Advancing Instance Segmentation and WBC Classification in Peripheral Blood Smear through Domain Adaptation: A Study on PBC and the Novel RV-PBS datasets

Jimut Bahan Pal[a,b] (pal.jimut@iitb.ac.in), Aniket Bhattacharyea[c] (aniket@abhattacharyea.dev), Debasis Banerjee[d] (debasis_park@yahoo.co.in), Br. Tamal Maharaj[a] (tamal@gm.rkmvu.ac.in)

[a] Department of Computer Science, Ramakrishna Mission Vivekananda Educational and Research Institute, Howrah, India

[b] Centre for Machine Intelligence and Data Science, Indian Institute of Technology, Bombay, Powai, Mumbai, 400076, India

[c] Department of Mathematics, Ramakrishna Mission Vivekananda Educational and Research Institute, Belur Math, Howrah, India

[d] School of Biological Sciences, Ramakrishna Mission Vivekananda Educational and Research Institute, Belur Math, Howrah, India

Corresponding Author:

Jimut Bahan Pal

CMInDS Fellow & Prime Minister's Research Fellow

Centre for Machine Intelligence and Data Science, Indian Institute of Technology, Bombay, Powai, Mumbai, 400076, India.

Email: pal.jimut@iitb.ac.in

Advancing Instance Segmentation and WBC Classification in Peripheral Blood Smear through Domain Adaptation: A Study on PBC and the Novel RV-PBS datasets

Jimut Bahan Pal[a,b,*], Aniket Bhattacharyea[c], Debasis Banerjee[d], Br. Tamal Maharaj[a]

[a]*Department of Computer Science, Ramakrishna Mission Vivekananda Educational and Research Institute, Howrah, India*
[b]*Centre for Machine Intelligence and Data Science, Indian Institute of Technology, Bombay, Powai, Mumbai, 400076, India.*
[c]*Department of Mathematics, Ramakrishna Mission Vivekananda Educational and Research Institute, Belur Math, Howrah, India*
[d]*School of Biological Sciences, Ramakrishna Mission Vivekananda Educational and Research Institute, Belur Math, Howrah, India*

Abstract

Automating blood cell counting and detection from smear slides holds significant potential for aiding doctors in disease diagnosis through blood tests. However, existing literature has not adequately addressed using whole slide data in this context. This study introduces the novel RV-PBS dataset, comprising ten distinct peripheral blood smear classes, each featuring multiple multi-class White Blood Cells per slide, specifically designed, for instance segmentation benchmarks. While conventional instance segmentation models like Mask R-CNN exhibit promising results in segmenting medical artifact instances, they face challenges in scenarios with limited samples and class imbalances within the dataset. This challenge prompted us to explore innovative techniques such as domain

*Review copy, do not distribute! The codes and datasets will be available at the following URL upon publication: `https://github.com/Jimut123/cellseg` and `https://github.com/Jimut123/RV-PBS`. The current affiliation of the first author is at the Centre for Machine Intelligence and Data Science, Indian Institute of Technology, Bombay, Powai, Mumbai, Maharashtra. This work was a part of M.Sc. thesis while the first author was at RKMVERI.

Email addresses: `pal.jimut@iitb.ac.in` (Jimut Bahan Pal), `aniket@abhattacharyea.dev` (Aniket Bhattacharyea), `debasis_park@yahoo.co.in` (Debasis Banerjee), `tamal@gm.rkmvu.ac.in` (Br. Tamal Maharaj)

adaptation using a similar dataset to enhance the classification accuracy of Mask R-CNN, a novel approach in the domain of medical image analysis. Our study has successfully established a comprehensive pipeline capable of segmenting, detecting, and classifying blood samples from slides, striking an optimal balance between computational complexity and accurate classification of medical artifacts. This advancement enables precise cell counting and classification, facilitating doctors in refining their diagnostic analyses.

Keywords: Automated blood test, detection, domain adaptation, instance segmentation, peripheral blood smear

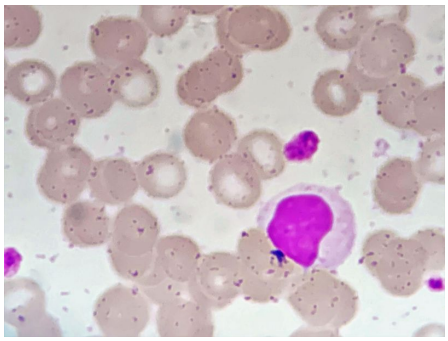
---

1. Introduction

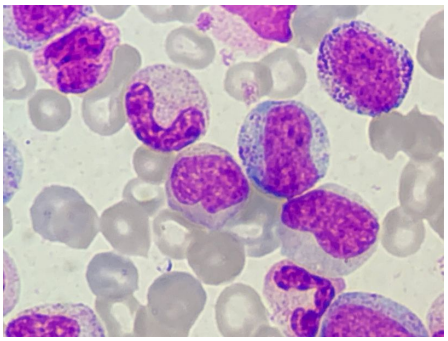
The peripheral blood smear is a procedure that is used to investigate and count blood samples (Linden et al., 2012) under a microscope. Getting the count of different white blood cells (WBCs) may help doctors to diagnose certain diseases [1]. Hematologists regularly receive blood samples (Chadaga et al., 2022; Kukar et al., 2021) to test for diseases. Because of the lack of present research and tools, hematologists manually count (Adewoyin & Nwogoh, 2014) and identify blood cells. This leads to a slow process of generating blood test results. A typical slide of peripheral blood smear collected from our dataset is shown in Figure 1. Two types of slides are present in the proposed dataset. First, annotated instances of smear slides are used for training, as shown in Figure 1a. Second, instances of mixed cells used for testing that are not annotated, as shown in Figure 1b.

This field lacks automation because of the unavailability of a multi-class peripheral blood smear dataset, which can suit all demography. Data is annotated and collected by expert medical practitioners (Bringay et al., 2006). The data also varies according to demography (Endeshaw et al., 2008; Pal, 2022) and the process of creation of blood slides to be investigated under a microscope. In this

[1] `https://www.ucsfhealth.org/medical-tests/wbc-count`



(a) An instance of Lymphocyte cell from the training dataset.



(b) Several instances of cells from the unannotated test dataset.

