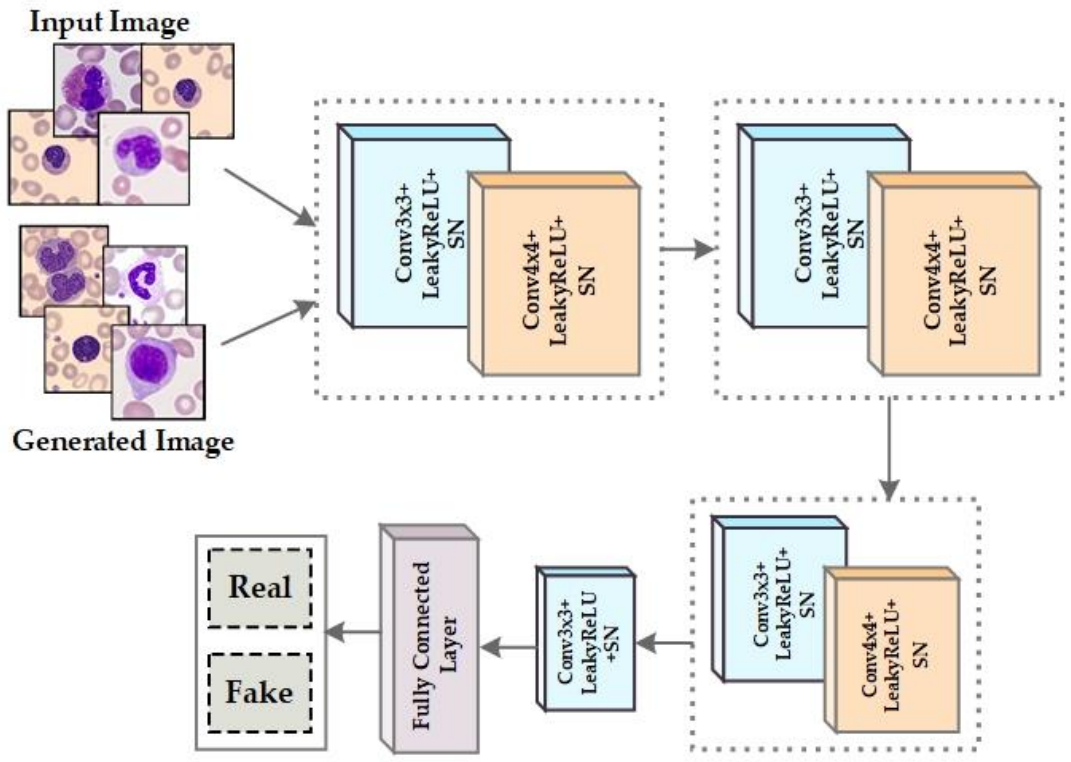

Figure 34: Architecture of the discriminator network of the few-shot image generation

3.7 Novel Deep Learning Model for Classification

After few-shot image generation, we get a balanced dataset by combining the original and the generated synthetic data. For classification of the microscopic single-cell images, we propose a novel deep learning model named as SENet-154-GE. The proposed model is based on three foundations: Squeeze-and-Excitation Networks (SENet), aggregated residual transformation-ResNeXt model and Gather-Excite framework (GE).

ResNeXt is a model that introduces a new dimension called cardinality on top of the ResNet architecture. It was proposed by Facebook AI research and UC San Diego for enhancing the performance of image classification [130]. In a simple neuron as shown in Figure 35, the output is a combination of splitting, transforming and merging and is represented by Equation 46:

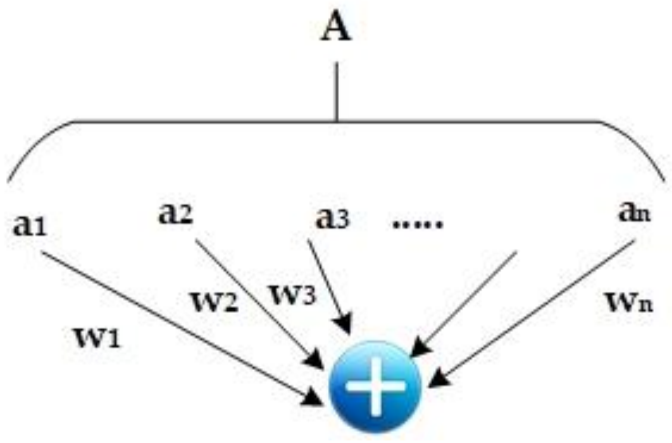

Figure 35: A simple neuron

$$\sum_{j=1}^{n} w_j a_j \qquad (46)$$

The vector a is split into a lower-dimensional embedding. In Figure 35 it is a single-dimension embedding a_j. The vector or input is then transformed i.e. $w_j a_j$ and then it is merged through the summation operator. In ResNeXt instead of the linear function $w_j a_j$, a generic function is introduced that expands along a new dimension and is referred to as the "Network-in-Neuron" instead of "Network-in-Network" which expands the dimension of depth [131]. The aggregated transformation in ResNeXt is presented in Equation 47:

$$\mathcal{F}(\mathrm{a}) = \sum_{\mathrm{i}=1}^{\mathrm{C}} \mathcal{T}_{\mathrm{j}}(\mathrm{a}) \qquad (47)$$

Similar to a simple neuron $\mathcal{T}_{\mathrm{j}}$ is an arbitrary function that project a into a low-dimension embedding and then transforms it. In Equation 47, C refers to the cardinality [132] that denotes the size of the set of transformation that has to be merged or aggregated. In Equation 47, we see C in the position of n in Equation 46, but C can be an arbitrary number and has not to be equal to n. The dimension of width represents the amount of simple transformations but the dimension of cardinality handles more complex transformations and is considered as a more important dimension than depth or width. In Figure 36, we show the relationship of ResNeXt block with Inception-ResNet and grouped convolution in AlexNet. Three of the architecture has same internal

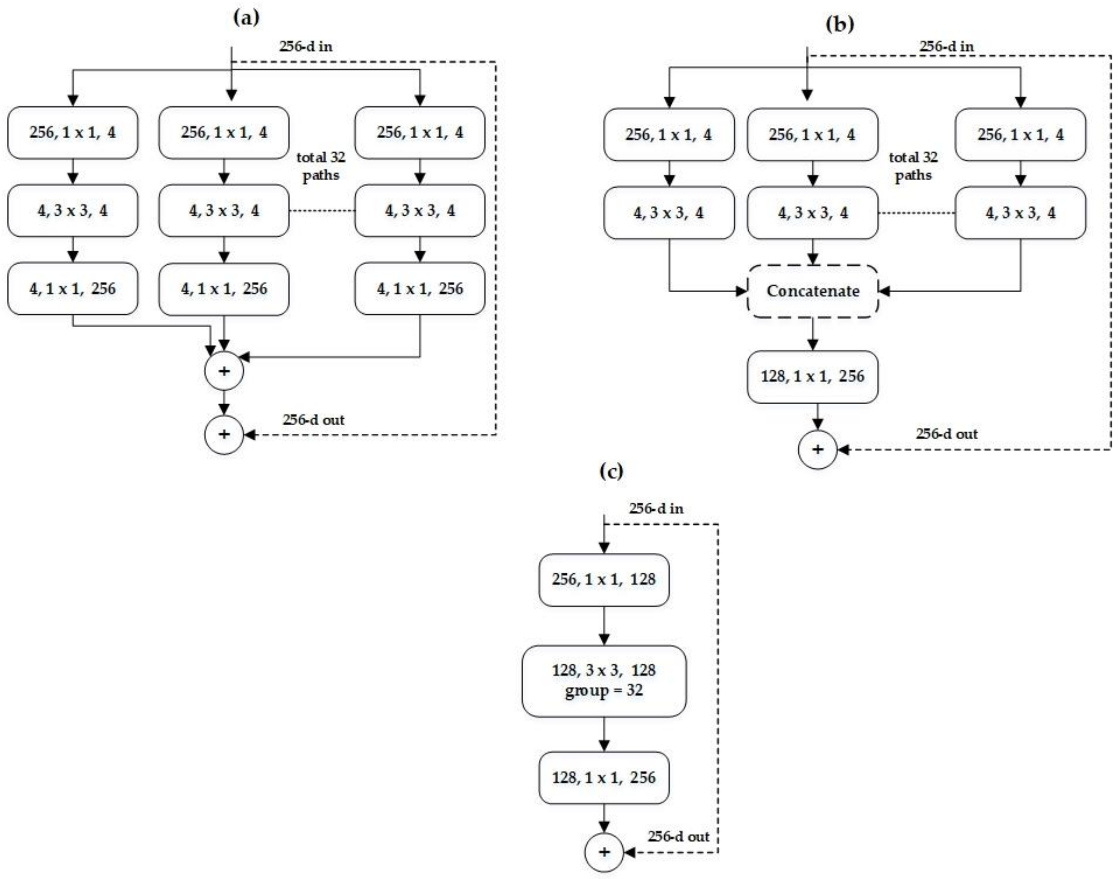

Figure 36: ResNeXt equivalent building blocks (a) Aggregated residual transformation (b) Inception -ResNet (c) Grouped convolution

dimensions. In Figure 36 (a) each convolution path consists of Conv 1x1- Conv 3x3 – Conv 1x1 block, which is the basic design of the ResNet. For each path the internal dimension is 4 which is represented with d and the cardinality C refers to the number of paths i.e. C=32. Aggregating the dimension of each Conv 3x3 i.e. d x C = 4 x 32, we get 128 as the dimension. In aggregated residual transformation Figure 36 (a), the dimension is directly raised from 4 to 256 and added together along with the addition of skip connections. In Inception-ResNet, the dimension needs to be increased from 4 to 128 and then to 256, so ResNeXt requires less effort to design each path and in ResNeXt the neurons of one path is not connected to the neurons of other paths like ResNet. Figure 36 (b) shows Inception-ResNet block that has Conv 1x1 – Conv 3x3 for each convolution path and is also of dimension 128. Here the outputs are concatenated and Conv 1x1 is utilized to regain the dimension from 128 to 256. The only difference is the concatenation. In Figure 36 (c), we see grouped convolution as was suggestes in AlexNet [133]. Every convolution path has Conv 1x1 – Conv 3x3 – Conv 1x1 layers. The low-dimension embeddings i.e. Conv 1x1 can be replaced by a wider and single layer (Conv 1x1, 128-d). There are 32 groups of convolutions with 4-dimensional input and output channels. The authors of ResNeXt concluded that grouped convolution performed better and faster compared to the other two (Figure 36 (b) and 36 (c)).

The squeeze-an-excitation block is an easy-to-plug-in module that consists of the squeeze module, the excitation module and the scaling module [134]. In convolutional neural networks lower layers can find trivial features while upper layers can understand complex structures. The whole network fuses spatial and channel information of images to extract valuable information for solving a task. SENet proposes squeeze module where the feature maps across the spatial dimension are aggregated to generate a channel descriptor. Each channel is squeezed to a single numeric value through global average pooling. Excitation module captures channel-wise dependencies fully and also learns about nonlinear relationship between channels. The excitation operation works as a gating mechanism. Embeddings are input to the excitation operation and it generates modulation weights of each channel all together. The nonlinearity is added through a fully connected layer that is followed by a ReLU function. Each channel receives a gating function through a second fully connected layer that is followed by a sigmoid activation. In the scaling module every feature map of the convolutional block is weighed according to the output of the side network. The transformation output is rescaled with the activations to generate the final output of the SE block. Figure 37, represents the squeeze-and-excitation block plugged into ResNeXt module where H is height of the feature map, W is width of the feature map and C is the number of channels of the feature map.

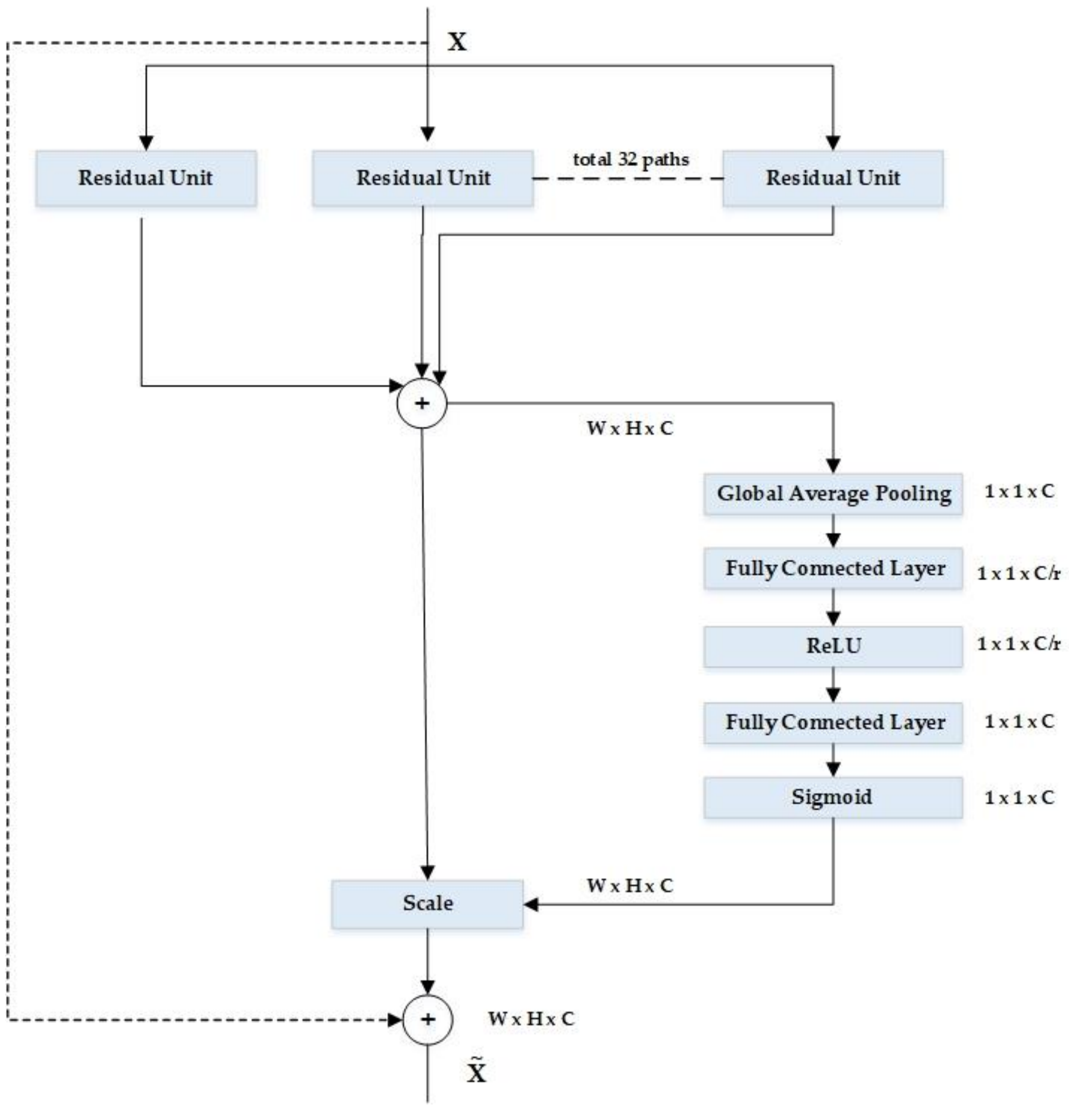

Figure 37: Schema of SE-ResNeXt module

For our classification model, we have incorporated squeeze-and-excitation (SE) block into ResNeXt-152 64x4d network. SENet-154 was introduced in [134]. ResNeXt-152 adopts the stacking of blocks strategy of ResNet-152 [135] in the modified version of ResNeXt-101. The modifications that have been made to the ResNeXt-152 model apart from the incorporating the SE block are [134] 1) The initial 1x1 convolutional channels were halved for every building block. 2) Incorporated three 3x3 convolutional layer in place of first 7x7conv layer. 3) 3x3conv layer with stride 2 was used instead of 1x1 downsampling convolutional layer with stride 2. 4) A dropout layer was added before the classification layer with a dropout ratio of 0.2. 5) Regularization (label smoothing) was [136] used for training. The overview of our proposed SENet-154-GE classification model is shown in Figure 38. The architecture is inspired from [137].

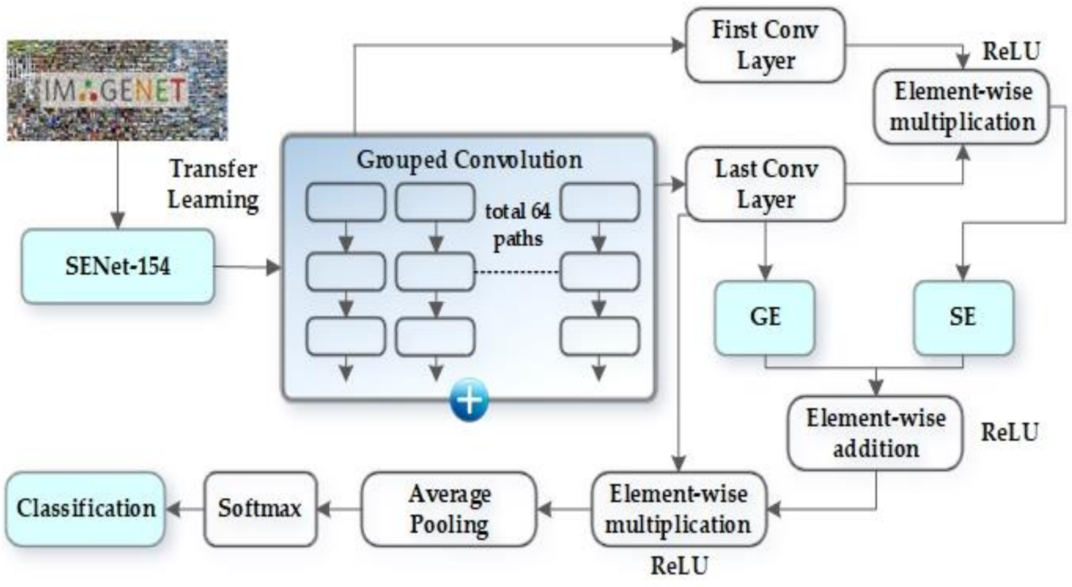

Figure 38: Overview of the SENet-154-GEmodel for classification

The proposed model utilizes grouped convolution of ResNeXt architecture. Features of the first and the last convolutional layer is fused through element-wise multiplication and passed to the squeeze-and-excitation module. Features of the last convolutional layer are passed to the gather-excite (GE) module [138]. The GE module introduces a pair of step-wise deep convolution known as gather operator $\mathcal{E}_G$ and an excite operator $\mathcal{E}_E$. The gather operator merges contextual information of each feature map on a large spatial scale and the excite operator conditions on the aggregates and modulates the feature maps. The excite operator distributes the merged information to the local features. By introducing the GE module, we exploit feature context of the microscopic single-cell images and improve the performance of feature extraction by focusing in detail on the local features. The layout of GE

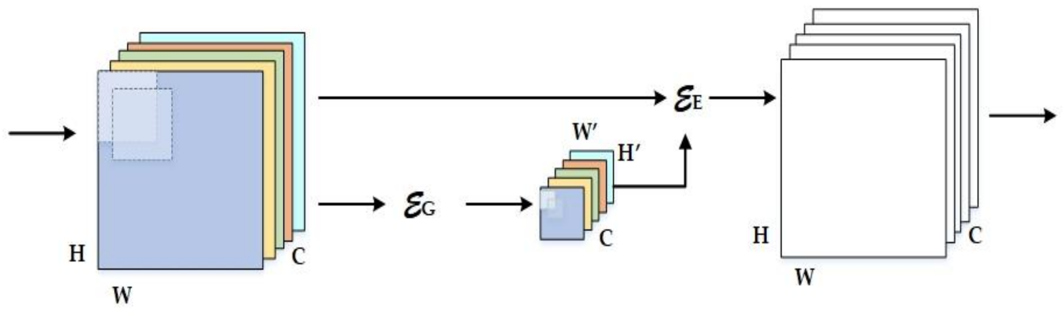

Figure 39: Layout of gather-excite module

model is shown in Figure 39. We get the attentional features by fusing the features of the GE and SE module through element-wise addition. Then, the fused feature of SE and GE module are fused with the feature of last convolutional layer. We then perform average pooling and use softmax for getting the classification results. The SENet-154-GE has been trained on twelve peripheral blood cell classes namely basophil, eosinophil, erythroblast, immature granulocyte, lymphocyte, monoblast, monocyte, myeloblast, neutrophil, platelet, promyelocyte and smudge cells.

Chapter 4: Experiments and Performance Analysis

In this section, we discuss about the experiments we performed for each module presented in the proposed methodology section and we present the results and performance analysis of each proposed module. In section 4.1, we present the result analysis of stain normalization. Section 4.2 presents the experiments and performance analysis for classifier-based generative adversarial networks. In section 4.3, performance of few-shot image generation is discussed. In next section 4.4, we validate our proposed SENet-154-GE classification model and in the end in conclusion we present an ablation study for our proposed methodology to enhance classification of microscopic single-cell images obtained from peripheral blood smears.

4.1 Performance Analysis of Stain Normalization

We assess the quality of stain normalized images generated by cycle consistency GAN in this section. The training time for our proposed stain normalization GAN model was 10hr 36min. Comparison of computation or training time for CycleGAN, pix2pix and our proposed CycleGAN is presented in Table 13. We use three evaluation metrics Fréchet Inception Distance (FID), Structural Similarity Index Measure (SSIM) and Inception Score (IS).

Table 13: Comparison of computation or training time for different models

CycleGAN [49]	pix2pix [50]	Proposed CycleGAN
12hr 6min	13hr 4min	10hr 36min

Fréchet Inception Distance (FID) is a popular evaluation metrics for GANs that computes the distance between features vectors of original data and features of generated data. Fréchet Inception

Distance (FID) score was introduced by Martin Heusel et al. as an improved version of inception score (IS) [139]. Through Fréchet distance, we can measure the distance between the real and the synthetic data distribution as well as similarity between curves that considers the order of points and the location along the curves. Fréchet distance for a univariate normal distribution is computed as shown in Equation 48:

$$\mathrm{FD}(A,B) = (\mu_A - \mu_B)^2 + (\sigma_A - \sigma_B)^2 \qquad (48)$$

Where, A and B are two normal distributions and μ and σ are mean and standard deviation of the distributions. The Fréchet distance for evaluating GAN uses Inception V3 pre-trained model on the Imagenet dataset and therefore the name is Fréchet Inception Distance (FID). FID uses the activations from the inception V3 pre-trained model and the activations are taken from the last pooling layer. There are 2048 activations in the output layer and every image is predicted as 2048 activation feature vector. The feature vector is predicted both for real and synthetic images and the output contains both the collection of 2048 feature vectors for original and synthetic images. The FID for multivariate normal distribution can be defined as in Equation 49:

$$\mathrm{FID} = \|\mu_R - \mu_S\|_2^2 + \mathrm{Tr}\left(C_R + C_S - 2(C_R C_S)^{\frac{1}{2}}\right) \qquad (49)$$

Where, (μ_R, C_R) and (μ_S, C_S) are the mean and covariance of real and synthetic features and Tr is the trace of the matrix.

The structural similarity index measure (SSIM) between two images ranges between zero and one. One indicates the two images are very similar and zero indicates the two images are different. The SSIM score is computed with a combination of three features: luminance, contrast and structure. Luminance is calculated by taking the average over all pixel values. The measure of standard deviation of all pixel values gives the contrast. The input is divided with its standard deviation to obtain a result containing unit standard deviation and by this we get a structural comparison. So, we define luminance (L), contrast (C) and structure (S) as shown in Equation 50, 51 and 52:

$$L(\text{real}, \text{gen}) = \frac{2\,\mu_{\text{real}}\,\mu_{\text{gen}} + C_1}{\mu_{\text{real}}^2 + \mu_{\text{gen}}^2 + C_1} \quad (50)$$

$$C(\text{real}, \text{gen}) = \frac{2\sigma_{\text{real}}\,\sigma_{\text{gen}} + C_2}{\sigma_{\text{real}}^2 + \sigma_{\text{gen}}^2 + C_2} \quad (51)$$

$$S(\text{real}, \text{gen}) = \frac{\sigma_{\text{realgen}} + C_3}{\sigma_{\text{real}}\,\sigma_{\text{gen}} + C_3} \quad (52)$$

Where μ and σ represents the mean and standard deviation and C_1, C_2 and C_3 are constants for introducing numerical stability. Therefore, the SSIM can be defined as in Equation 53:

$$\text{SSIM}(\text{real},\text{gen}) = [L(\text{real},\text{gen})]^{\alpha} \cdot [C(\text{real},\text{gen})]^{\beta} \cdot [S(\text{real},\text{gen})]^{\gamma} \quad (53)$$

Where α, β and γ are the weighting factors and signifies the importance of each metrics.

Inception score (IS) measures the quality of the images generated by the generative models. It uses an Inception v3 network which is pretrained on the ImageNet dataset to evaluate the performance of the network on the generated images. The inception score (IS) can be computed with Equation 54:

$$\text{IS}(G) = \exp\left(\mathbb{E}_{x\sim p_{gen}} KL_D\big(p(y \mid x) \parallel p(y)\big)\right) \quad (54)$$

Where, $x \sim p_{\mathbf{gen}}$ denotes that image x is sampled from the distribution p_{gen}. $KL_D(p||q)$ indicates the KL divergence between p distribution and q distribution. p(y|x) represents conditional class distribution, p(y) is the marginal class distribution and exp is used for easier comparison of values.

Table 14: FID scores between real, generated and reconstructed images for domain A and B

FID	CycleGAN [49]	pix2pix [50]	Proposed CycleGAN
A^{real} vs. A^{gen}	9.9732	14.5771	4.9834
A^{real} vs. A^{rec}	5.6637	9.8405	3.7155
B^{real} vs. B^{gen}	11.7466	13.5682	4.2130
B^{real} vs. B^{rec}	6.4624	8.3391	3.7126

In Table 14., we compare the FID scores of CycleGAN for stain normalization as proposed by [140], pix2pix and our proposed modified CycleGAN for stain normalization. FID scores between real and generated as well as reconstructed images are presented in Table 14. A represents domain A (Mendeley dataset stain) and B represents domain B (Cancer imaging archive data stain). A^{real} denotes real image in domain A and A^{gen} represents the generated image through the adversarial networks. Similarly, B^{real} is real image in domain B and B^{gen} denotes generated images. A^{rec} and B^{rec} are the reconstructed image in domain A and domain B. The lower the FID score the better the quality of the generated images. As can be seen in Table 14 our proposed modified CycleGAN produces lower FID score as compare to general CycleGAN model and pix2pix model with an average score of 4.9834 for real vs. generated images in domain A, 3.7155 for real vs. reconstructed in domain A, 4.2130 for real vs. generated images in domain B and 3.7126 for real and reconstructed images in domain B.

Table 15: SSIM scores between real and reconstructed images

SSIM	CycleGAN [49]	pix2pix [50]	Proposed CycleGAN
A^{real} vs. A^{rec}	0.9535	0.9122	0.9834
B^{real} vs. B^{rec}	0.9662	0.9256	0.9869

Table 16: Inception score comparison

	CycleGAN [49]	pix2pix [50]	Proposed CycleGAN
IS	88.34	85.67	96.26

In Table 15, we present the SSIM score between real and reconstructed images in domain A and domain B. We compare the result of CycleGAN and pix2pix with proposed modified CycleGAN for stain normalization. The SSIM score as mentioned earlier ranges from 0 to 1 and the score nearer to 1 represents images with higher similarity. As can be seen in Table 15, proposed CycleGAN achieved highest SSIM score compared to CycleGAN and pix2pix.

In Table 16, we provide the inception score of the three models and proposed CycleGAN achieved highest inception score indicating our model can generate various distinct images. In Figure 40, we present the training loss of modified CycleGAN for stain normalization on our dataset. It has been trained for 150 epoch and as we can see from Figure 40, the loss became stable at around 130 epochs which means the proposed modified CycleGAN model reached Nash Equilibrium and therefore the training was stopped at 150 epochs. The loss value was between 0.2-0.5 for every network.

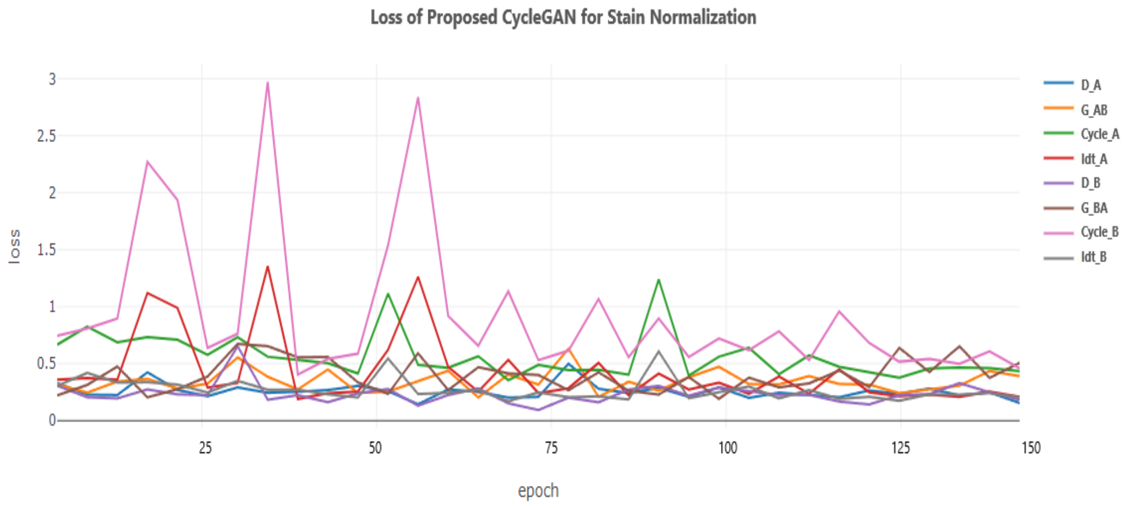

Figure 40: Loss of proposed CycleGAN for stain normalization

We have performed stain normalization for six cell types: basophil, eosinophil, erythroblast, lymphocyte, immature granulocyte and monocyte. Except for lymphocyte, the stain from the Cancer Imaging Archive dataset has been normalized to the stain of the Mendeley dataset (B → A) as images were less for each cell types in the Cancer Imaging Archive dataset. For lymphocyte images in the Mendeley dataset was more so we performed stain normalization from (A → B). For the other cell types, we have not performed stain normalization since images contained the same stain color. From Figure 41 to Figure 46, we present the few samples of real, stain normalized and the reconstructed images of six cell types using our proposed modified CycleGAN. Reconstructed images are fake or synthetic images which should look and possess characteristics exactly like the original image.