Figure 1: Samples of typical smear slides containing different White Blood Cells (WBCs) and Red Blood Cells (RBCs) were collected from our novel RV-PBS dataset. We are interested in understanding the type of WBC present on a single slide (as shown in purple). RBCs, usually of biconcave-disc shaped (Hoffman, 2016) and present in enormous amounts, are irrelevant in predicting the class of WBCs through slides.

study, we introduce a novel 10-class segmentation peripheral blood smear dataset comprising about 727 images, the Ramakrishna Vivekananda peripheral blood smear (RV-PBS) dataset. A sample of this dataset is shown in Figure 4. This dataset is created using Leishman staining as stated in 3. Related studies which deal with classifying (Acevedo et al., 2019) and detecting (Kouzehkanan et al., 2022) peripheral blood smears often deal with single-cell per slide. According to the authors' knowledge and at the time of writing, this is the first dataset with many cells per slide for ten different categories. ALL-IDB1 and ALL-IDB2 (Labati et al., 2011) datasets also have many cells per slide but have as little as 108 images which were collected at different magnification ranges, specifically for research in detecting Leukemia. The proposed data simulate actual settings, which can help the doctors get an approximate overview of the different cells in the smear slide. Other studies use a comparatively greater amount of data, i.e., about 40K image samples (Kouzehkanan et al., 2022), which have a single cell per slide with only bounding-box annotations. For this study, our dataset has fewer images, i.e., 727 smear slides from the RV-PBS dataset, containing a different

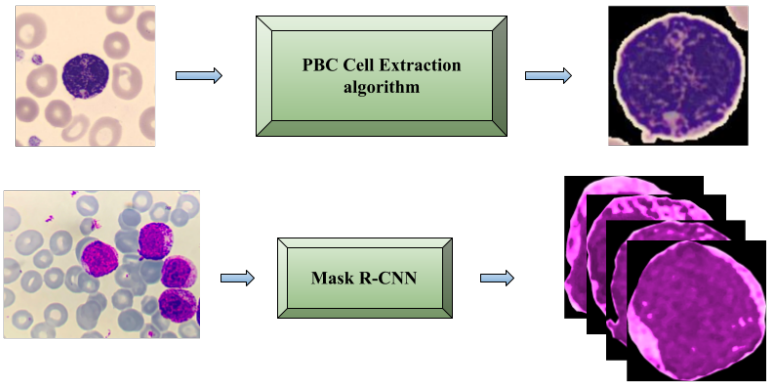


Figure 2: Schematic diagram of the process for extracting cells for domain adaptation pipeline. We use these extracted cells images to pass to the domain adaptation pipeline from both datasets. This will help the model focus on only relevant WBC images rather than unnecessary RBC images. The cells are extracted from the PBC dataset to create C-PBC dataset for domain adaptation using the algorithm discussed in Section 3.4 is shown at the top. We use Mask R-CNN to extract cell images from the RV-PBS dataset to create CRV-PBS dataset. The cell images of the CRV-PBS dataset are passed to the domain adaptation pipeline for improving classification is shown at the bottom.

number of WBCs per slide. The lower number of images and imbalanced classes gives us the opportunity to employ different strategies that help the model attain acceptable performance for benchmarking and deployment.

The process of detection is as follows. Initially, Mask R-CNN (He et al., 2017) is used to segment and detect the images from the RV-PBS dataset. We record the detected classes, i.e., 1 of 10 classes. After segmenting, we extract the cell images for the initial classification pipeline of the 10 classes. Once the initial classification is done, and if we find the classified class to be one of the 8 common classes (with PBC data as discussed later), we send the extracted cell image to the domain adaptation pipeline. The domain adaptation (Ganin & Lempitsky, 2015) pipeline then gives a more refined classification, which helps to increase the detection capability of the Mask R-CNN pipeline. Since Deep Learning models need a tremendous amount of data, we use a relatively larger but similar dataset to perform domain adaptation (Ganin & Lempitsky, 2015)

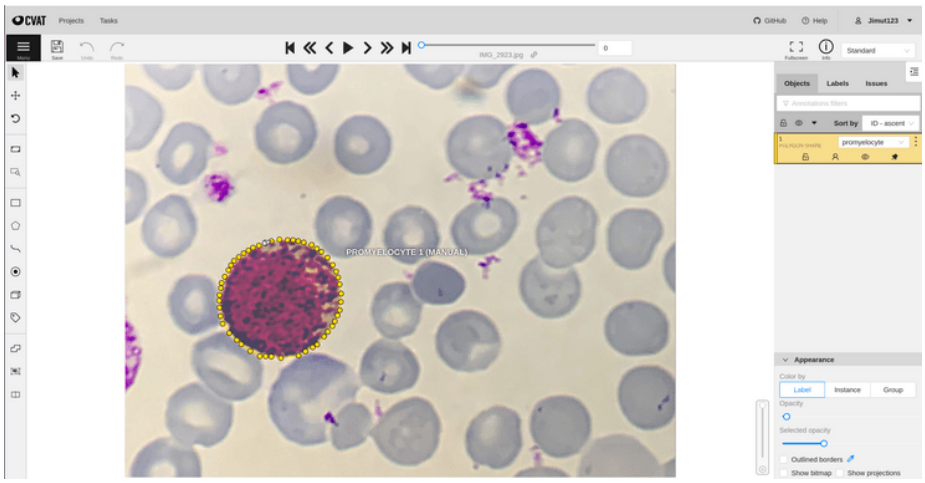


Figure 3: A sample of Basophil cell being annotated using CVAT (Computer Vision Annotation Tool) software to create the RV-PBS dataset.

and improve the current results.

In this paper, we use a similar dataset named PBC normal DIB dataset (Acevedo et al., 2020, 2019), commonly known as the PBC dataset comprising 17092 images. This dataset shares 8 common classes with the RV-PBS dataset. This is the reason for choosing only this dataset for the domain adaptation pipeline since it shares the maximum common classes with our proposed dataset which is available openly. There are no known existing datasets, according to the knowledge of the authors and at the time of writing this paper, which has the maximum number of common classes. We use transfer learning (Zoph et al., 2018; Pal & Paul, 2021), data augmentations, and various preprocessing techniques which help to improve a model's performance. This study mainly focuses on creating a pipeline for automating the count of WBCs with relatively less and imbalanced data and using domain adaptation which has not been explored much in this field of medical image segmentation. In summary, the principal contributions of this work are -

- We have created a high-resolution novel 10-class peripheral blood smear dataset, the RV-PBS dataset. The dataset can be used, for instance-

segmentation benchmarks. To the authors' knowledge, there is no such instance segmentation dataset that is available for research that has 10-class White Blood Cells (WBCs) and multiple different cells per slide.

- Employing Mask R-CNN for cell image extraction from high-resolution RV-PBS dataset for passing it into the domain adaptation (Guan & Liu, 2022) pipeline. Extensively comparing the SOTA CNN models to get better feature representation than previously reported (Acevedo et al., 2019) on the PBC Cropped dataset. Unifying PBC Cropped and CRV-PBS dataset by using a custom-made non-deep-learning segmentation algorithm.

- Performing domain adaptation using the best selected common transfer learning network backbone on the cropped images of both the dataset, i.e., CRV-PBS and C-PBC datasets, and improving the existing metrics, i.e., precision, recall, and F1-score of classification on the newly created dataset, making it suitable for benchmarking and deployment.

The overall pipeline is discussed in more detail in Figure 23 of section 4.3.1 of this paper. We structured this paper, beginning with an overview of the related works, followed by a description of the newly constructed dataset and its properties. We extensively investigated a similar classification dataset that helped us to get insights into making a more robust classification pipeline via transfer learning. The procedure for extracting the cell images to create a preprocessed dataset has also been discussed briefly. Next, we analyzed the application of Mask R-CNN in extracting cell images from the RV-PBS dataset, followed by domain adaptation to improve existing classification accuracy. Finally, we conclude with future scope and ideas for further progress to convert this research into production software that could cater to hematologists of different demography.

2. Related work

In the current literature, a wide variety of models can perform medical image segmentation Pal & Mj (2023) holistically. One such architecture is U-Net

(Ronneberger et al., 2015), which yields SOTA segmentation results in some datasets and is widely used for benchmarking. This architecture comprises a contracting path to capture context and a symmetric expanding path that enables localization. Researchers have proposed several variants of U-Net (Ronneberger et al., 2015; Thomas et al., 2021; Zhou et al., 2018; Ibtehaz & Rahman, 2020) in the current literature that delivers improved performance. The UNet++ (Zhou et al., 2018) architecture has nested and dense skip connections, potentially making it more capable of capturing fine-grained details by gradually enriching the high-resolution feature maps from the encoder network and then fusing them with the feature maps from the decoder network. Researchers have also implemented an attention gate in a standard U-Net architecture (Schlemper et al., 2019; Thomas et al., 2021) that allows the attention coefficient to be more specific to local regions. Recent studies, exemplified by the Vision Transformer (ViT) Dosovitskiy et al. (2021), demonstrate that a reliance on convolution is not essential. Utilizing a pure transformer-basedVaswani et al. (2017) architecture, originally designed for natural language processing tasks when applied directly to a sequence of image patches, proves highly effective for image classification.

Some research works have already attempted to address the problem of WBC differential based on images using Machine Learning (ML) methods (Wang et al., 2019; Deshpande et al., 2022, 2021). For example, they used object detection techniques, namely Single Shot Multi-box Detector (SSD) (Liu et al., 2016) and You Only Look Once (YOLOv3) (Redmon et al., 2016), for 11 types of peripheral leukocyte recognition. A few of the other well-known deep learning-based object detection architectures include Faster R-CNN (Ren et al., 2015), Mask R-CNN (He et al., 2017), Region-based Fully Convolutional Networks (R-FCN) (Dai et al., 2016) which have been used widely for detection purposes in computer vision. Researchers have proposed models to solve classification using unsupervised domain adaptation (Pandey et al., 2020) when cell images are captured from different lighting and camera conditions. Some scholars have also worked on self-supervised learning techniques (Zheng et al., 2018) to extract WBCs from slides via standard k-means and Support Vector Machine (SVM) methods, which

lack precision segmentation masks. A popular dataset containing about 40K cell slide images, known as Raabin dataset (Kouzehkanan et al., 2022), has only bounding boxes of about 5 classes of WBCs in them. The dataset also has segmentation masks of about 1145 cells for the nucleus and cytoplasm, which can be used for the early detection of haematologic diseases. Researchers have used a novel 1K dataset having only point annotation of cells for detection. This is done by a U-Net-like architecture called as DCNet (Lee et al., 2021) (Differential Count Network), which detects the cells. However, these methods lack accurate instance segmentation and detection of cells. The works in the current literature only detect a limited number of cell types for minimal purposes. Our work can be used in the medical sector for automating the blood testing procedure and diagnosing diseases by counting the major types of white blood cells. Our dataset has 10 classes, proper segmentation masks, and class annotations for all the cells in the slides. We have used Mask R-CNN (He et al., 2017) in one of our pipelines for extracting the cells from the RV-PBS dataset.

There have been studies in Unsupervised domain adaptation by backpropagation (Ganin & Lempitsky, 2015), which utilizes labeled data to learn features from some unlabelled data of similar domains to get domain invariant features. We use this method for our domain adaptation pipeline. Recent studies leverage the power of adversarial networks, which outperform models on standard datasets using data from different domains. Domain adaptation can also be applied in image segmentation (Murez et al., 2018) problems, but one dataset needs to have a large number of proper segmentation masks. Researchers have performed detection (Hsu et al., 2019) by training models via domain adaptation. A few works in medical sectors (Guan & Liu, 2022) use domain adaptation. Domain adaptation using generative (Al-qudah & Suen, 2020) latent search can help to classify cell types. Scientists have also studied different techniques using standard deep learning (Pandey et al., 2020) architectures, which help to detect cell types. But all these methods use a large amount of data, and their data has very little class imbalance, a very less variation in the number of different classes of the cells. This study explicitly handles the class imbalance and data

sufficiency problem via domain adaptation.

WBC class name	RV-PBS distribution	CRV-PBS distribution
Band Cell	60	63
Basophil	26	28
Blast Cell	68	153
Eosinophils	67	93
Lymphocytes	60	62
Myelocytes	83	95
Metamyelocytes	31	32
Monocytes	53	54
Neutrophil	99	116
Promyelocytes	36	56
Mixed for testing	144	NA

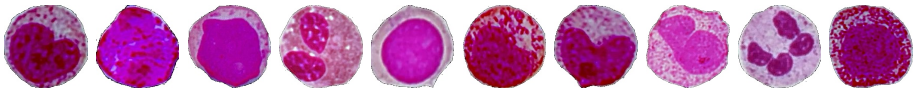
Table 1: Distribution of individual classes of WBCs from RV-PBS dataset with 727 slide images. All the classes except the mixed are annotated using CVAT. The corresponding total cell count from each class is the distribution of the CRV-PBS dataset.

3. Datasets

3.1. RV-PBS dataset

For this study, we have created a novel WBC dataset comprising 10 classes known as the Ramakrishna Vivekananda peripheral blood smear (RV-PBS) dataset. Air-dried peripheral blood smears are stained by Leishman stain following standard protocol and examined under an oil immersion lens using 10X eyepiece magnification (final magnification–1000X) — photographs taken by iPhone XR 12-megapixel camera with f/1.8 aperture. The dataset comprises high-resolution (4032 x 3024) images of blood smear slides. The cell images of the RV-PBS dataset were annotated using Computer Vision Annotation Tool (CVAT) (Sekachev et al., 2020) as shown in Figure 3. We show a few samples of the same in Figure 1. From Figure 1a, we find that there are several artifacts (of

the same purple color as the WBCs) present on the slide. While creating instance segmentation masks, we have not included these artifacts. These artifacts are chemicals released by individual WBCs.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 4: Images of individual samples from the CRV-PBS dataset. There are 10 classes, From top left: (a) Band Cell, (b) Basophil, (c) Blast Cell, (d) Eosinophils, (e) Lymphocytes, (f) Myelocytes, (g) Metamyelocytes, (h) Monocytes, (i) Neutrophil, and (j) Promyelocytes. These samples will be used to do domain adaptation.