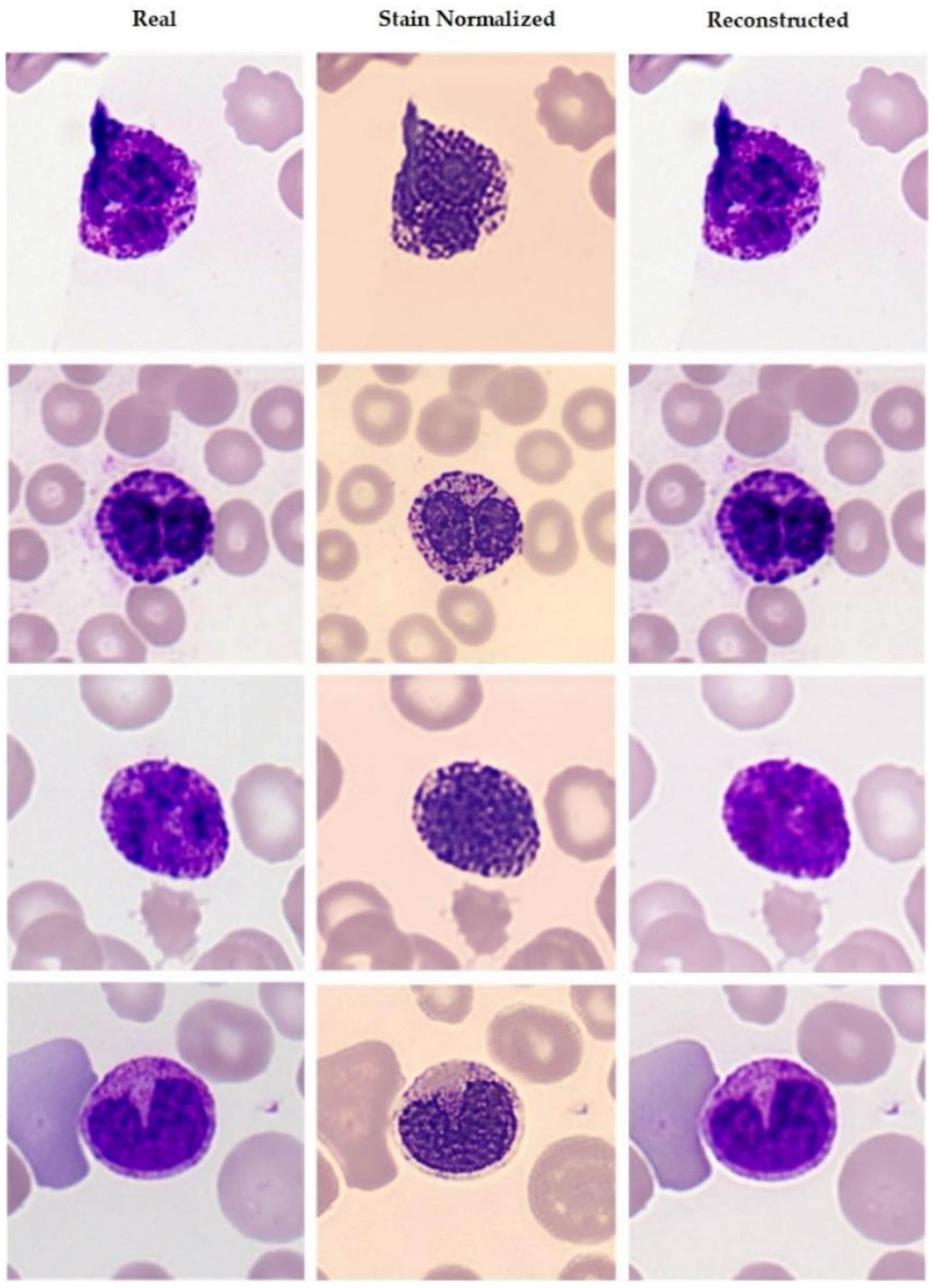

Figure 41: Real, stain normalized and reconstructed images for Basophil

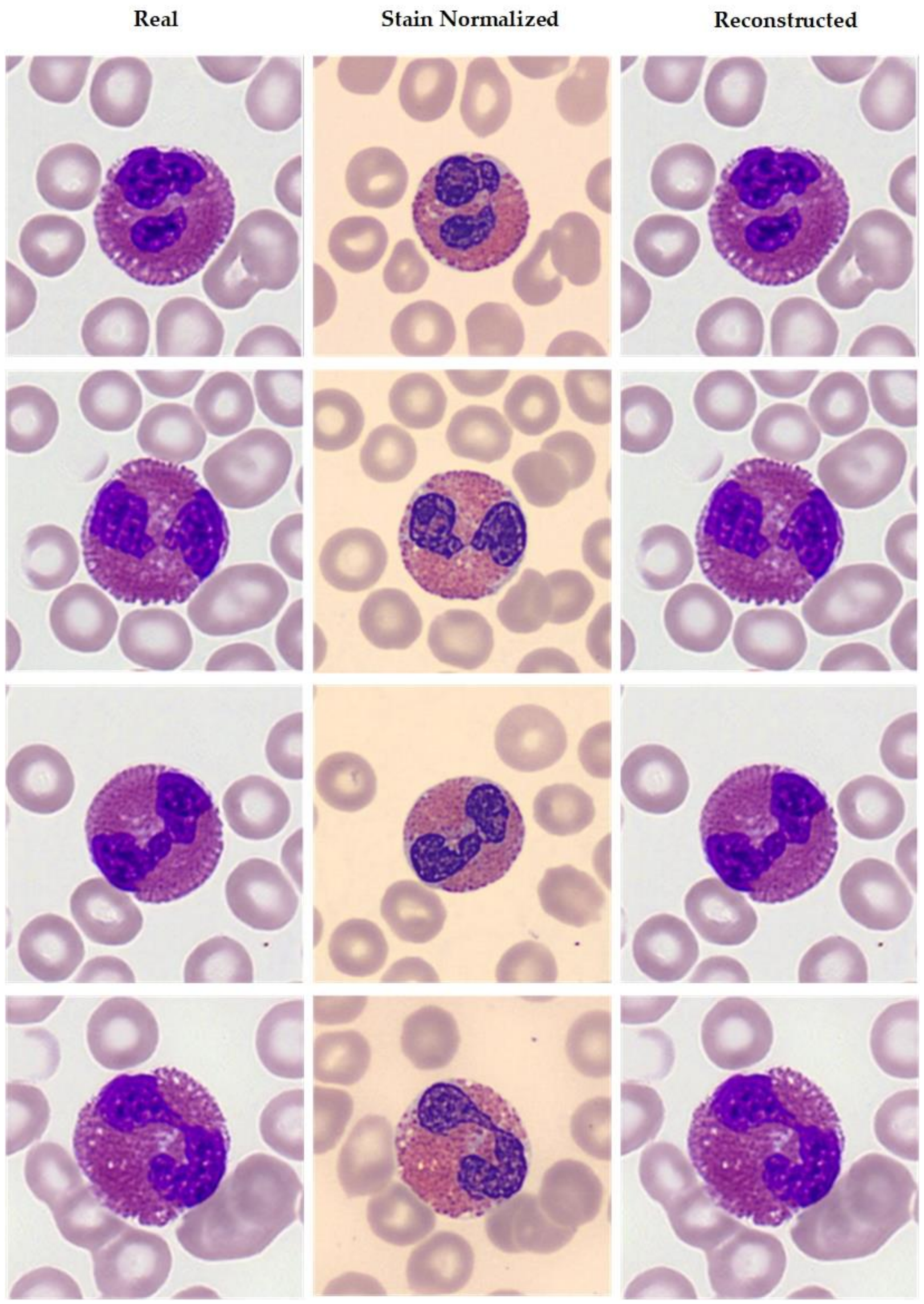

Figure 42: Real, stain normalized and reconstructed images for Eosinophil

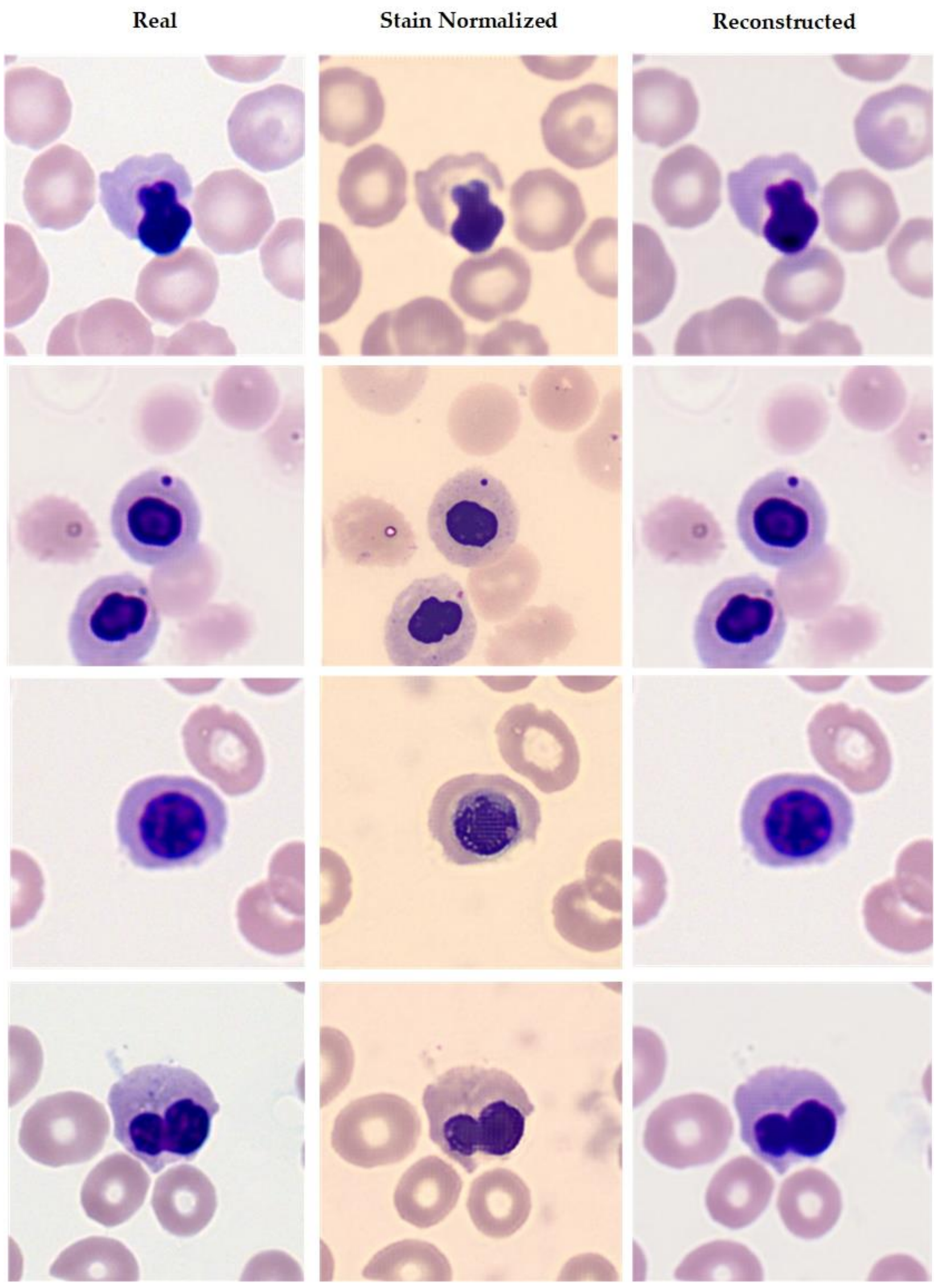

Figure 43: Real, stain normalized and reconstructed images for Erythroblast

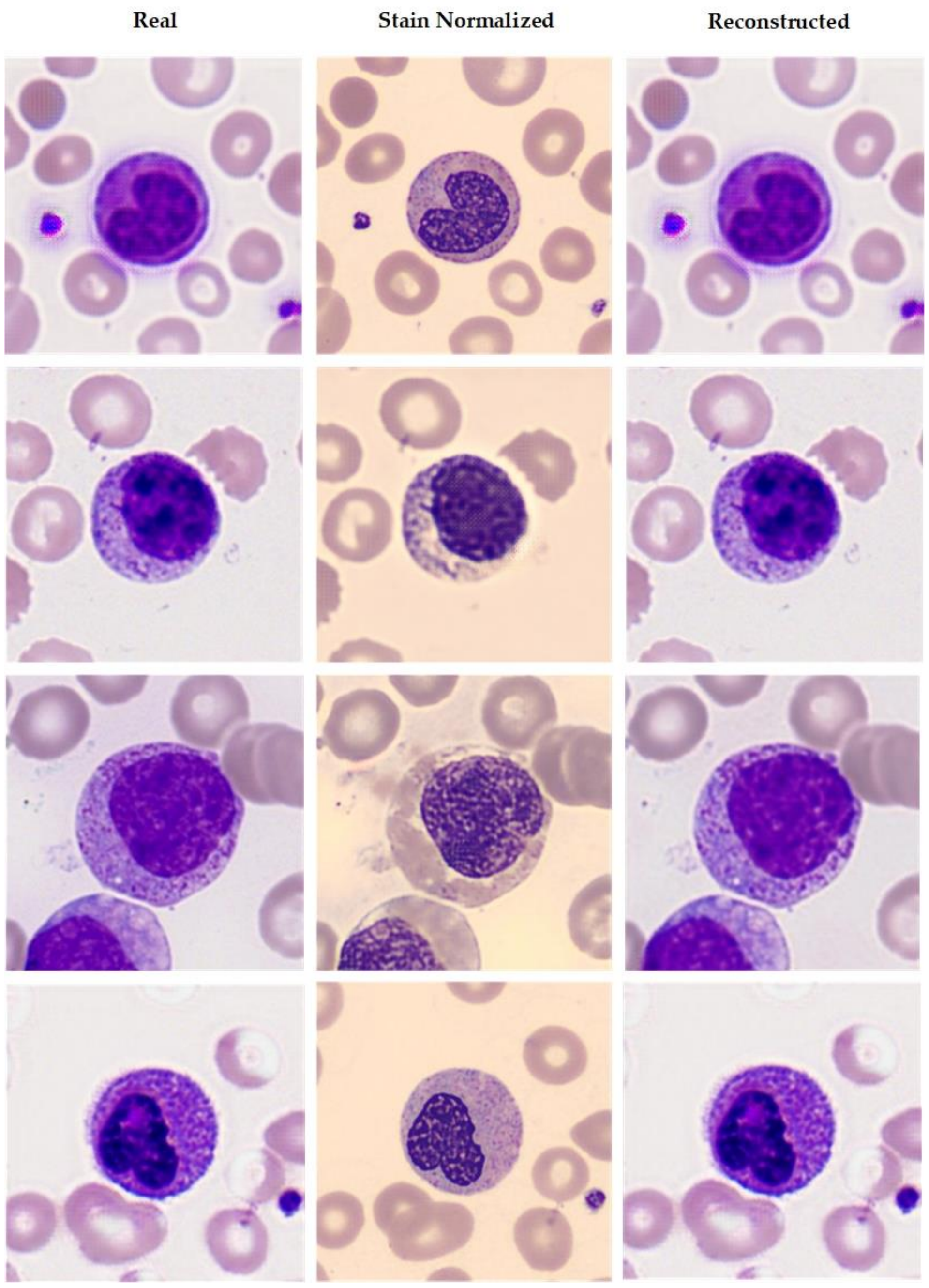

Figure 44: Real, stain normalized and reconstructed images for Immature Granulocytes

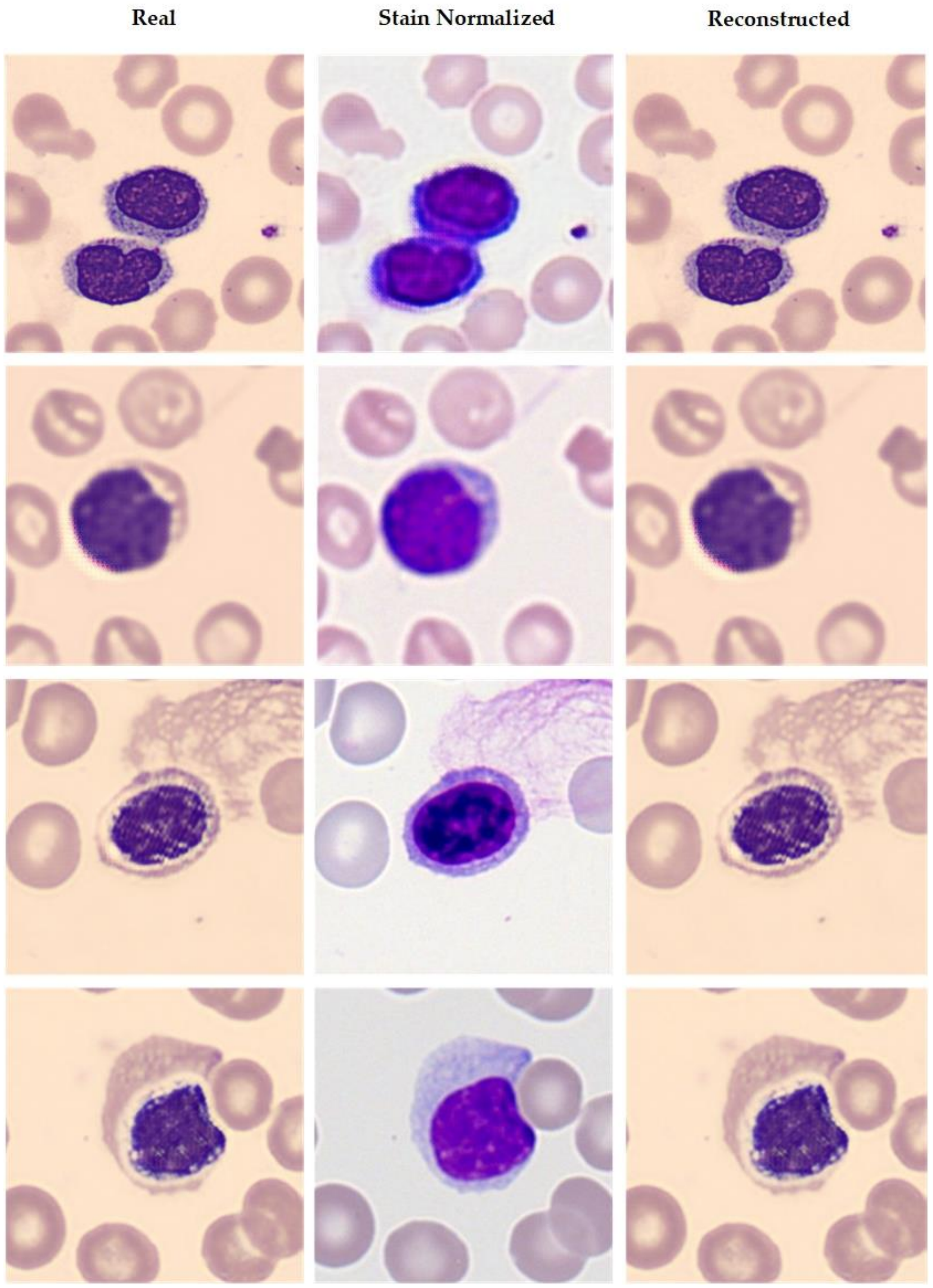

Figure 45: Real, stain normalized and reconstructed images for Lymphocyte

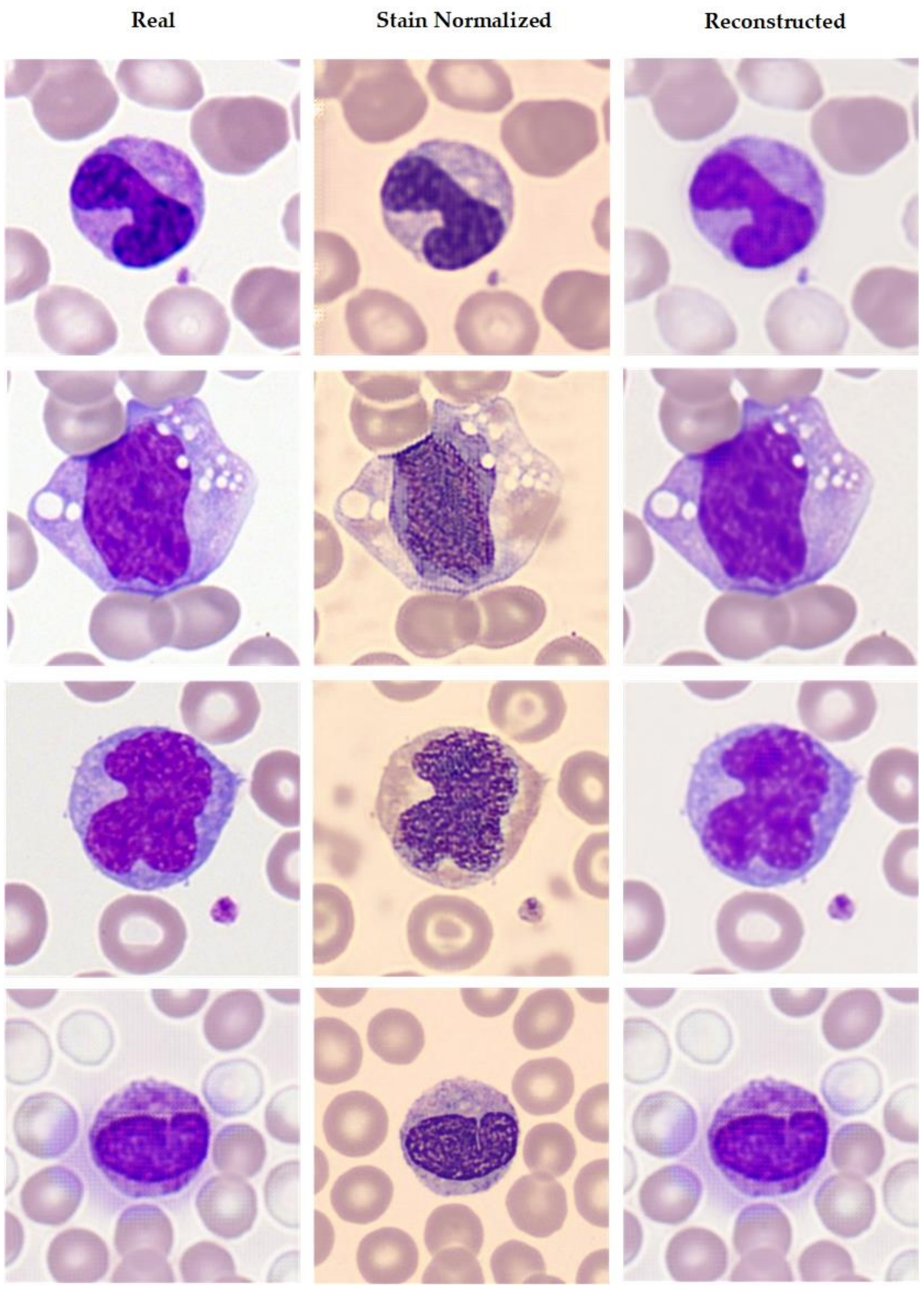

Figure 46: Real, stain normalized and reconstructed images for Monocyte

4.2 Performance Analysis of Classifier-Based Generative Adversarial Networks

For evaluating our proposed classifier-based generative adversarial networks (C-WGAN-GP) we have used various evaluation metrics. The training time for C-WGAN-GP was 9hr 13min. We compared the performance of C-WGAN-GP model with auxiliary classifier GAN (AC-GAN), conditional Wasserstein GAN with gradient penalty (CWGAN-GP), information maximizing GAN (Info-GAN) [141], deep convolutional GAN (DCGAN) [142] and conditional GAN (CGAN). Comparison of computation or training time for above mentioned model with our proposed model is presented in Table 17. The evaluation metrics which we have used for quantitative analysis of our proposed model are Fréchet inception distance (FID), precision, F1 score, learned perceptual image patch similarity (LPIPS), recall, inception score (IS), peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), l1 and l2 error.