Table 1 shows the distribution of individual classes for this dataset under the column titled RV-PBS distribution. From the table, we can see that the Basophil class has 26 slides and 28 instances, whereas Neutrophil has 99 slides and 116 instances. All the slides in the training dataset have the same class of cells per slide. The mixed slides have different classes of cell types per slide. We used mixed classes for testing only. All the instances of cells from this dataset except the mixed class are annotated. This is a developing dataset and we are planning to add more annotated samples to this dataset in the future.

3.2. CRV-PBS dataset

In the interest of domain adaptation, we extracted individual cells from the proposed RV-PBS dataset to create this classification dataset, the cropped RV-PBS (CRV-PBS) dataset. Figure 4 shows the samples of individual classes of the dataset. The distribution of this dataset is present in Table 1 under the column titled CRV-PBS distribution. Note that the distribution of slides varies from the individual cell count. This is because there might be many cells present on one slide. Since the test mixed class is not annotated, hence individual cropped cells for each class are not available (NA) as shown in Table 1. This dataset has a total of 762 cell images. The highest count is blast cell with 153 images, and the lowest count is the class of basophil with 28 images.

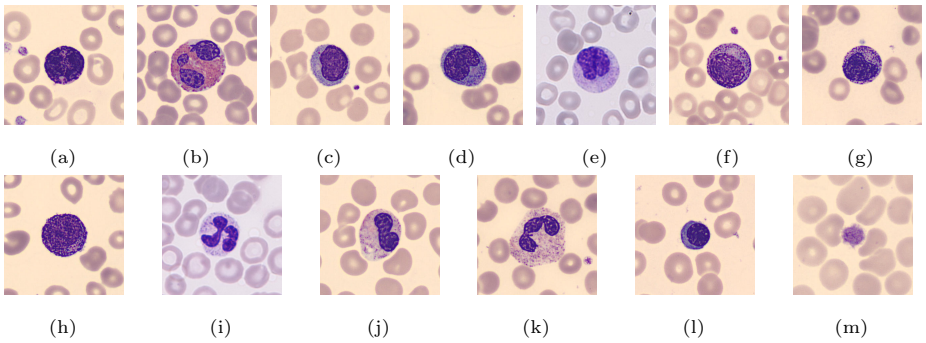


Figure 5: Images of samples taken from the PBC dataset. From top left to bottom right: (a) **Basophil** (BA), (b) **Eosinophil** (EO), (c) **Lymphocyte** (LY), (d) **Monocyte** (MO), (e) **Immature Granulocytes** (IG), (f) Myelocyte (MY), (g) Metamyelocyte (MMY), (h) Promyelocytes (PMY), (i) **Neutrophil** (NEUTROPHIL), (j) Band Neutrophils (BNE), (k) Segmented Neutrophils (SNE), (l) **Erythroblast** (ERB), (m) **Platelet** (PLATELET).

3.3. PBC dataset

This dataset (Acevedo et al., 2020) is retrieved from Mendeley [2] . There are 17092 images of normal cells which are gained using analyzer CellaVision DM96 camera from the Core Laboratory at the Hospital Clinic of Barcelona. These images are of 360x363 pixel resolution and are present in JPG format. Each of the individual slides contains only one cell as shown in Figure 5.

This dataset contains 8 classes. Some classes have subclasses, as shown in Table 2. From Table 2, we can see that the Immature Granulocyte class has 4 subclasses whereas Neutrophil has 3 subclasses. This labeled dataset is used to benchmark machine learning algorithms that are used for classification. To the authors of the dataset's knowledge (Acevedo et al., 2019), this is a first-of-its-kind canonical dataset for model benchmarking. We have used a standard convolutional backbone and reported better results than the results previously reported by the original authors.

[2] `https://data.mendeley.com/datasets/snkd93bnjr/1`

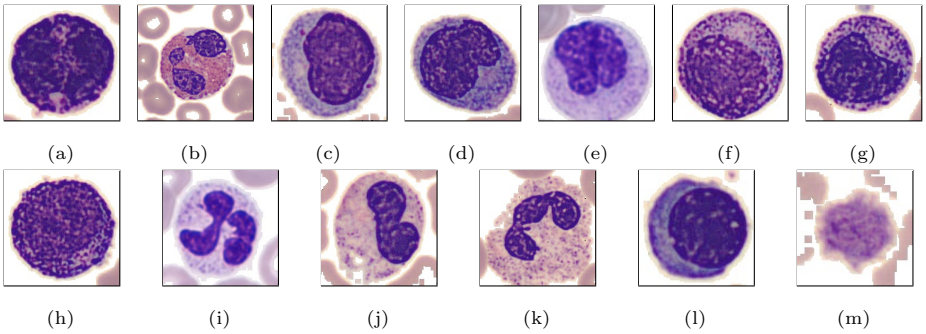


Figure 6: Images of samples taken from the C-PBC dataset. From top left to bottom right: (a) **Basophil** (BA), (b) **Eosinophil** (EO), (c) **Lymphocyte** (LY), (d) **Monocyte** (MO), (e) **Immature Granulocytes** (IG), (f) Myelocyte (MY), (g) Metamyelocyte (MMY), (h) Promyelocytes (PMY), (i) **Neutrophil** (NEUTROPHIL), (j) Band Neutrophils (BNE), (k) Segmented Neutrophils (SNE), (l) **Erythroblast** (ERB), (m) **Platelet** (PLATELET). The classes in bold show the superclass, which may have individual subclasses of cell types.

Class name	Subclass name	Total image count
Basophil (BA)		**1218**
Eosinophil (EO)		**3117**
Lymphocyte (LY)		**1214**
Monocyte (MO)		**1420**
Immature Granulocytes (IG)		**2895**
	Immature Granulocytes (IG)	151
	Myelocyte (MY)	1137
	Metamyelocyte (MMY)	1015
	Promyelocytes (PMY)	592
Neutrophil (NEUTROPHIL)		**3329**
	Neutrophil (NEUTROPHIL)	50
	Band Neutrophils (BNE)	1633
	Segmented Neutrophils (SNE)	1646
Erythroblast (ERB)		**1551**
Platelet (PLATELET)		**2348**

Table 2: Distribution of individual classes of PBC and C-PBC dataset with 17092 images.