Table 17: Comparison of training time of GAN models

AC-GAN	CWGAN-GP	InfoGAN	DCGAN	CGAN	C-WGAN-GP
11hr 12min	13hr 10min	12hr 33min	16hr 3min	14hr 2min	9hr 13min

We have used recall for analyzing the quantity of synthetic images created by the generative models. Precision has been used to evaluate the quality of the generated microscopic cell images and also to evaluate the performance of C-WGAN-GP on generating images that can be classified correctly by the classifier. The F1 score measures the harmonic mean between the recall and the precision. Higher value of F1, precision and recall indicate better performance of the model. To compute the distance between image patches, we have used LPIPS metrics. Smaller distance indicates more similarity and better output. PSNR measures the peak signal-to-noise ratio between the reconstructed or generated image and the original image. Higher the value of PSNR, better is the quality of the generated image. We compute the error rate through l1 and l2 loss function where l1 computes the least absolute deviation and l2 computes the squared differences between the real and generated data. We have used C-WGAN-GP to generate images of eight cell types i.e. basophil, eosinophil, erythroblast, immature granulocyte, lymphocyte, monocyte, myeloblast and platelet. In Table 18, Table 19 and Table 20, we present the quantitative analysis result for different GAN models and compare the performance with our proposed C-WGAN-GP model. As can be seen from Table 18, precision, recall and F1-Score are higher for C-WGAN-GP on our dataset indicating that our proposed model performs better compared to existing GAN models. Also, in Table 19, proposed C-WGAN-GP achieves high inception score and lower FID and LPIPS reflecting better quality generation of images. We have presented in Table 20, that our model generates lower error rates, higher structural similarity and better-quality images.

Table 18: Quantitative analysis of different GAN models

Models	Precision	Recall	F1-Score
AC-GAN	93.98	92.16	93.55
CWGAN-GP	87.23	87.46	88.74
InfoGAN	91.66	87.11	87.46
DCGAN	88.62	88.90	88.33
CGAN	94.78	92.33	93.60
C-WGAN-GP	97.93	97.60	97.15

Table 19: FID, IS and LPIPS score comparison of different GAN models

Models	FID	IS	LPIPS
AC-GAN	62.66	13.32 ± 0.59	0.28
CWGAN-GP	72.48	9.83 ± 0.25	0.36
InfoGAN	70.32	9.91 ± 0.10	0.33
DCGAN	77.91	8.44 ± 0.32	0.43
CGAN	71.33	11.59 ± 0.62	0.32
C-WGAN-GP	57.82	17.73 ± 0.11	0.21

Table 20: PSNR, l1, l2 and SSIM comparison of various GAN models

Models	PSNR	SSIM	L1	L2
AC-GAN	35.62	0.9134	12.94%	8.92%
CWGAN-GP	33.46	0.8762	13.26%	9.11%
InfoGAN	29.04	0.9316	10.91%	7.47%
DCGAN	30.83	0.9232	11.63%	8.32%
CGAN	29.99	0.9461	13.22%	8.11%
C-WGAN-GP	40.18	0.9862	6.91%	3.26%

In Figure 47, we have presented sample images of eight cell types generated by proposed C-WGAN-GP. The generated results show that C-WGAN-GP could understand the features of each cell type perfectly and the generated images are worthy of using it for classification of cell types for medical diagnosis and research purpose.

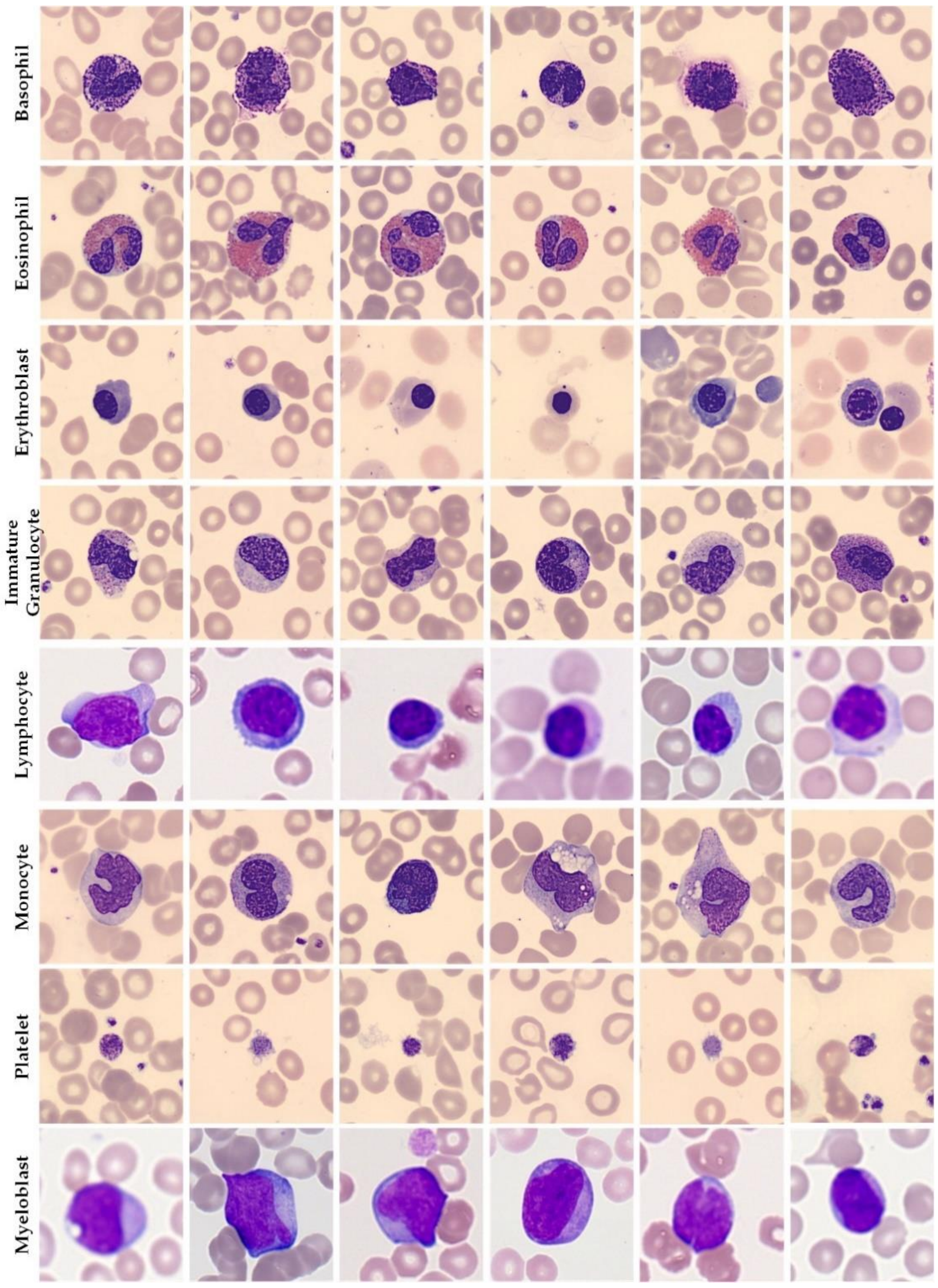

Figure 47: Sample of images generated by C-WGAN-GP

4.3 Evaluation of Few-Shot Image Generation

For evaluating our proposed few-shot image generation model we have used FID, IS and LPIPS evaluation metrics. The few-shot image generation model has been used to generate images for three cell types: Smudge cells, promyelocyte and monoblast. We have trained the model for twenty thousand episodes and the training time was five hours eleven minutes. In below Table 21 and Table 22 we compare the performance of our proposed few-shot image generation model with existing few-shot image generation models. We compare proposed model with Few-shot Image Generation with Reptile (FIGR) [143], FastGAN [144], DiffAugment [145], MineGAN [146] and our proposed classifier-based GAN model.

The diversity in the generation of images were verified by the inception score. The more the inception score, the more capable the model is in generating various images that are distinct from each other. As we can see from Table 21, our proposed few-shot image generation model achieves higher inception score than existing few-shot image generation model as well as our proposed C-WGAN-GP model. Table 21 proves the need for building the few-shot image generation model instead of using only the C-WGAN-GP for producing images for every cell type in our dataset. The image quality is evaluated through the FID score. The lower the FID score, the better is the quality of the generated images. Our proposed few-shot image generation model produced the lowest FID score indicating better performance than the existing models. We measure LPIPS to evaluate the similarity between the original and the synthetic data or the generated images for each cell types. So, lesser distance indicates greater similarity between the original and the generated images. Our proposed model produces images that are most similar to the original data compared to other models. In Table 22, we compare the training time of each model on our dataset, the total number of episodes to achieve the best result and the number of parameters consumed by each model. Comparison shows that our proposed model requires less parameters, less epoch and less training time than existing models.

Table 21: FID, IS and LPIPS comparison of various few-shot image generation models

Models	FID	IS	LPIPS
FIGR	98.83	15.77 ± 0.23	0.6721
FastGAN	67.33	09.91 ± 0.10	0.4517
DiffAugment	102.27	19.52 ± 0.58	0.7783
MineGAN	88.90	12.32 ± 0.44	0.6013
C-WGAN-GP	70.36	10.03 ± 0.27	0.5934
Proposed	51.62	03.16 ± 0.28	0.2956

Table 22: Time, episodes and number of parameters used by each model

Models	Time	Episodes	No. of Parameters
FIGR	19hr	80000	11.5 Million
FastGAN	7hr 30min	45000	7.6 Million
DiffAugment	21hr 12min	850000	13.4 Million
MineGAN	17hr 13min	42000	10.2 Million
C-WGAN-GP	8hr 23min	40000	9.7 Million
Proposed	5hr 11min	20000	5.8 Million

In Figure 48, Figure 49 and Figure 50, we present the sample images from episode 0, 2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000 for each cell types in the generation phase of the few-shot image generation model i.e. promyelocyte, smudge cells and monoblast. As we can see from Figure 48, for promyelocyte we get almost realistic images after 15000 episodes, for smudge cells it takes around 19000 episodes to generate realistic images and for monoblast it takes around 17500 episodes to generate realistic images that can be used for enhancing classification accuracy.

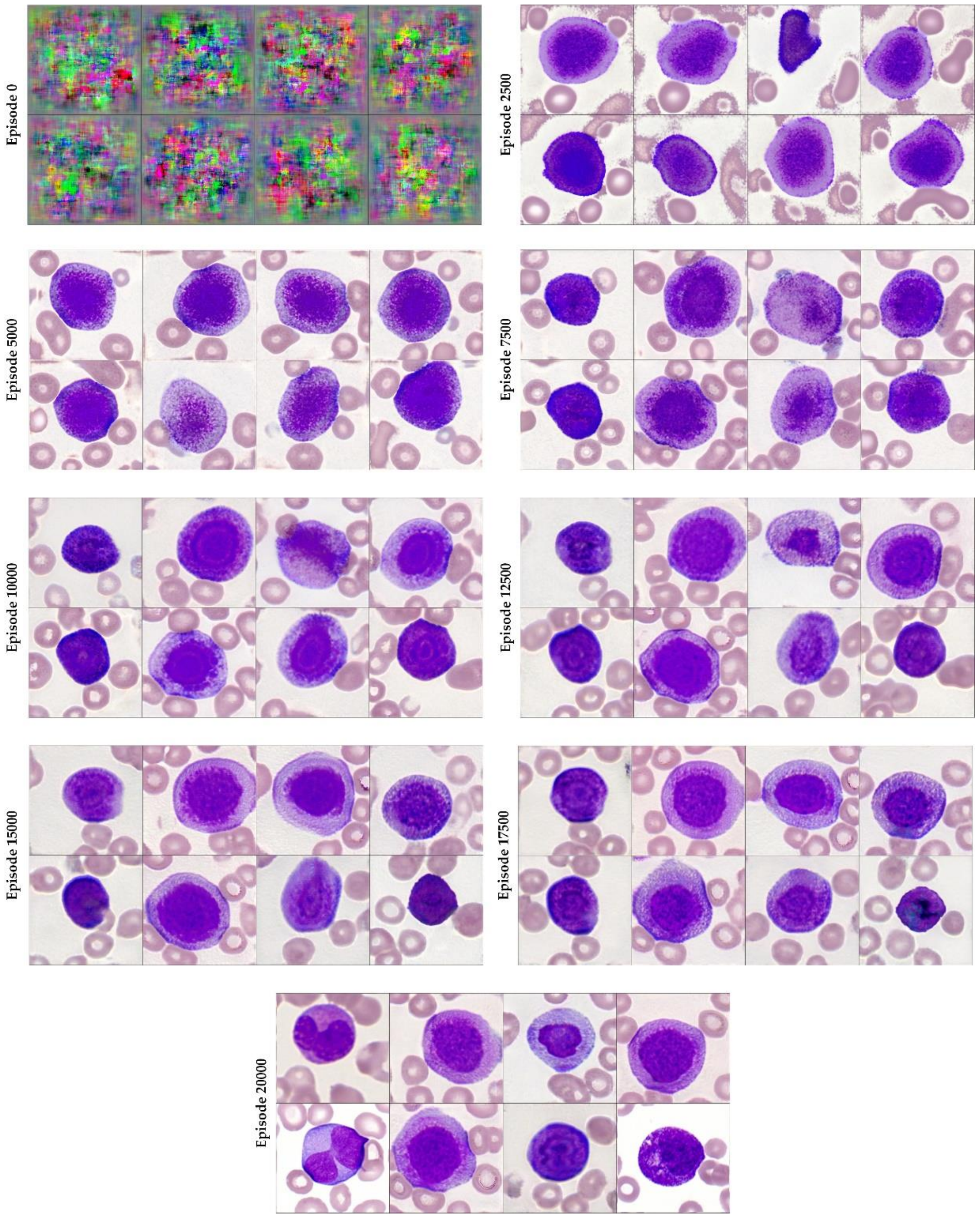

Figure 48: Promyelocyte cell type - Sample of images for 0, 2500, 5000, 7500, 10000, 12500, 15000, 17500 and 20000 episodes

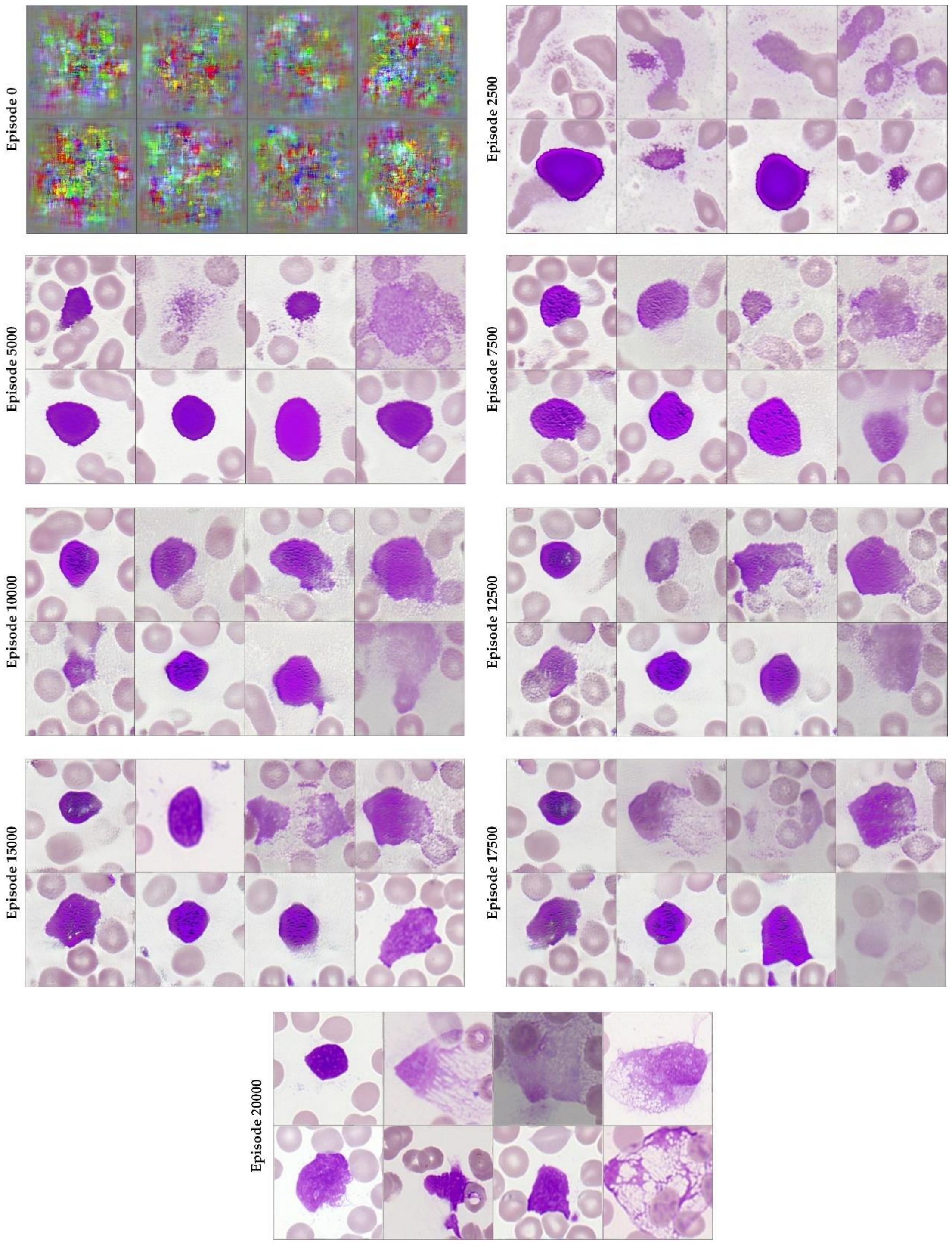

Figure 49: Smudge cell type - Sample of images for 0, 2500, 5000, 7500, 10000, 12500, 15000, 17500 and 20000 episodes