(a) (b) (c) (d) (e) (f) (g)

Figure 7: The process of automating the cropping of the PBC dataset. From top left to bottom: (a) A sample image from the PBC dataset, (b) the image is converted from BGR to HSV, (c) mask obtained after applying the lower and upper threshold of blue color, (d) the dilated mask to make it more prominent, (e) *bitwise AND* operation with the mask and the original image, (f) Contours obtained by applying the watershed algorithm on the mask, (g) The cropped out segmented image.

3.4. C-PBC dataset

We are cropping out only the relevant cell samples from the RV-PBS dataset to identify WBC samples to create the cropped PBC (C-PBC) dataset. This will help the domain adaptation pipeline to concentrate on only the essential information from an image, i.e., the WBCs, and not the textured background or RBCs. We do this by taking the common image classes from both datasets to train the domain adaptation pipeline. This motivated us to create a cropped version of the PBC dataset, the C-PBC dataset. Since each slide has only one image present, this dataset has the same distribution of classes as the previous dataset. Samples from this dataset are shown in Figure 6. We use a simple algorithm to extract images from the PBC dataset.

The algorithm selects a standard image from the PBC dataset as shown in Figure 7a. It then converts this image from BGR (Blue Green Red) to HSV (Hue Saturation Value) [3] color scale, as shown in Figure 7b. We handpicked an average threshold of lower and upper color values that will mask out the cell in an unsupervised manner. The lower blue has an HSV value of (100, 20, 20), and the upper blue has (300, 245, 245). We mask the HSV image with this threshold, and the result of this process is shown in Figure 7c. Since there might be slight noise, we smoothen the mask by using a box kernel of size 3x3 for three

[3] `https://en.wikipedia.org/wiki/HSL_and_HSV`

iterations, as shown in Figure 7d. After obtaining the mask, we perform bitwise AND operation with the mask to remove irrelevant background and RBCs as much as possible, as shown in Figure 7e. To extract the individual contours of cells, we compute the exact Euclidean distance from every binary pixel to the nearest zero pixels. We select peaks in this distance mask with a minimum distance of 20. A connected component analysis on the local peaks is used to select markers [4]. We apply the watershed algorithm (Beucher, 1994) using the peaks, markers, and masks and draw circles as shown in Figure 7f. This is done since we don't have any proper mask for extracting the cells from this dataset. The extraction of cells is done in an unsupervised manner using classical and traditional techniques since this is a classification dataset, and we need to make this similar to the other dataset that will be used in the domain adaptation module. We noted a pattern in the PBC dataset that allows us to crop the most prominent circle from the center of the image to extract individual cells. The final image is shown in Figure 7g. We repeat this for every other image of the PBC dataset to get the C-PBC dataset. In this extraction process, we considered two assumptions:

- The cells are present in the exact center of the PBC slide.
- The size of the biggest circle detected by the algorithm is the size of the diameter of the cell. We use this information to crop out relevant squares from the PBC dataset.

It took **0.1448 seconds** to process an image on average using the algorithm. Efforts were made to remove some artifacts, such as RBCs, but some images, such as the cell shown in Figure 6b, have traces of artifacts.

[4] `https://pyimagesearch.com/2015/11/02/watershed-opencv/`

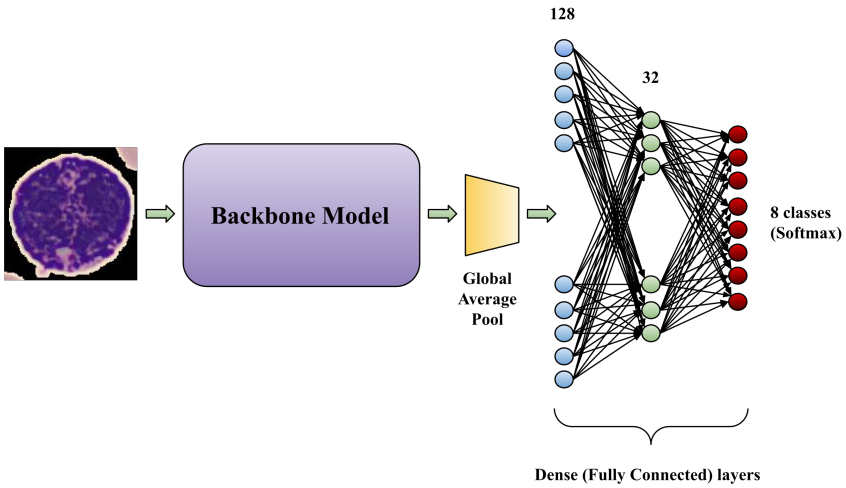


Figure 8: The deep learning architecture is used in building the classification models to compare popular architecture's convolutional neural network backbones. We use a global average pooling layer after extracting features from the CNN layer. The last layer has Softmax as activation, containing the different number of classes as the number of output neurons. In this model, we have 8 classes as activation outputs.

4. Methodology

4.1. Transfer learning

We apply transfer learning to all the classification datasets that were used in this study, i.e., PBC, C-PBC, and CRV-PBS datasets. We have used a simple architecture to compare different popular architectures, as shown in Figure 8. The architecture uses the transfer learning convolutional neural network backbone, which is replaced with different popular architectures. We needed to do an evaluative study since some architecture backbone might be good for a particular dataset (Pal & Paul, 2021) but not good for another. This is only found out when we actually apply the model and do an extensive benchmarking test on the datasets (Pal & Paul, 2021). The transfer learning backbone architecture is followed by a global average pooling layer and a set of fully connected layers. Input images to the model are of size 360x360x3, i.e., 3-channel RGB images. The input image is passed through a series of convolutional layers; for example,

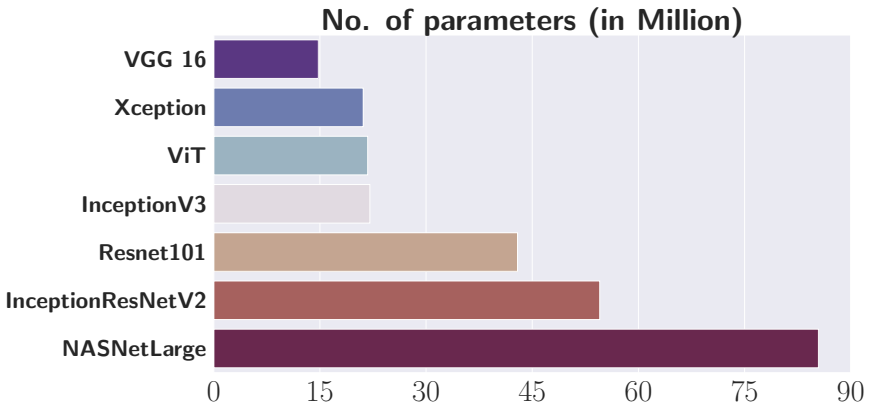
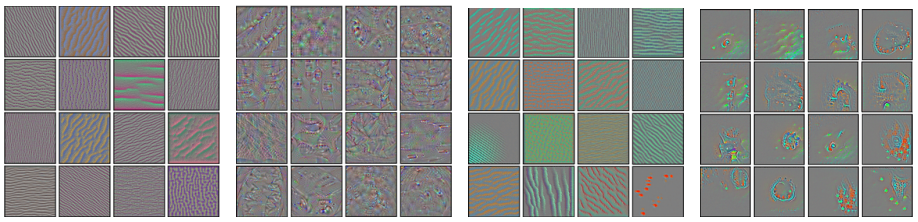


Figure 9: The number of parameters in million for different models. VGG16 with 14.78 M, Xception with 21.12 M, InceptionV3 with 22.06 M, Resnet101 with 42.92 M, InceptionResNetV2 with 54.53 M, and NASNetLarge with 85.43 M parameters.

in the case of the VGG16 model, when using Keras API, just before the Global Average Pooling, we get a feature volume of size 11x11x512. The Global average pooling layer in Keras takes the most prominent feature from each of these 512 channels creating a 512 pooled feature vector. This feature vector is fully connected with a layer containing 128 neurons. The output Softmax layer was tuned according to the number of classes present in the dataset. We use a set of three metrics, i.e., precision, recall, and F-1 score, to measure the performance of different models in different datasets. These metrics are discussed in Appendix 7. The parameters of the models are also shown in Figure 9. Batch size of 8, learning rate of 1e-04, and other hyperparameters were kept constant in this study, to ensure proper evaluation of different models.

We incorporated the Vision Transformer (ViT) Dosovitskiy et al. (2021) architecture, distinguished by its departure from traditional convolutional layers. Despite this divergence, it seamlessly integrates into our training framework for comparative studies. ViT underwent training akin to conventional convolutional architectures, with the penultimate layer featuring an equivalent number of fully connected/dense layers. During training, we initially froze the entire architecture except the penultimate layer, followed by full



(a) (b) (c) (d)

Figure 10: An illustration showing the patterns that the selected layers of the VGG16 model learn. Figure 10a and 10c is **block2_conv2**, one of the initial layers and Figures 10b and 10d is **block5_conv3**, one of the final layers. Figures 10a and 10b show the freezed version of the weights, i.e., the model trained on only the ImageNet dataset capturing the natural imagery's features, and the Figures 10c and 10d show the fully trained version of the weights, i.e., the features that are changed to identify only cell images, hence restricting to a restricted and specialized problem.

training in a subsequent phase. To integrate ViT, we adapted the openly available implementation from Keras (`https://keras.io/examples/vision/image_classification_with_vision_transformer/`), making specific modifications. A patch size of 6, as prescribed by the Keras implementation, and a projection dimension of 64 were utilized as hyperparameters for tuning the transformer architecture. To maintain consistency, we retained the same batch size and other hyperparameters for reusability.

The transformer architecture demonstrates precision rates of approximately 95.75% on the C-PBC test dataset and 92.67% on the PBC test dataset as shown in Table 3. While slightly lower than state-of-the-art architectures, achieving 99.02% precision on the PBC datasets, this variance is attributed to resource constraints for fine-tuning transformers. Transformer architectures demand substantial computing resources, and our results reflect the best performance achievable under these limitations. Transformers are also sensitive to learning rates, and after an extensive grid search, the reported precision corresponds to the optimal learning rate identified. However, the transformer's performance was suboptimal for the CRV-PBS and PBC datasets. Consequently, we employed the Xception model for Domain Adaptation, which outperformed across all metrics

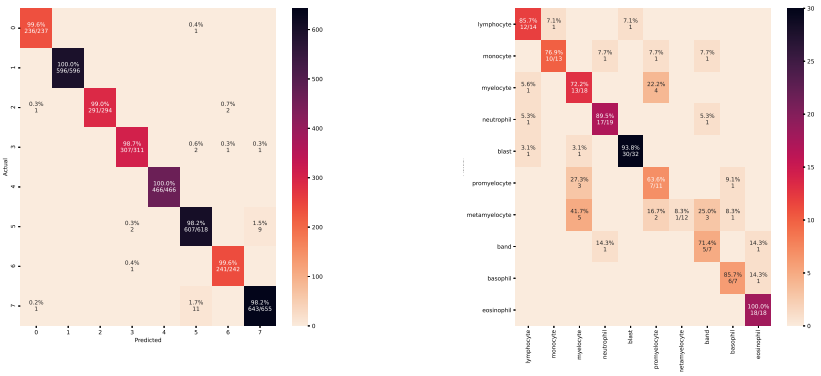
and datasets.

Model Name	Dataset	Type	Precision	Recall	F1-score
Inception V3 (Szegedy et al., 2016)	C-PBC	fully trained	96.01	95.74	95.70
	C-PBC	freezed	91.15	91.15	90.97
	PBC	fully trained	98.73	98.88	98.89
	PBC	freezed	91.61	91.42	91.28
Inception-ResNetV2 (Szegedy et al., 2016)	C-PBC	fully trained	98.84	98.99	98.82
	C-PBC	freezed	93.73	93.90	93.66
	PBC	fully trained	**99.02**	**99.07**	**98.93**
	PBC	freezed	93.60	93.41	93.45
NASNetLarge (Zoph et al., 2018)	C-PBC	fully trained	96.63	96.51	96.64
	C-PBC	freezed	92.51	92.38	92.39
	PBC	fully trained	98.84	98.92	98.75
	PBC	freezed	93.63	93.67	93.78
VGG16 (Liu & Deng, 2015)	C-PBC	fully trained	98.49	98.29	98.22
	C-PBC	freezed	95.03	94.85	95.05
	PBC	fully trained	98.73	98.78	98.69
	PBC	freezed	93.60	93.23	93.27
Xception (Chollet, 2017)	C-PBC	fully trained	98.96	98.81	98.75
	C-PBC	freezed	93.59	93.46	93.29
	PBC	fully trained	98.84	98.81	98.75
	PBC	freezed	91.51	91.19	91.27
Resnet101 (He et al., 2016)	C-PBC	fully trained	98.37	98.56	98.40
	C-PBC	freezed	71.05	68.95	68.47
	PBC	fully trained	98.84	98.92	98.75
	PBC	freezed	68.87	70.22	68.03
ViT (Dosovitskiy et al., 2021)	C-PBC	fully trained	95.75	94.31	93.16
	C-PBC	freezed	87.45	86.17	85.39
	PBC	fully trained	92.67	92.76	92.51
	PBC	freezed	83.21	82.17	82.10

Table 3: Application of various state-of-the-art models on PBC and C-PBC dataset with Adam Optimizer, a batch size of 16, and a learning rate of 1e-4 on the test set. Here, only NASNetLarge has a batch size of 8 because of limited computational resources. The results are shown in %-age.