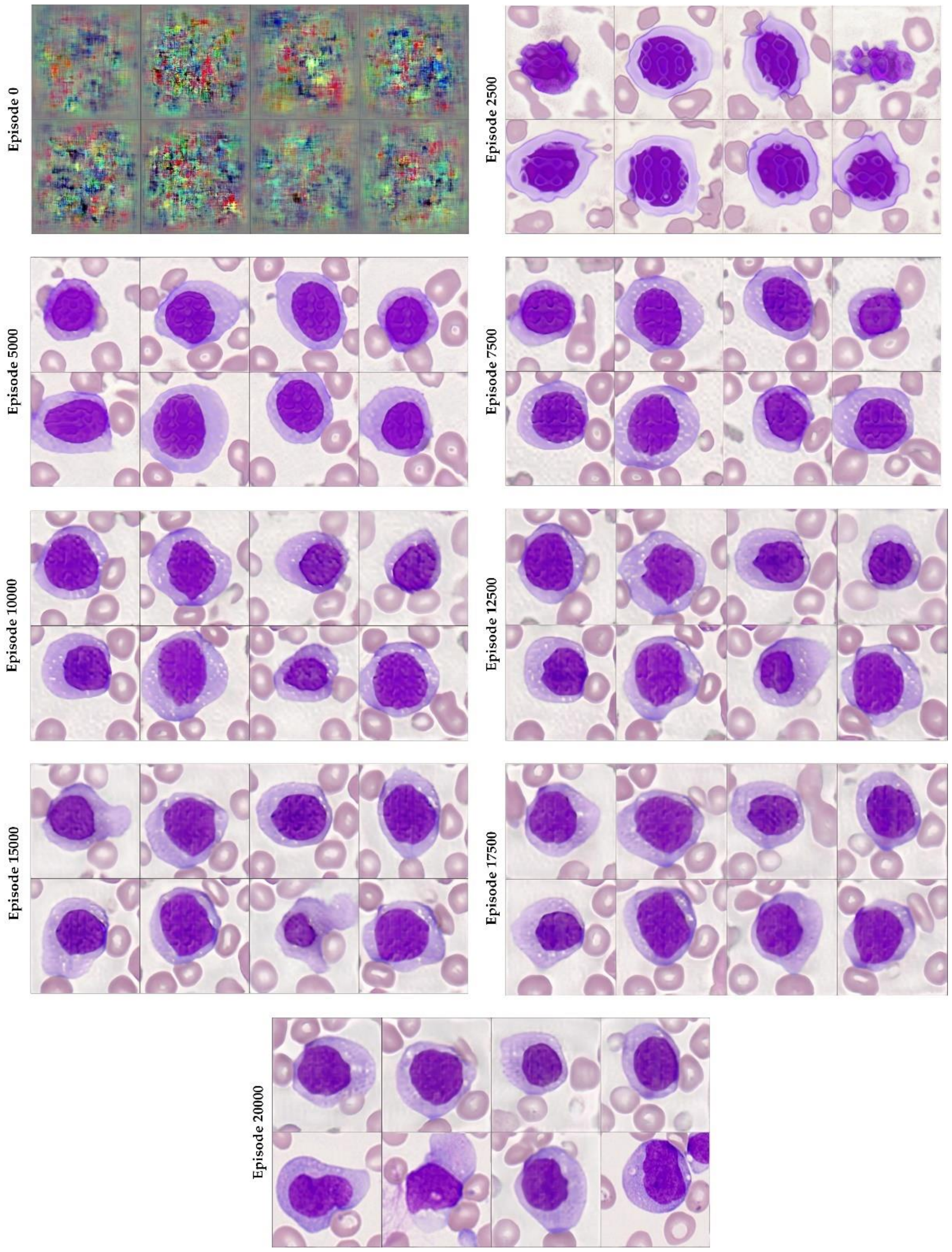

Figure 50: Monoblast cell type - Sample of images for 0, 2500, 5000, 7500, 10000, 12500, 15000, 17500 and 20000 episodes

4.4 Evaluation of SENet-154-GE Classification Model

In this section, we present the evaluation results for our proposed classification model SENet-154-GE. For all our experiments we have utilized the system configuration mentioned in Table 23. First, we present the training and validation accuracy and loss in Figure 51. Our model achieved a training accuracy of 99.8% and a validation accuracy of 98.7%. Training loss for SENet-154-GE was 0.04 and validation loss was 0.12. We trained the model for two hundred epochs which took around one hour twenty minutes time. There were two hundred fifty-six iterations for each epoch. The data was split into 70% training and 30% testing. For validation, we used 10% of the training data.

Table 23: Details of implementation environment

System Component	Description
Operating System	Ubuntu 20.04.2 LTS
Operating System Type	64-bit
Processor	Intel® Core™ i7-8700 CPU @ 3.20GHz × 12
Graphics	NVIDIA Corporation TU104 [GeForce RTX 2080 SUPER]
RAM	32 GB
Programming Language	Python 3.6

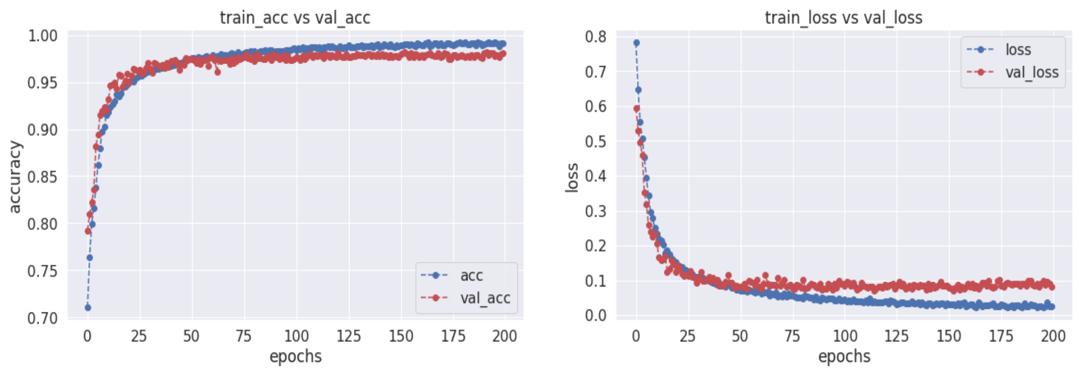

Figure 51: Training and validation accuracy and loss graph

We compare the performance of SENet-154-GE with existing classification models like VGG-19 [147], ResNet-50 [148], InceptionV3 [149], Xception [150], EfficientNet [151], ResNeXt-101 [152], Naïve Bayes [153], SE-ResNeXt-50 [154], and Sequential CNN. We have presented the accuracy comparison of different classification model on our stain normalized and balanced dataset in Figure 52. VGG-19 achieved an accuracy of 83.6%, accuracy of ResNet-50 is 85.38%. InceptionV3 has an accuracy of 90.19%, Xception model has a lower accuracy of 84.20%, EfficientNet achieved 92.44% accuracy, ResNeXt-101 reached 93.32% accuracy. Naïve Bayes achieved an accuracy of 81.85%, SE-ResNeXt-50 performed well with an accuracy of 95.90%. Sequential CNN also performed good with an accuracy of 96.83% and our proposed model achieved an accuracy of 98.4%. The accuracies mentioned above and as shown in Figure 52 are the accuracies obtained on the test dataset. For all the model the data was split into 70% training, 30% testing and 10% of training data was considered as validation dataset. Also, comparison of training time for different classification model is presented in Table 24.

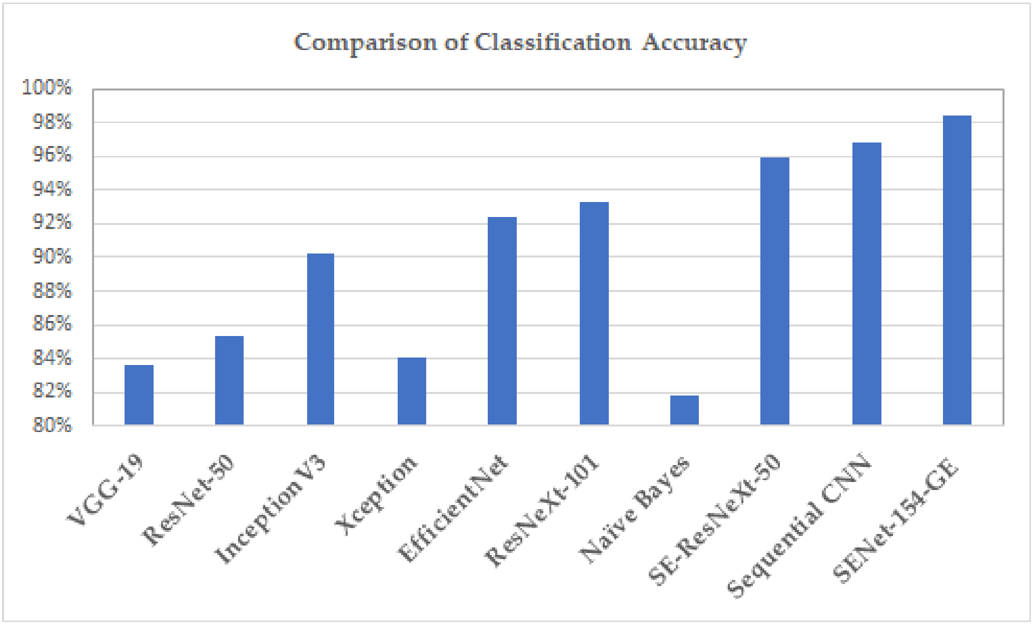

Figure 52: Accuracy comparison of various classification model on balanced dataset

The performance of our proposed classifier SENet-154-GE for each cell type is shown in

Table 24: Training time comparison of different classification models

VGG-19	EfficientNet	Inception V3	SE-ResNeXt-50	ResNeXt-101	SENet-154-GE
16hr 22min	12hr 47min	13hr 11min	9hr 6min	10hr 55min	8hr 3min

Figure 53 and Figure 54. We present the accuracy, specificity and sensitivity of each cell type for the original data and the balanced dataset. For the original dataset, myeloblast achieves the highest accuracy of 94% and the lowest accuracy of 91% is for smudge cells and lymphocyte. In the original dataset we achieve an overall accuracy of 92.41% on the test dataset. In the original dataset as shown in Figure 53, myeloblast has the highest value for sensitivity i.e. 94.5 and basophil, immature granulocyte, monoblast and promyelocyte has the lowest value i.e. 92.5. Similarly, specificity is highest for myeloblast (93.5) and lowest for smudge cells and lymphocyte (91.2). SENet-154-GE achieves an overall value of 92.16 for specificity and 92.66 for sensitivity.

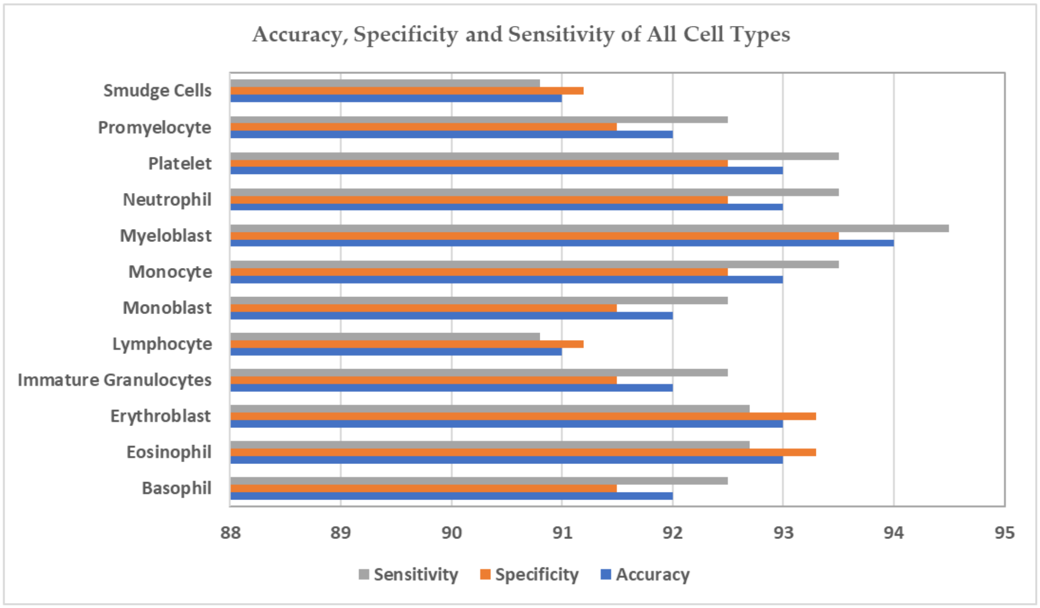

Figure 53: Performance of SENet-154-GE on individual cell types for original dataset

In Figure 54, we present the accuracy, specificity and sensitivity of individual cell type for the stain normalized and balanced dataset we prepared. SENet-154-GE produces the highest accuracy of 99.4% for immature granulocytes and lowest accuracy of 97% for monocyte and smudge cells. The highest specificity value of 99.6 is for neutrophil and the lowest is for monocyte with a score of 97.3. Basophil, eosinophil and immature granulocyte achieved the highest sensitivity value of 99.5 and the lowest sensitivity score of 96.7 was for monocyte. The overall accuracy for the balanced dataset is 98.4% with a specificity score of 98.21 and sensitivity score of 98.58. Our proposed approach of stain normalization, classifier-based GAN model and few-shot image generation increased the accuracy rate by 6%.

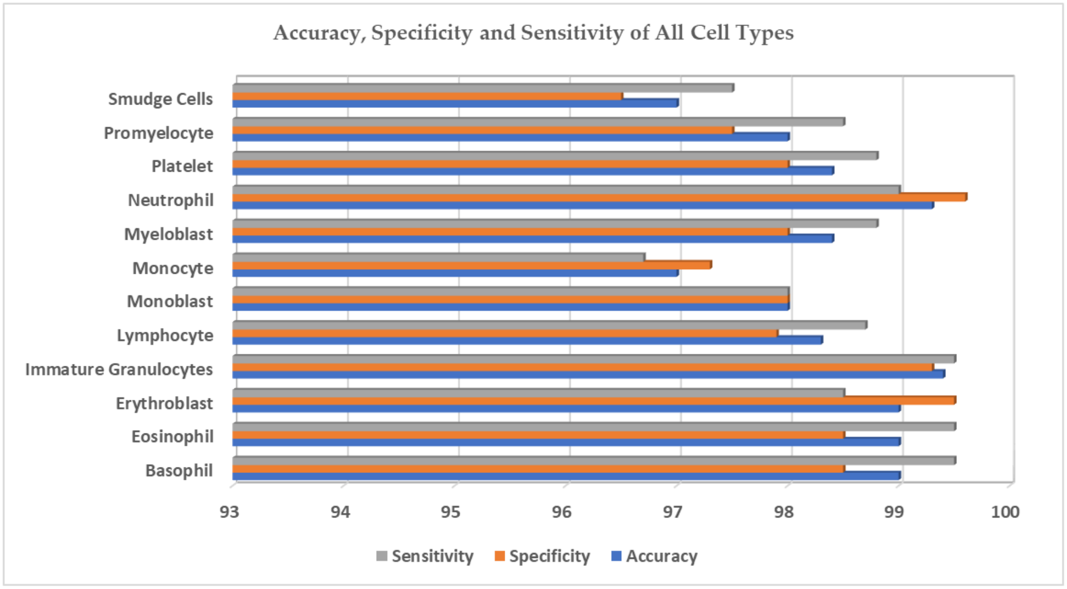

Figure 54: Performance of SENet-154-GE on individual cell types for balanced dataset

In Figure 55 and Figure 56, we present the screen grabs of classification results for the twelve cell types. Our implementation of SENet-154-GE model provides how much percentage the model is certain about the classified cell type for a test image. In Figure 55 and Figure 56, the cell type mentioned in the bracket is the original cell type in our dataset and the cell type with the percentage written is the classified cell type with the confidence score by SENet-154-GE. For example, Lymphocyte 100% (Lymphocyte) would represent that the original cell type is lymphocyte and SENet-154-GE could classify lymphocyte correctly with 100% confidence. In Figure 55 and

Figure 56, 0-11 represents the labels of the cell types. 0 for basophil, 1 is eosinophil, 2 is erythroblast, 3 is immature granulocytes, 4 is for lymphocyte, 5 is for monoblast, 6 for monocyte, 7 represents myeloblast, 8 is neutrophil, 9 is platelet, 10 is promyelocyte and 11 is smudge cells.