Model Name	Precision (%)	Recall (%)	F1-score (%)
Inception V3	77.41	75.57	74.44
Inception-ResNetV2	81.07	76.91	74.93
NASNetLarge	61.79	54.37	51.29
VGG16	73.54	72.72	72.62
Xception	**82.15**	**78.80**	**76.96**
Resnet101	72.94	76.84	74.39
ViT	56.76	55.28	54.32

Table 4: Application of various state-of-the-art models on CRV-PBS dataset with Adam Optimizer and batch size of 16 and learning rate of 1e-4 on the test set. Here, only NASNetLarge has a batch size of 8 because of computational limitations.



(a) PBC dataset using InceptionResNetV2 model.

(b) CRV-PBS using Xception model.

Figure 11: Plots for the confusion matrix got from (a) The PBC dataset with 8 classes and (b) The CRV-PBS dataset. The corresponding labels for the PBC dataset are - 0 (Eosinophil), 1 (Neutrophil), 2 (Monocyte), 3 (IG - Inactive Granulocytes), 4 (Basophil), 5 (Erythroblast), 6 (Platelet), 7 (Lymphocyte).

4.1.1. Experiments

According to the original paper on the PBC dataset (Acevedo et al., 2019), the researchers used about 80% of the whole dataset in training and the rest 20% in testing. Among the 80% of the training set, 20% were used as a validation set. So, they used a 64-16-20 split of the whole dataset as a training-validation-test set. We have kept this ratio the same for a fair evaluation of standard architectures and found some other backbone to surpass their VGG-16 and Inception-V3 baselines (Acevedo et al., 2019). We have also used a seed of 42 using Python-3's Numpy's random function to reproduce the dataset split in every scenario reasonably. To keep things simple, we did not use data augmentation like the original authors of the dataset. Adam Optimizer (Kingma & Ba, 2015) with a learning rate of 1e-04 and a batch size of 8 was used for training the datasets. We perform a similar set of experiments on the CRV-PBS dataset, but here with only fine-tuned training. We train the entire network to find the best model to classify the 151 test images. The dataset has 762 images, and we split the dataset in train-validation-test similarly, i.e., 64-16-20.

The evaluation results on the test set for PBC and C-PBC datasets are shown in Table 3. We have not used the Accuracy metric in our benchmarking results since it might mislead with high scores (Johnson & Khoshgoftaar, 2019) and incorrectly show excellent performance. This is because our dataset is highly imbalanced, and hence the accuracy metrics will not give meaningful results. The training datasets, i.e., CRV-PBS and C-PBC datasets, have been augmented to the same number to alleviate the class imbalance problem. We use a combination of different training procedures for comparing the metrics. The Freezed version of the training uses the pre-trained backbone model as it is and only updates the weights and biases of the fully connected layer. A fully trained mechanism trains the complete model, i.e., the pre-trained backbone and the fully connected layer. This comparison is necessary since we wanted to show that even the features captured by the weights of the ImageNet (Deng et al., 2009) dataset can successfully classify cell images by tweaking the fully connected layer's weight

and biases.

To get an intuition of what the model learns, we have shown the weights [5] that the model learns to respond to in the initial and final layers of the model as shown in Figure 10. The block2_conv2 (shallow layer of VGG16 model) and block5_conv3 (deeper layer of VGG16 model) are the pre-trained default layer names available from the standard Keras API, (available at `https://keras.io/api/applications/vgg/`). In Keras, a model.summary() after invoking the model will give the layer names accordingly. We have used the following methodology for visualizing the features learned by the model. A specific loss function, as proposed in (Simonyan et al., 2014), is used, which maximizes the value of a given filter in a convolutional layer. We use Stochastic Gradient Descent(SGD) optimizer to adjust the values of the input image (which is a 360×360×3 size gaussian noise) so as to maximize the activation values. This process is continued for 80 iterations which continuously maximizes the response to the input image by adding the gradient to represent those features that a filter learns to respond to. This analysis is necessary for us to check what kind of patterns the deep learning model learns to respond to and the change in the response when we go from pre-trained model to fine-tuned model. The initial layers of the freezed training capture patterns of natural imagery as shown in Figure 10a and 10b. The final layers capture more abstract (Zeiler & Fergus, 2014) features. Similarly, the initial and final layers capture more problem-specific features, as shown in Figure 10c and 10d, pertaining to cell images.

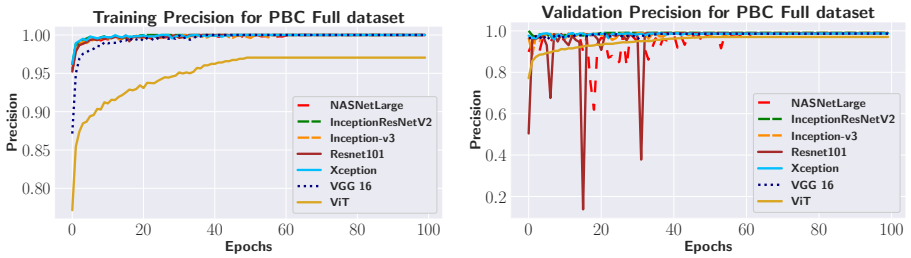
4.1.2. Results

Table 3 shows that InceptionResNetV2 (Szegedy et al., 2016) performs the best in terms of all the metrics in classifying the PBC dataset. The model achieves a precision of **99.02%**, recall of **99.07%**, and an F1-score of **98.93%** on the test dataset comprising 2609 images. The training and validation graph

[5] `https://keras.io/examples/vision/visualizing_what_convnets_learn/`

recorded for the model is shown in Figure 12. There are kinks in the plot of validation precision, which might happen when there are outliers in the data, which does not help in optimizing using mini-batch gradient descent [6]. Confusion matrix obtained on the test dataset is shown in Figure 11a. The major mistake the model makes is by predicting lymphocytes as erythroblasts. This is clear when we see Figure 5, since even normal humans might make the same mistake as the cells look similar.

From Table 3, we find the C-PBC dataset doing relatively badly in classifying cells in terms of precision, recall, and F1-score. This might be because of the neighboring information that is present in the cell, which is unnoticed by the normal human eye. Here, the Xception (Chollet, 2017) model performs the best with a precision of **98.96%**, recall of **98.81%** and an F1-score of **98.75%**. We note this information for further study.