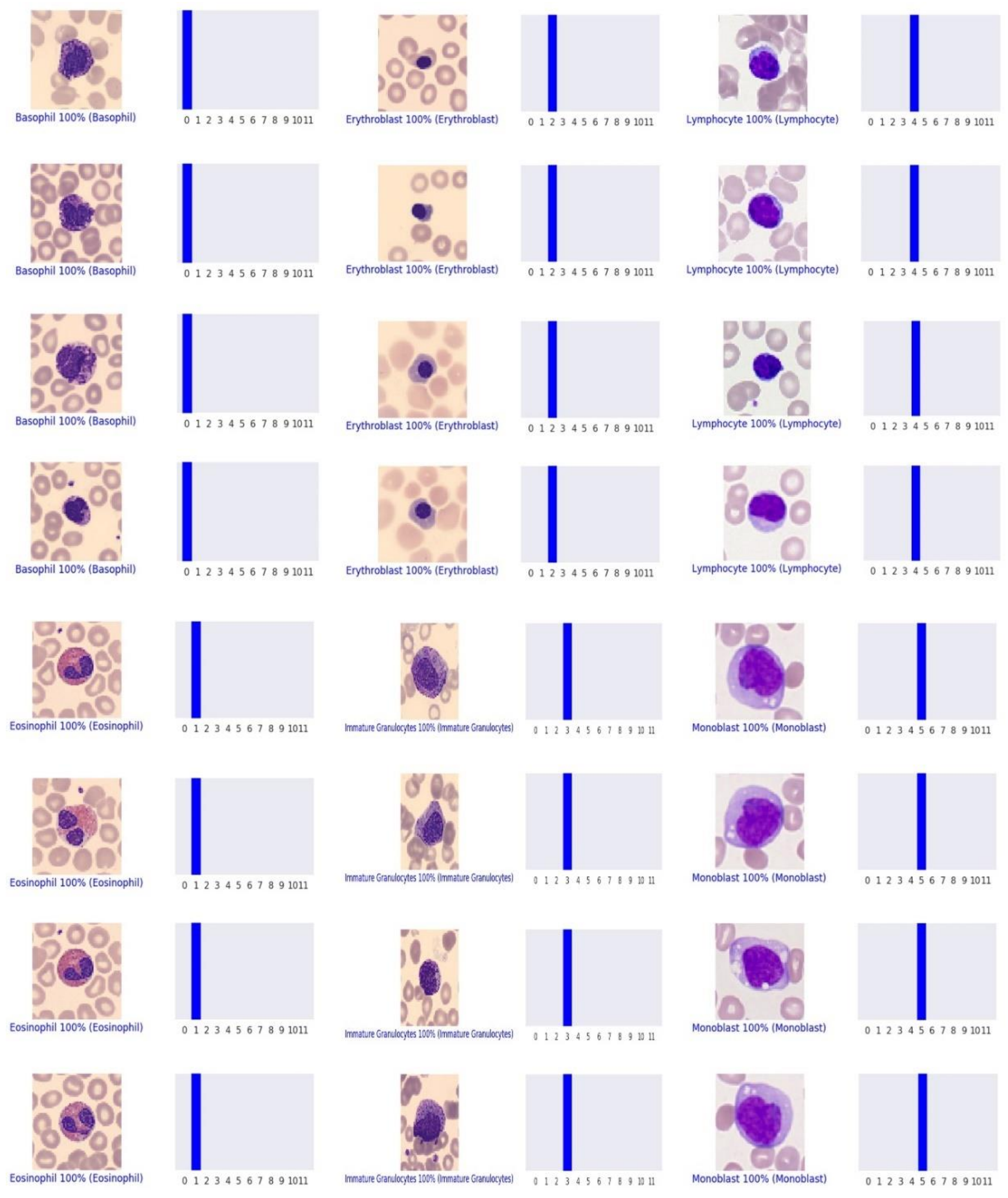
Basophil 100% (Basophil)
Erythroblast 100% (Erythroblast)
Lymphocyte 100% (Lymphocyte)
Eosinophil 100% (Eosinophil)
Immature Granulocytes 100% (Immature Granulocytes)
Monoblast 100% (Monoblast)
0 1 2 3 4 5 6 7 8 9 10 11

Figure 55: Samples of classification result for basophil, eosinophil, erythroblast, immature granulocyte, Lymphocyte and monoblast

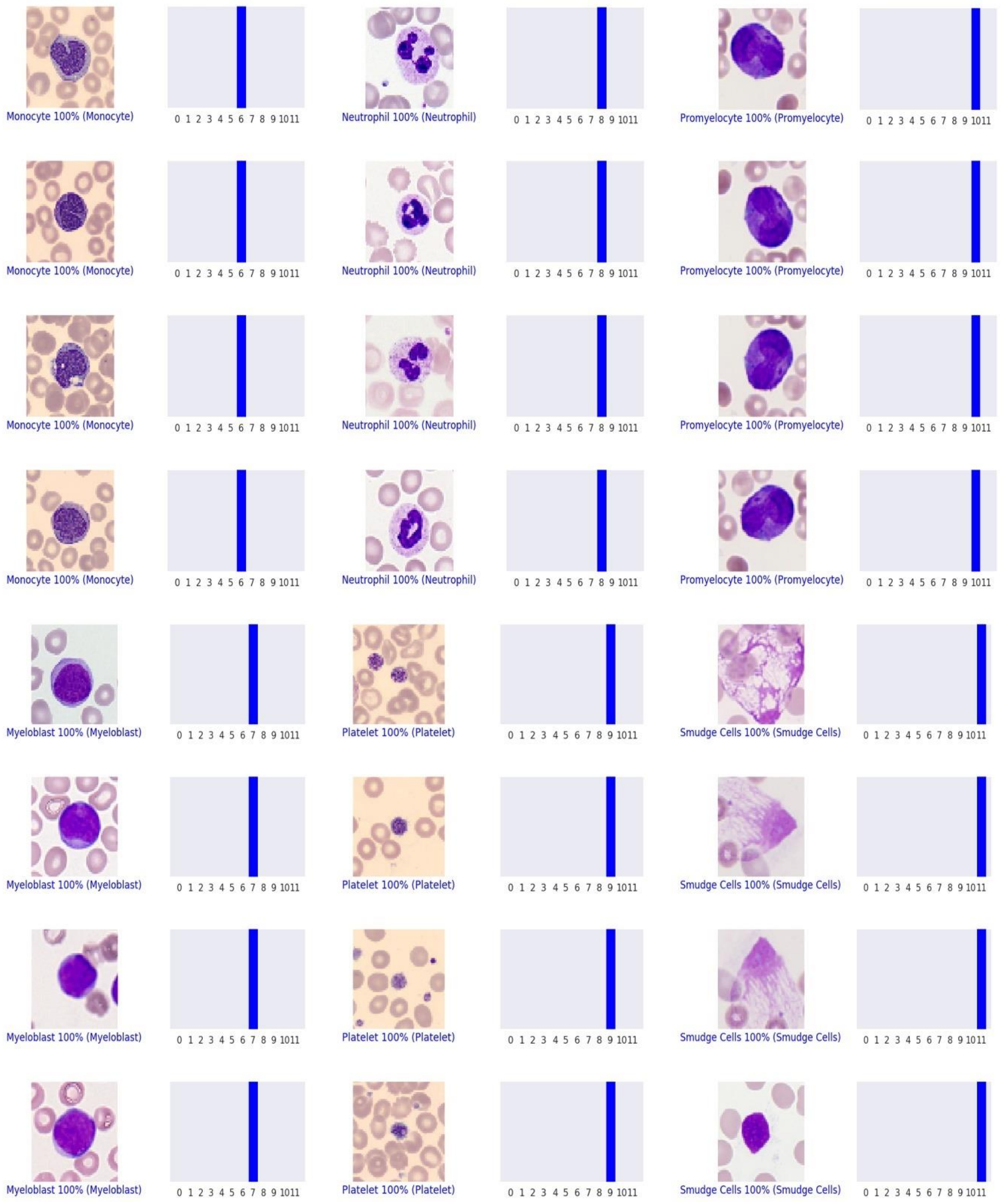
Monocyte 100% (Monocyte)
Neutrophil 100% (Neutrophil)
Promyelocyte 100% (Promyelocyte)
Myeloblast 100% (Myeloblast)
Platelet 100% (Platelet)
Smudge Cells 100% (Smudge Cells)
0 1 2 3 4 5 6 7 8 9 10 11

Figure 56: Samples of classification result for monocyte, myeloblast, neutrophil, platelet, promyelocyte and smudge cells

In Figure 57, we show the screen grabs of some incorrect result or misclassification result. As mentioned earlier, the cell type in the bracket is the original cell type mentioned in the dataset. For example, in Figure 57, for the first image original cell type is basophil but SENet-154-GE misclassified it as neutrophil with a confidence rate of 71% but it was 29% it classified it as basophil. In the second image the original cell type was lymphocyte, but SENet-154-GE was confused between three cell types: myeloblast, lymphocyte and monocyte. For the third image, the original cell type was erythroblast, but SENet-154-GE misclassified it as promyelocyte with 97% confidence rate.

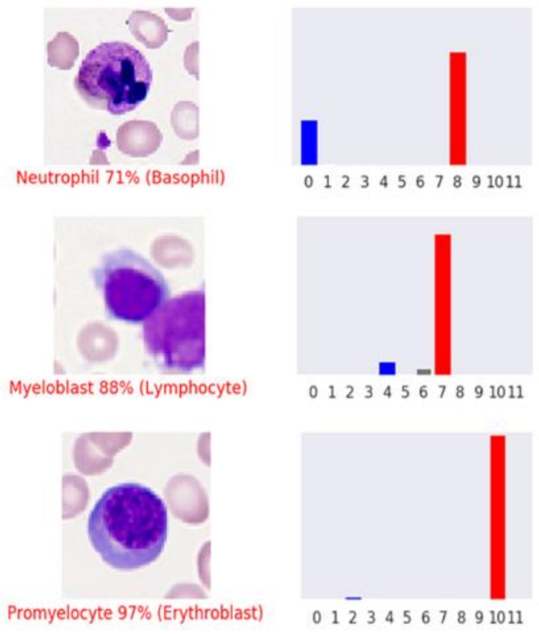
Neutrophil 71% (Basophil)
0 1 2 3 4 5 6 7 8 9 10 11
Myeloblast 88% (Lymphocyte)
0 1 2 3 4 5 6 7 8 9 10 11
Promyelocyte 97% (Erythroblast)
0 1 2 3 4 5 6 7 8 9 10 11

Figure 57: Screen grabs of misclassification result by SENet-154-GE

Chapter 5: Conclusion and Discussion

For robust and automated diagnosis, the goal of various deep learning researchers is to build accurate and efficient segmentation and classification model for white blood cell and red blood cell [155,156]. Deep learning model requires powerful and compatible resources and often that leads to barriers in accessibility [157]. Also, data imbalance and data scarcity are a major problem that complicates the training of deep learning models or algorithms [158]. In medical image analysis, especially for work dealing with blood cells, stain plays an important role that can affect the accuracy of any classification model. If the dataset contains multiple stain colors, sometimes it becomes difficult for the classification model to identify similar cell types with different stain colors. In our work, we first collect microscopic single cell images obtained from peripheral blood smears, from two different sources. We took the help of medical experts to eliminate redundancy of data and to filter the data in the data preprocessing stage.

After data preprocessing, we perform stain normalization through modified cycle consistency GAN since our dataset is from two different sources with two different stain colors. We compare the number of images only between the common and similar cell type in the two datasets. The cell type containing less images in a dataset is stain normalized to the stain color of the other dataset containing higher number of images for that particular cell type. After stain normalization, we merge the two datasets to form a single dataset. We merge images from the same cell type and also cell types belonging to the same cell family to form a single cell type or group. The final dataset contained images of twelve cell types. After forming the final dataset, we use a classifier-based GAN (C-WGAN-GP) network to generate images for cell types containing more than thousand images. For eight cell types, we used our proposed C-WGAN-GP model that consists of a generator, a discriminator and a classifier. The generator in C-WGAN-GP is trained from the feedback of both the discriminator and the classifier. For cell types containing less than hundred images, we implemented few-shot image generation that follows meta learning concept. We perform few-shot image generation for three cell types. Neutrophil was excluded from the image generation phase since it already contained a lot of images. After few-shot image generation, we combined the original and the synthetic data to form the balanced dataset. The balanced dataset is then fed to the proposed SENet-154-GE classification model. The combination of balanced dataset and SENet-154-GE model achieves a classification accuracy of 98.4% which is 6% more than the original data.

We presented the evaluation results for stain normalization module with FID, SSIM and inception score. For C-WGAN-GP, we used precision, recall, F1-score, LPIPS, FID and IS as evaluation metrics. Also, we used PSNR, SSIM and L1 and L2 for evaluating the quality of the generated images. For few-shot image generation we used FID, LPIPS and IS as evaluation metrics. Also, we have compared the time of training, the number of episodes and the number of parameters for our proposed model with different existing few-shot image generation models. At last, we evaluated our proposed SENet-154-GE classification model. Results for each section shows how our proposed approach achieves better outputs quantitatively and qualitatively compared to existing methodologies. For each module, we also provided the visual results of the images showing the transformation happening from each proposed module. Results show that our GAN models and classification model performs significantly better than existing models. But in this section, we would also like to discuss how removing each module would affect the performance of the classification model and to what extent it would impact the accuracy of the classification model. We perform an ablation study to demonstrate the importance of each module in our proposed methodology. In Figure 58, we show the results for ablation study, i.e., how the accuracy of SENet-154-GE classification model is impacted when individual modules are not implemented or included in the proposed methodology. Ablation refers to the removal of AI components to investigate how a system would perform if certain components are removed. This study helps in understanding the contribution of every component in a proposed system. The overall accuracy of the model is 98.4%, but when the GE module from the proposed SENet-154-GE classification model is removed and only SENet-154 model is used for classification, the accuracy reduces to 96.9%, which indicates the GE module is relevant in improving the accuracy of the proposed approach. When the stain normalization module is removed and we use images of same cell types with different stain colors, the accuracy reduces to 95.8%. Stain normalization is important as different stain colors has different intensities and leads to misclassification. We also experimented by using only few-shot image generation process for each cell type rather than using C-WGAN-GP. Results show that the accuracy reduces by 3.4% from the original accuracy. Removing C-WGAN-GP, we achieved an accuracy of 95%. But at the same time if we try to generate images of the cell type containing less than hundred images by using C-WGAN-GP, the accuracy of the classification model is further reduced to 94.3% which depicts that few-shot image generation plays an important role in enhancing the accuracy of our proposed approach.

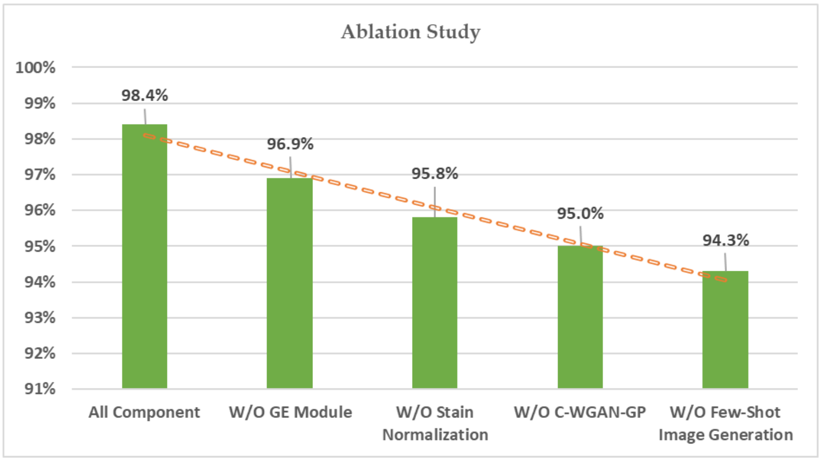

Figure 58: Ablation study-impact on accuracy on removing individual modules

In Figure 59, Figure 60, Figure 61 and Figure 62, we demonstrate how the removal of individual modules impact the classification accuracy, specificity and sensitivity of each cell type. In Figure 59, we show the results on removing the stain normalization module from the proposed methodology. As we can see from the figure, accuracy specificity and sensitivity reduce for the cell types basophil, eosinophil, erythroblast, immature granulocyte, lymphocyte and monocyte for which we had performed stain normalization in the actual proposed methodology for enhancing classification of microscopic single-cell images obtained from peripheral blood smears. For each of this mentioned cell types, the accuracy, specificity and sensitivity reduce by almost 5%.

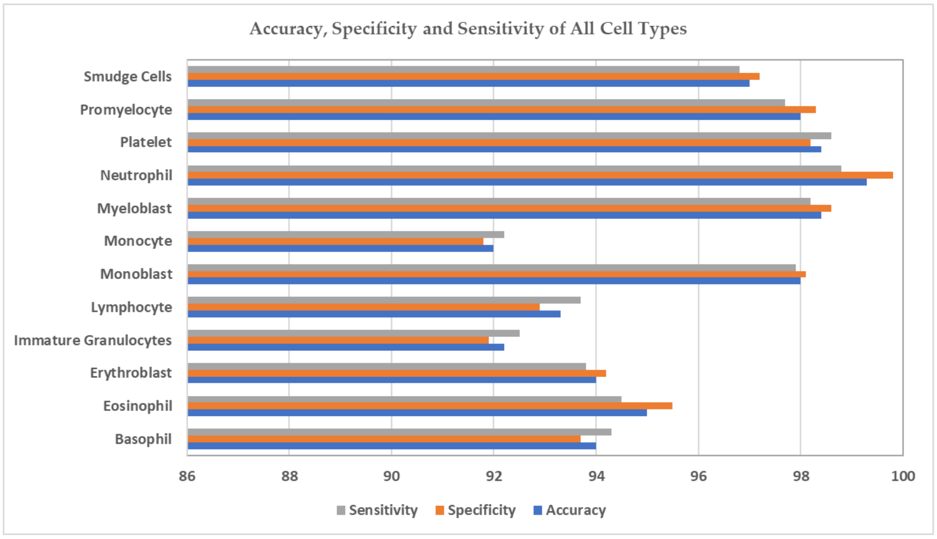

Figure 59: Impact on accuracy, specificity and sensitivity of individual cell type on removing stain normalization module

In Figure 60, we present the impact on classification accuracy, specificity and sensitivity of individual cell types on removing the GE module proposed in SENet-154-GE classification model. The overall accuracy of the model reduces by 1.5% on removing the GE module from the classification model. For all the cell types, almost 1-2.5% classification accuracy is reduced and so as the value for specificity and sensitivity.

Similarly, we remove the few-shot image generation module from the proposed approach to

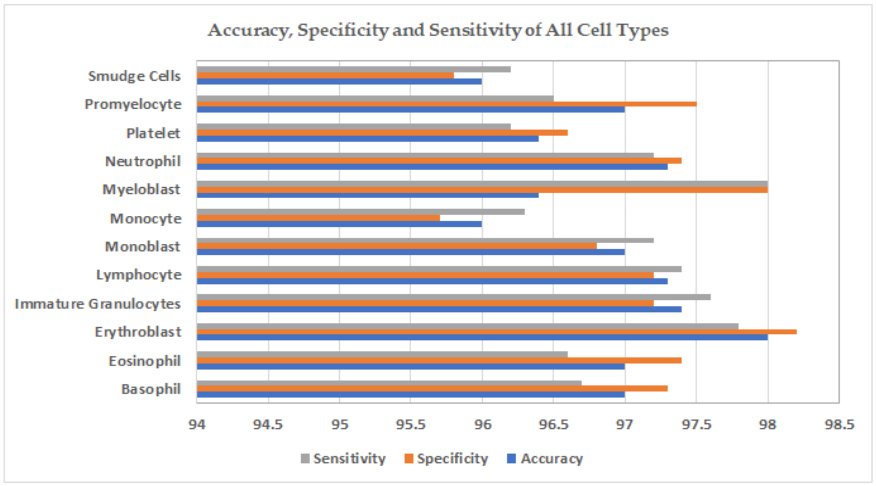

Figure 60: Impact on accuracy, specificity and sensitivity of individual cell type on removing the GE module from SENet-154-GE classification model

present its impact on accuracy, specificity and sensitivity of each cell type. We try to generate images of each cell type through C-WGAN-GP model. As we can see from Figure 61, accuracy, specificity and sensitivity significantly reduces for the cell types containing less than hundred images, i.e., smudge cells, promyelocyte and monocyte. This indicates that few-shot image generation plays an important role in improving the performance of the classification model an is a necessary inclusion for the proposed approach.

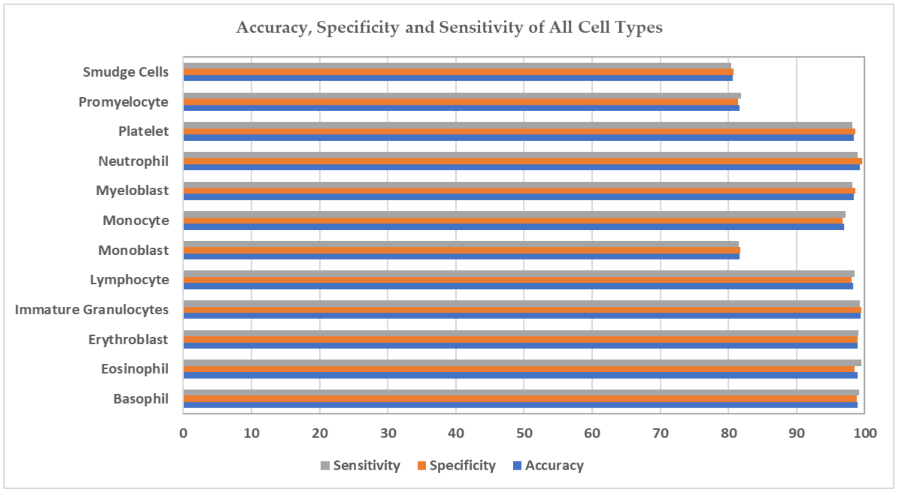

Figure 61: Impact on accuracy, specificity and sensitivity of individual cell type on removing the few-shot image generation model

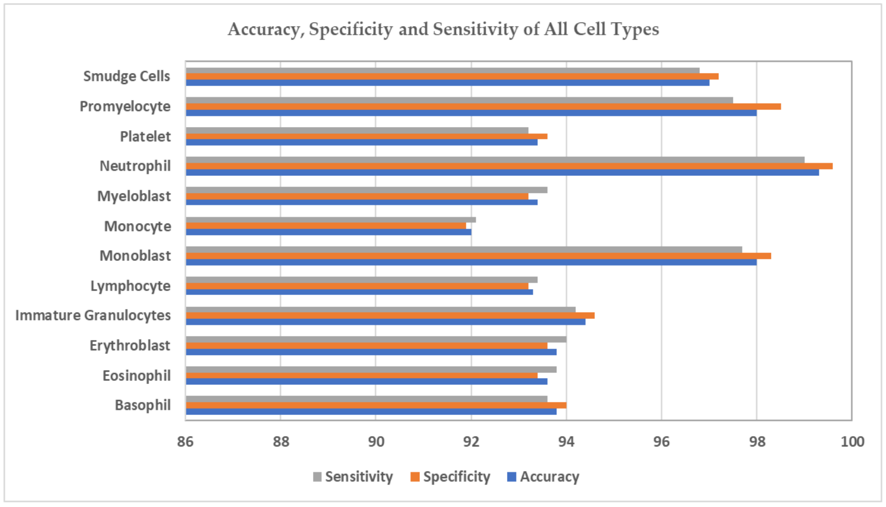

Figure 62:Impact on accuracy, specificity and sensitivity of individual cell type on removing the C-WGAN-GP model from the proposed methodology

Also, we have tried to generate images of each cell type using the few-shot image generation module by removing the C-WGAN-GP model as shown in Figure 62. The overall classification accuracy reduces by 4.1% indicating that C-WGAN-GP performs better in scenarios where there are more than thousand images present for every cell type. The ablation study helps us in demonstrating the importance and necessity of each module in our proposed approach for enhancing classification of microscopic single-cell images obtained from peripheral blood smears.

Single-cell images are used for training and testing of machine learning and deep learning models for microscopic image-based hematological diagnosis [159-164]. It can also be used for training models for segmentation as well as automatic classification of peripheral blood cells that helps in detecting abnormalities in cells [160-163]. Single-cell image dataset can be used as a model weight initializer. This means we can use the available images to pre-train learning models, which can be further trained for disease diagnosis [159-164]. We plan to apply our proposed SENet-154-GE model for classifying multiple cells in a single image in our future studies. To apply SENet-154-GE for multiple cell images, we need annotated data. We need to prepare .xml files of annotation and train the model with the labeled images. Finetuning the model by changing the last or the classification layer can help in generating the desired output for multiple cell classification in a single image.

References

1) Saleem, Saba, Javeria Amin, Muhammad Sharif, Muhammad Almas Anjum, Muhammad Iqbal, and Shui-Hua Wang. "A deep network designed for segmentation and classification of leukemia using fusion of the transfer learning models." Complex & Intelligent Systems (2021): 1-16.

2) https://www.verywellhealth.com/blood-smear-uses-and-results-4586471

3) https://www.vectorstock.com/royalty-free-vector/type-blood-cell-vector-34580213

4) https://towardsdatascience.com/will-we-ever-solve-the-shortage-of-data-in-medical-applications-70da163e2c2d

5) https://towardsdatascience.com/ai-in-medical-diagnosis-dealing-with-medical-datasets-b746e8bda9e5#:~:text=Medical%20datasets%20have%20one%20big,using%20the%20models%20we%20build.

6) https://www.ibm.com/topics/artificial-intelligence-medicine

7) https://www.mantiscope.com/

8) Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).

9) Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired image-to-image translation using cycle-consistent adversarial networks." In Proceedings of the IEEE international conference on computer vision, pp. 2223-2232. 2017.

10) Litjens, Geert, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I. Sánchez. "A survey on deep learning in medical image analysis." *Medical image analysis* 42 (2017): 60-88.

11) Cheplygina, V., de Bruijne, M. & Pluim, J. P. W. Not-so-supervised: a survey of semi-supervised, multi-instance, and transfer learning in medical image analysis. Med. Image Anal. 54, 280–296 (2019).

12) Zhou, S. K. et al. A review of deep learning in medical imaging: Image traits, technology trends, case studies with progress highlights, and future promises. Proceedings of the IEEE1-19 (2020).

13) Kim, Hee E., Alejandro Cosa-Linan, Nandhini Santhanam, Mahboubeh Jannesari, Mate E. Maros, and Thomas Ganslandt. "Transfer learning for medical image classification: a literature review." BMC Medical Imaging 22, no. 1 (2022): 1-13.

14) Zhang Y, Zhang S et al (2016) Theano: A Python framework for fast computation of mathematical expressions, arXiv e-prints, abs/1605.02688. http://arxiv.org/abs/1605.02688.

15) Pang S, Yang X (2016) Deep Convolutional Extreme learning Machine and its application in Handwritten Digit Classifcation. Hindawi Publ Corp Comput Intell Neurosci 2016:3049632. https://doi.org/10.1155/2016/3049632.

16) Liu, Xiaoqing, Kunlun Gao, Bo Liu, Chengwei Pan, Kongming Liang, Lifeng Yan, Jiechao Ma et al. "Advances in deep learning-based medical image analysis." *Health Data Science* 2021 (2021).

17) Cheplygina, Veronika, Marleen de Bruijne, and Josien PW Pluim. "Not-so-supervised: a survey of semi-supervised, multi-instance, and transfer learning in medical image analysis." *Medical image analysis* 54 (2019): 280-296.

18) Weese, Jürgen, and Cristian Lorenz. "Four challenges in medical image analysis from an industrial perspective." *Medical image analysis* 33 (2016): 44-49.

19) De Bruijne, Marleen. "Machine learning approaches in medical image analysis: From detection to diagnosis." *Medical image analysis* 33 (2016): 94-97.

20) Azizi, Shekoofeh, Basil Mustafa, Fiona Ryan, Zachary Beaver, Jan Freyberg, Jonathan Deaton, Aaron Loh et al. "Big self-supervised models advance medical image classification." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3478-3488. 2021.

21) Quellec, Gwenolé, Guy Cazuguel, Béatrice Cochener, and Mathieu Lamard. "Multiple-instance learning for medical image and video analysis." *IEEE reviews in biomedical engineering* 10 (2017): 213-234.

22) Suganyadevi, S., V. Seethalakshmi, and K. Balasamy. "A review on deep learning in medical image analysis." *International Journal of Multimedia Information Retrieval* 11, no. 1 (2022): 19-38.

23) Uria, Benigno, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. "Neural autoregressive distribution estimation." *The Journal of Machine Learning Research* 17, no. 1 (2016): 7184-7220.

24) Germain, Mathieu, Karol Gregor, Iain Murray, and Hugo Larochelle. "Made: Masked autoencoder for distribution estimation." In *International Conference on Machine Learning*, pp. 881-889. PMLR, 2015.

25) Goodfellow, Ian. "Nips 2016 tutorial: Generative adversarial networks." *arXiv preprint arXiv:1701.00160* (2016).

26) Van Oord, Aaron, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." In *International conference on machine learning*, pp. 1747-1756. PMLR, 2016.

27) Van den Oord, Aaron, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, and Alex Graves. "Conditional image generation with pixelcnn decoders." *Advances in neural information processing systems* 29 (2016).

28) Yi X, Walia E, Babyn P. Generative adversarial network in medical imaging: A review. Medical image analysis. 2019 Dec 1;58:101552.

29) Brock A, Donahue J, Simonyan K. Large scale GAN training for high fidelity natural

image synthesis. arXiv preprint arXiv:1809.11096. 2018 Sep 28.

30) Karras T, Aila T, Laine S, Lehtinen J. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196. 2017 Oct 27.

31) Han C, Murao K, Noguchi T, Kawata Y, Uchiyama F, Rundo L, Nakayama H, Satoh SI. Learning more with less: conditional PGGAN-based data augmentation for brain metastases detection using highly-rough annotation on MR images. InProceedings of the 28th ACM International Conference on Information and Knowledge Management 2019 Nov 3 (pp. 119-127)

32) Karras T, Laine S, Aila T. A style-based generator architecture for generative adversarial networks. InProceedings of the IEEE/CVF conference on computer vision and pattern recognition 2019 (pp. 4401-4410).

33) Wang J, Perez L. The effectiveness of data augmentation in image classification using deep learning. Convolutional Neural Networks Vis. Recognit. 2017 Dec;11:1-8.

34) Frid-Adar M, Diamant I, Klang E, Amitai M, Goldberger J, Greenspan H. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. Neurocomputing. 2018 Dec 10;321:321-31.

35) Bowles C, Chen L, Guerrero R, Bentley P, Gunn R, Hammers A, Dickie DA, Hernández MV, Wardlaw J, Rueckert D. Gan augmentation: Augmenting training data using generative adversarial networks. arXiv preprint arXiv:1810.10863. 2018 Oct 25.

36) Han C, Rundo L, Araki R, Furukawa Y, Mauri G, Nakayama H, Hayashi H. Infinite brain MR images: PGGAN-based data augmentation for tumor detection. InNeural approaches to dynamics of signal exchanges 2020 (pp. 291-303). Springer, Singapore.

37) Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC. Improved training of wasserstein gans. Advances in neural information processing systems. 2017;30.

38) Mao X, Li Q, Xie H, Lau RY, Wang Z, Paul Smolley S. Least squares generative adversarial networks. InProceedings of the IEEE international conference on computer vision 2017 (pp. 2794-2802).

39) Mescheder L, Geiger A, Nowozin S. Which training methods for GANs do actually converge?. InInternational conference on machine learning 2018 Jul 3 (pp. 3481-3490). PMLR.

40) Miyato T, Kataoka T, Koyama M, Yoshida Y. Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957. 2018 Feb 16.

41) Miyato T, Koyama M. cGANs with projection discriminator. arXiv preprint arXiv:1802.05637. 2018 Feb 15.

42) Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434. 2015 Nov 19.

43) Zhang H, Goodfellow I, Metaxas D, Odena A. Self-attention generative adversarial networks. InInternational conference on machine learning 2019 May 24 (pp. 7354-7363).

PMLR.

44) Denton EL, Chintala S, Fergus R. Deep generative image models using a laplacian pyramid of adversarial networks. Advances in neural information processing systems. 2015;28.

45) Karras T, Aila T, Laine S, Lehtinen J. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196. 2017 Oct 27.

46) Liu S, Wang T, Bau D, Zhu JY, Torralba A. Diverse image generation via self-conditioned gans. InProceedings of the IEEE/CVF conference on computer vision and pattern recognition 2020 (pp. 14286-14295).

47) Brock A, Donahue J, Simonyan K. Large scale GAN training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096. 2018 Sep 28.

48) Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T. Analyzing and improving the image quality of stylegan. InProceedings of the IEEE/CVF conference on computer vision and pattern recognition 2020 (pp. 8110-8119).

49) Chen T, Zhai X, Ritter M, Lucic M, Houlsby N. Self-supervised gans via auxiliary rotation loss. InProceedings of the IEEE/CVF conference on computer vision and pattern recognition 2019 (pp. 12154-12163).

50) Reinhard E, Adhikhmin M, Gooch B, Shirley P. Color transfer between images. IEEE Computer graphics and applications. 2001 Jul;21(5):34-41.

51) Macenko M, Niethammer M, Marron JS, Borland D, Woosley JT, Guan X, Schmitt C, Thomas NE. A method for normalizing histology slides for quantitative analysis. In2009 IEEE international symposium on biomedical imaging: from nano to macro 2009 Jun 28 (pp. 1107-1110). IEEE.

52) Khan AM, Rajpoot N, Treanor D, Magee D. A nonlinear mapping approach to stain normalization in digital histopathology images using image-specific color deconvolution. IEEE Transactions on Biomedical Engineering. 2014 Jan 28;61(6):1729-38.

53) Bejnordi BE, Litjens G, Timofeeva N, Otte-Höller I, Homeyer A, Karssemeijer N, van der Laak JA. Stain specific standardization of whole-slide histopathological images. IEEE transactions on medical imaging. 2015 Sep 4;35(2):404-15.

54) BenTaieb A, Hamarneh G. Adversarial stain transfer for histopathology image analysis. IEEE transactions on medical imaging. 2017 Dec 8;37(3):792-802.

55) Kang H, Luo D, Feng W, Zeng S, Quan T, Hu J, Liu X. Stainnet: a fast and robust stain normalization network. Frontiers in Medicine. 2021;8.

56) Cai S, Xue Y, Gao Q, Du M, Chen G, Zhang H, Tong T. Stain style transfer using transitive adversarial networks. InInternational Workshop on Machine Learning for Medical Image Reconstruction 2019 Oct 17 (pp. 163-172). Springer, Cham.

57) Salehi P, Chalechale A. Pix2pix-based stain-to-stain translation: a solution for robust stain normalization in histopathology images analysis. In2020 International Conference on Machine Vision and Image Processing (MVIP) 2020 Feb 18 (pp. 1-7). IEEE.

58) Shaban MT, Baur C, Navab N, Albarqouni S. Staingan: Stain style transfer for digital histological images. In2019 Ieee 16th international symposium on biomedical imaging (Isbi 2019) 2019 Apr 8 (pp. 953-956). IEEE.

59) Tellez D, Litjens G, Bándi P, Bulten W, Bokhorst JM, Ciompi F, Van Der Laak J. Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology. Medical image analysis. 2019 Dec 1;58:101544.

60) Cho H, Lim S, Choi G, Min H. Neural stain-style transfer learning using GAN for histopathological images. arXiv preprint arXiv:1710.08543. 2017 Oct 23.

61) Zheng Y, Jiang Z, Zhang H, Xie F, Hu D, Sun S, Shi J, Xue C. Stain standardization capsule for application-driven histopathological image normalization. IEEE journal of biomedical and health informatics. 2020 Mar 30;25(2):337-47.

62) Chen X, Yu J, Chen L, Zeng S, Liu X, Cheng S. Multi-stage domain adversarial style reconstruction for cytopathological image stain normalization. arXiv preprint arXiv:1909.05184. 2019 Sep 11.

63) Gupta A, Duggal R, Gehlot S, Gupta R, Mangal A, Kumar L, Thakkar N, Satpathy D. GCTI-SN: Geometry-inspired chemical and tissue invariant stain normalization of microscopic medical images. Medical Image Analysis. 2020 Oct 1;65:101788.

64) Hoque MZ, Keskinarkaus A, Nyberg P, Seppänen T. Retinex model based stain normalization technique for whole slide image analysis. Computerized Medical Imaging and Graphics. 2021 Jun 1;90:101901.

65) Liang H, Plataniotis KN, Li X. Stain style transfer of histopathology images via structure-preserved generative learning. InInternational Workshop on Machine Learning for Medical Image Reconstruction 2020 Oct 8 (pp. 153-162). Springer, Cham.

66) Hoque MZ, Keskinarkaus A, Nyberg P, Seppänen T. Retinex model based stain normalization technique for whole slide image analysis. Computerized Medical Imaging and Graphics. 2021 Jun 1;90:101901.

67) Janowczyk A, Basavanhally A, Madabhushi A. Stain normalization using sparse autoencoders (StaNoSA): application to digital pathology. Computerized Medical Imaging and Graphics. 2017 Apr 1;57:50-61.

68) Mahapatra D, Bozorgtabar B, Thiran JP, Shao L. Structure preserving stain normalization of histopathology images using self supervised semantic guidance. InInternational Conference on Medical Image Computing and Computer-Assisted Intervention 2020 Oct 4 (pp. 309-319). Springer, Cham.

69) Sun Q, Liu Y, Chua TS, Schiele B. Meta-transfer learning for few-shot learning. InProceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019 (pp. 403-412).

70) Walsh R, Abdelpakey MH, Shehata MS, Mohamed MM. Automated human cell classification in sparse datasets using few-shot learning. Scientific Reports. 2022 Feb 21;12(1):1-1.

71) Fei-Fei L, Fergus R, Perona P. One-shot learning of object categories. IEEE transactions on pattern analysis and machine intelligence. 2006 Feb 21;28(4):594-611.

72) Shaban A, Bansal S, Liu Z, Essa I, Boots B. One-shot learning for semantic segmentation. arXiv preprint arXiv:1709.03410. 2017 Sep 11.

73) Rakelly K, Shelhamer E, Darrell T, Efros AA, Levine S. Few-shot segmentation propagation with guided networks. arXiv preprint arXiv:1806.07373. 2018 May 25.

74) Rakelly K, Shelhamer E, Darrell T, Efros A, Levine S. Conditional networks for few-shot semantic segmentation.

75) Zhang X, Wei Y, Yang Y, Huang TS. Sg-one: Similarity guidance network for one-shot semantic segmentation. IEEE Transactions on Cybernetics. 2020 Jun 4;50(9):3855-65.

76) Hu T, Yang P, Zhang C, Yu G, Mu Y, Snoek CG. Attention-based multi-context guiding for few-shot semantic segmentation. InProceedings of the AAAI conference on artificial intelligence 2019 Jul 17 (Vol. 33, No. 01, pp. 8441-8448).

77) Dong N, Xing EP. Few-shot semantic segmentation with prototype learning. InBMVC 2018 Sep (Vol. 3, No. 4).

78) Wang K, Liew JH, Zou Y, Zhou D, Feng J. Panet: Few-shot image semantic segmentation with prototype alignment. InProceedings of the IEEE/CVF International Conference on Computer Vision 2019 (pp. 9197-9206).

79) Feng R, Zheng X, Gao T, Chen J, Wang W, Chen DZ, Wu J. Interactive Few-Shot Learning: Limited Supervision, Better Medical Image Segmentation. IEEE Transactions on Medical Imaging. 2021 Feb 19;40(10):2575-88.

80) Tang H, Liu X, Sun S, Yan X, Xie X. Recurrent mask refinement for few-shot medical image segmentation. InProceedings of the IEEE/CVF International Conference on Computer Vision 2021 (pp. 3918-3928).

81) Robb E, Chu WS, Kumar A, Huang JB. Few-shot adaptation of generative adversarial networks. arXiv preprint arXiv:2010.11943. 2020 Oct 22.

82) Li Y, Zhang R, Lu J, Shechtman E. Few-shot image generation with elastic weight consolidation. arXiv preprint arXiv:2012.02780. 2020 Dec 4.

83) Ojha U, Li Y, Lu J, Efros AA, Lee YJ, Shechtman E, Zhang R. Few-shot image generation via cross-domain correspondence. InProceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021 (pp. 10743-10752).

84) Xiao J, Li L, Wang C, Zha ZJ, Huang Q. Few Shot Generative Model Adaption via Relaxed Spatial Structural Alignment. arXiv preprint arXiv:2203.04121. 2022 Mar 6.

85) Bartunov S, Vetrov D. Few-shot generative modelling with generative matching networks. InInternational Conference on Artificial Intelligence and Statistics 2018 Mar 31 (pp. 670-678). PMLR.

86) Liang W, Liu Z, Liu C. Dawson: A domain adaptive few shot generation framework. arXiv preprint arXiv:2001.00576. 2020 Jan 2.

87) Wang Y, Wu C, Herranz L, van de Weijer J, Gonzalez-Garcia A, Raducanu B. Transferring gans: generating images from limited data. InProceedings of the European Conference on Computer Vision (ECCV) 2018 (pp. 218-234).

88) Mo S, Cho M, Shin J. Freeze the discriminator: a simple baseline for fine-tuning gans. arXiv preprint arXiv:2002.10964. 2020 Feb 25.

89) Noguchi A, Harada T. Image generation from small datasets via batch statistics adaptation. InProceedings of the IEEE/CVF International Conference on Computer Vision 2019 (pp. 2750-2758).

90) Clouâtre L, Demers M. Figr: Few-shot image generation with reptile. arXiv preprint arXiv:1901.02199. 2019 Jan 8.

91) Lake B, Salakhutdinov R, Gross J, Tenenbaum J. One shot learning of simple visual concepts. InProceedings of the annual meeting of the cognitive science society 2011 (Vol. 33, No. 33).

92) Litjens G, Kooi T, Bejnordi BE, Setio AA, Ciompi F, Ghafoorian M, Van Der Laak JA, Van Ginneken B, Sánchez CI. A survey on deep learning in medical image analysis. Medical image analysis. 2017 Dec 1;42:60-88.

93) Shen D, Wu G, Suk HI. Deep learning in medical image analysis. Annual review of biomedical engineering. 2017 Jun 21;19:221-48.

94) Faes L, Wagner SK, Fu DJ, Liu X, Korot E, Ledsam JR, Back T, Chopra R, Pontikos N, Kern C, Moraes G. Automated deep learning design for medical image classification by health-care professionals with no coding experience: a feasibility study. The Lancet Digital Health. 2019 Sep 1;1(5):e232-42.

95) Liu Y, Jain A, Eng C, Way DH, Lee K, Bui P, Kanada K, de Oliveira Marinho G, Gallegos J, Gabriele S, Gupta V. A deep learning system for differential diagnosis of skin diseases. Nature medicine. 2020 Jun;26(6):900-8.

96) McKinney SM, Sieniek M, Godbole V, Godwin J, Antropova N, Ashrafian H, Back T, Chesus M, Corrado GS, Darzi A, Etemadi M. International evaluation of an AI system for breast cancer screening. Nature. 2020 Jan;577(7788):89-94.

97) Menegola A, Fornaciali M, Pires R, Bittencourt FV, Avila S, Valle E. Knowledge transfer for melanoma screening with deep learning. In2017 IEEE 14th international symposium on biomedical imaging (ISBI 2017) 2017 Apr 18 (pp. 297-300). IEEE.

98) Xie H, Shan H, Cong W, Zhang X, Liu S, Ning R, Wang G. Dual network architecture for few-view CT-trained on ImageNet data and transferred for medical imaging. InDevelopments in X-ray Tomography XII 2019 Sep 10 (Vol. 11113, p. 111130V). International Society for Optics and Photonics.

99) Alzubaidi L, Fadhel MA, Al-Shamma O, Zhang J, Santamaría J, Duan Y, R Oleiwi S. Towards a better understanding of transfer learning for medical imaging: a case study. Applied Sciences. 2020 Jan;10(13):4523.

100) Graziani M, Andrearczyk V, Müller H. Visualizing and interpreting feature reuse of

pretrained CNNs for histopathology. InIrish Machine Vision and Image Processing Conference (IMVIP 2019), Dublin, Ireland 2019 Aug.

101) Heker M, Greenspan H. Joint liver lesion segmentation and classification via transfer learning. arXiv preprint arXiv:2004.12352. 2020 Apr 26.

102) Raghu M, Zhang C, Kleinberg J, Bengio S. Transfusion: Understanding transfer learning for medical imaging. Advances in neural information processing systems. 2019;32.

103) He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 770-778).

104) Azizi S, Mustafa B, Ryan F, Beaver Z, Freyberg J, Deaton J, Loh A, Karthikesalingam A, Kornblith S, Chen T, Natarajan V. Big self-supervised models advance medical image classification. InProceedings of the IEEE/CVF International Conference on Computer Vision 2021 (pp. 3478-3488).

105) Saleem, Saba, Javeria Amin, Muhammad Sharif, Muhammad Almas Anjum, Muhammad Iqbal, and Shui-Hua Wang. "A deep network designed for segmentation and classification of leukemia using fusion of the transfer learning models." Complex & Intelligent Systems (2021): 1-16.

106) Acevedo, A.; Merino, A.; Alférez, S.; Molina, Á.; Boldú, L.; Rodellar, J. A dataset of microscopic peripheral blood cell images for development of automatic recognition systems. Data Brief 2020, 30, 105474

107) Matek, C.; Schwarz, S.; Marr, C.; Spiekermann, K. A single-cell morphological dataset of leukocytes from AML patients and non-malignant controls (AML Cytomorphology_LMU). The Cancer Imaging Archive (TCIA)[Internet]. 2019. Available online: https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=61080958 (accessed on 20 April 2022).

108) Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." In ICML deep learning workshop, vol. 2, p. 0. 2015.

109) Hoffer, Elad, and Nir Ailon. "Deep metric learning using triplet network." In International workshop on similarity-based pattern recognition, pp. 84-92. Springer, Cham, 2015.

110) Vinyals, Oriol, Charles Blundell, Timothy Lillicrap, and Daan Wierstra. "Matching networks for one shot learning." Advances in neural information processing systems 29 (2016).

111) Snell, Jake, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning." Advances in neural information processing systems 30 (2017).

112) Santoro, Adam, David Raposo, David G. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. "A simple neural network module for relational reasoning." Advances in neural information processing systems 30 (2017).

113) Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." In International conference on machine learning, pp. 1126-1135. PMLR, 2017.

114) Ravi, Sachin, and Hugo Larochelle. "Optimization as a model for few-shot learning." (2016).

115) Edwards, Harrison, and Amos Storkey. "Towards a neural statistician." arXiv preprint arXiv:1606.02185 (2016).

116) Hariharan, B., and R. Girshick. "Low-shot learning by shrinking and hallucinating features." In IEEE International Conference on Computer Vision. 2017.

117) Ren, Jian, Ilker Hacihaliloglu, Eric A. Singer, David J. Foran, and Xin Qi. "Unsupervised domain adaptation for classification of histopathology whole-slide images." Frontiers in bioengineering and biotechnology 7 (2019): 102.

118) Reinhard, Erik, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. "Color transfer between images." IEEE Computer graphics and applications 21, no. 5 (2001): 34-41.

119) Macenko, Marc, Marc Niethammer, James S. Marron, David Borland, John T. Woosley, Xiaojun Guan, Charles Schmitt, and Nancy E. Thomas. "A method for normalizing histology slides for quantitative analysis." In 2009 IEEE international symposium on biomedical imaging: from nano to macro, pp. 1107-1110. IEEE, 2009.

120) Vahadane, Abhishek, Tingying Peng, Amit Sethi, Shadi Albarqouni, Lichao Wang, Maximilian Baust, Katja Steiger, Anna Melissa Schlitter, Irene Esposito, and Nassir Navab. "Structure-preserving color normalization and sparse stain separation for histological images." IEEE transactions on medical imaging 35, no. 8 (2016): 1962-1971.

121) Khan, Adnan Mujahid, Nasir Rajpoot, Darren Treanor, and Derek Magee. "A nonlinear mapping approach to stain normalization in digital histopathology images using image-specific color deconvolution." IEEE Transactions on Biomedical Engineering 61, no. 6 (2014): 1729-1738.

122) Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A., 2017.
Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028.

123) Li, Chuan, and Michael Wand. "Precomputed real-time texture synthesis with markovian generative adversarial networks." In European conference on computer vision, pp. 702-716. Springer, Cham, 2016.

124) Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).

125) Odena, Augustus, Christopher Olah, and Jonathon Shlens. "Conditional image synthesis with auxiliary classifier gans." In International conference on machine learning, pp. 2642-2651. PMLR, 2017.

126) Phaphuangwittayakul, Aniwat, Yi Guo, and Fangli Ying. "Fast adaptive meta-learning for few-shot image generation." IEEE Transactions on Multimedia (2021).

127) Q. Mao, H. Y. Lee, H. Y. Tseng, S. Ma, and M. H. Yang, “Mode seeking generative adversarial networks for diverse image synthesis,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2019-Jun., 2019, pp. 1429–1437.

128) Liu, Bingchen, Yizhe Zhu, Kunpeng Song, and Ahmed Elgammal. "Towards faster and

stabilized gan training for high-fidelity few-shot image synthesis." In International Conference on Learning Representations. 2020.

129) Dauphin, Yann N., Angela Fan, Michael Auli, and David Grangier. "Language modeling with gated convolutional networks." In International conference on machine learning, pp. 933-941. PMLR, 2017.

130) Xie, Saining, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated residual transformations for deep neural networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1492-1500. 2017.

131) Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." arXiv preprint arXiv:1312.4400 (2013).

132) G. Cantor. Uber unendliche, lineare punktmannich- ¨ faltigkeiten, arbeiten zur mengenlehre aus den jahren 1872-1884. 1884.

133) Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).

134) Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7132-7141. 2018.

135) K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in CVPR, 2016.

136) C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in CVPR, 2016.

137) Wang, Ziyi, Jiewen Xiao, Jingwen Li, Hongjun Li, and Luman Wang. "WBC-AMNet: Automatic classification of WBC images using deep feature fusion network based on focalized attention mechanism." PloS one 17, no. 1 (2022): e0261848.

138) Hu J, Shen L, Albanie S, Sun G, Vedaldi A. Gather-excite: Exploiting feature context in convolutional neural networks. In: 32nd Conference on Neural Information Processing Systems (NeurIPS); 2018.

139) Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." Advances in neural information processing systems 30 (2017).

140) Runz, Marlen, Daniel Rusche, Stefan Schmidt, Martin R. Weihrauch, Jürgen Hesser, and Cleo-Aron Weis. "Normalization of HE-stained histological images using cycle consistent generative adversarial networks." Diagnostic Pathology 16, no. 1 (2021): 1-10.

141) Chen, Xi, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." Advances in neural information processing systems 29 (2016).

142) Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

143) Clouâtre, Louis, and Marc Demers. "Figr: Few-shot image generation with reptile." arXiv preprint arXiv:1901.02199 (2019).

144) Liu, Bingchen, Yizhe Zhu, Kunpeng Song, and Ahmed Elgammal. "Towards faster and stabilized gan training for high-fidelity few-shot image synthesis." In International Conference on Learning Representations. 2020.

145) Zhao, Shengyu, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. "Differentiable augmentation for data-efficient gan training." Advances in Neural Information Processing Systems 33 (2020): 7559-7570.

146) Wang, Yaxing, Abel Gonzalez-Garcia, David Berga, Luis Herranz, Fahad Shahbaz Khan, and Joost van de Weijer. "Minegan: effective knowledge transfer from gans to target domains with few images." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9332-9341. 2020.

147) Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

148) He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.

149) Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826. 2016.

150) Chollet, François. "Xception: Deep learning with depthwise separable convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1251-1258. 2017.

151) Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." In International conference on machine learning, pp. 6105-6114. PMLR, 2019.

152) Xie, Saining, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated residual transformations for deep neural networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1492-1500. 2017.

153) Webb, Geoffrey I., Eamonn Keogh, and Risto Miikkulainen. "Naïve Bayes." Encyclopedia of machine learning 15 (2010): 713-714.

154) Hira, Swati, Anita Bai, and Sanchit Hira. "An automatic approach based on CNN architecture to detect Covid-19 disease from chest X-ray images." Applied Intelligence 51, no. 5 (2021): 2864-2889.

155) Lam, X.H., Ng, K.W., Yoong, Y.J., Ng, S.B., 2021. Wbc-based segmentation and classification on microscopic images: a minor improvement. F1000Research 10, 1168

156) Ryu, D., Kim, J., Lim, D., Min, H.S., Yoo, I.Y., Cho, D., Park, Y., 2021. Label-free white blood cell classification using refractive index tomography and deep learning. BME Frontiers 2021.

157) von Chamier, L., Laine, R.F., Jukkala, J., Spahn, C., Krentzel, D., Nehme, E.,Lerche, M., Hern´andez-P´erez, S., Mattila, P.K., Karinou, E., et al., 2021. Democratising deep learning for microscopy with zerocostdl4mic. Nature communications 12, 1–18.

158) Bansal, M.A., Sharma, D.R., Kathuria, D.M., 2022. A systematic review on data scarcity problem in deep learning: Solution and applications. ACM Computing Surveys (CSUR).

159) Acevedo, Andrea, Anna Merino, Santiago Alférez, Ángel Molina, Laura Boldú, and José Rodellar. "A dataset of microscopic peripheral blood cell images for development of automatic recognition systems." Data in Brief, ISSN: 23523409, Vol. 30,(2020) (2020).

160) Alférez, Santiago, Anna Merino, Laura Bigorra, Luis Mujica, Magda Ruiz, and Jose Rodellar. "Automatic recognition of atypical lymphoid cells from peripheral blood by digital image analysis." American journal of clinical pathology 143, no. 2 (2015): 168-176.

161) Alférez, Santiago, Anna Merino, Andrea Acevedo, Laura Puigví, and José Rodellar. "Color clustering segmentation framework for image analysis of malignant lymphoid cells in peripheral blood." Medical & biological engineering & computing 57, no. 6 (2019): 1265-1283.

162) Rodellar, J., S. Alférez, Andrea Acevedo, A. Molina, and Anna Merino. "Image processing and machine learning in the morphological analysis of blood cells." International journal of laboratory hematology 40 (2018): 46-53.

163) Merino, Anna, Laura Puigví, L. Boldú, S. Alférez, and J. Rodellar. "Optimizing morphology through blood cell image analysis." International journal of laboratory hematology 40 (2018): 54-61.

164) Boldú, Laura, Anna Merino, Santiago Alférez, Angel Molina, Andrea Acevedo, and José Rodellar. "Automatic recognition of different types of acute leukaemia in peripheral blood by image analysis." Journal of Clinical Pathology 72, no. 11 (2019): 755-761.