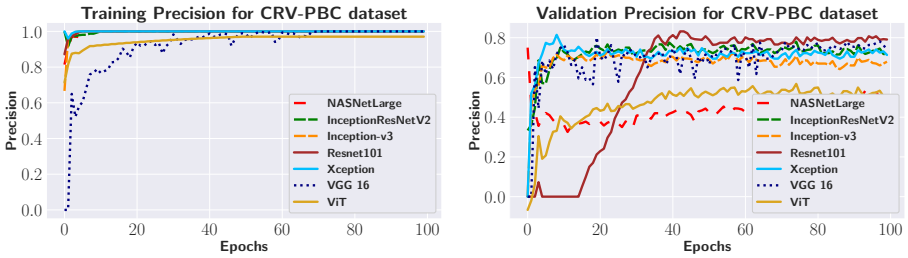
(a) Training

(b) Validation

Figure 12: Training and Validation graph for PBC full dataset. The models were trained for 100 epochs using the fine-tuned procedure, as discussed in Table 3.

The results of the experiments conducted on CRV-PBS datasets are shown in Table 4, and we find Xception model performs better than others. We assume more data can capture more features, which will make the metrics even higher. The Xception model has a precision of **82.15%**, a recall of **78.80%**, and an F1-score of **76.96%**. The confusion matrix of the Xception model on the test

[6] https://stats.stackexchange.com/questions/303857/explanation-of-spikes-in-training-loss-vs-iterations-with-adam-optimizer



(a) Training

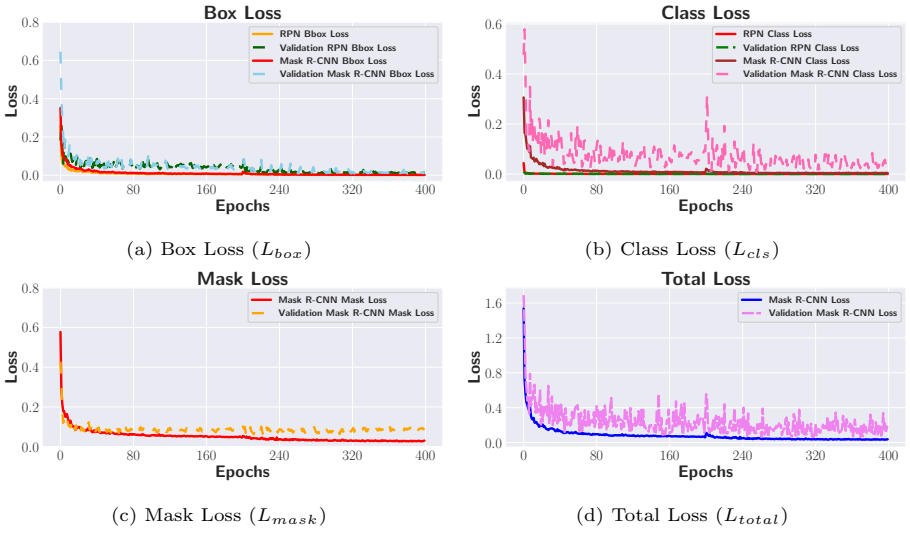
(b) Validation

Figure 13: Training and Validation graph for CRV-PBS dataset. The models were trained for 100 epochs using the fine-tuned procedure, as discussed in Table 4.

set is shown in Figure 11b. Training and validation loss using the fine-tuned procedure is shown in Figure 13. While the training graph for precision is smooth, the validation graph is turbulent. This is because of the unavailability of a huge amount of data and the class imbalance problem. We address this issue by performing unsupervised domain adaptation on the common classes of the C-PBC and CRV-PBS datasets to improve performance in classifying the data. We find an interesting observation about the Xception model, which performs better in classifying cropped images. We use this model as the backbone of the domain adaptation pipeline.

4.2. Mask R-CNN pipeline

For classifying images, we need to extract the cell images from the RV-PBS dataset. Mask R-CNN (He et al., 2017) can extract the cell images efficiently. Mask R-CNN outputs bounding boxes, classes present, and precise segmentation masks of different objects present in an input image. It uses a convolutional backbone, i.e., Feature Pyramid Network (FPN) for preserving features at different scales. Mask R-CNN is based upon Faster R-CNN (Ren et al., 2015) network which uses Region Proposal Networks (RPN). RPN uses bounding boxes also known as anchors to detect objects faster without searching the entire image. The major contribution of Mask R-CNN is it uses Region of Interest (ROI) Align (He et al., 2017) to align features at different scales using



(a) Box Loss (L_{box})

(b) Class Loss (L_{cls})

(c) Mask Loss (L_{mask})

(d) Total Loss (L_{total})

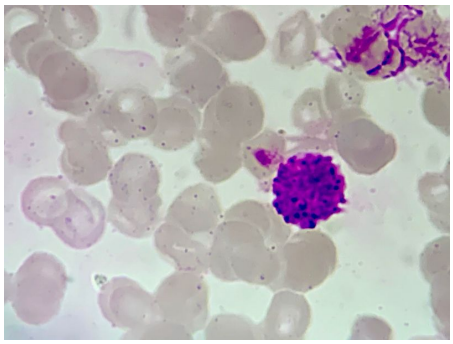
Figure 14: Training and Validation Losses of Mask R-CNN. The first 200 epochs are for training the network heads and the rest 200 epochs are for training the entire network of Mask R-CNN.

the bilinear interpolation method. This helps to remove location misalignment caused because of ROI pooling, hence, significantly increasing performance in getting segmentation masks.

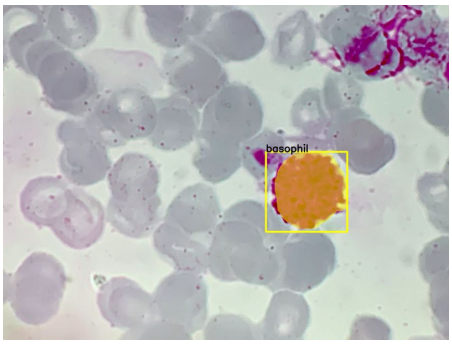
4.2.1. Experiments

We have used Matterport's implementation of Mask R-CNN (available at: `https://github.com/matterport/Mask_RCNN`). The architecture uses ResNet 101 backbone for getting the features, which is trained on the MS COCO (Microsoft Common Objects in Context) (Lin et al., 2014) dataset. The RV-PBS dataset containing 727 images was divided into 80-10-10, i.e., 80 training, 10 validation, and 10 test to perform training.

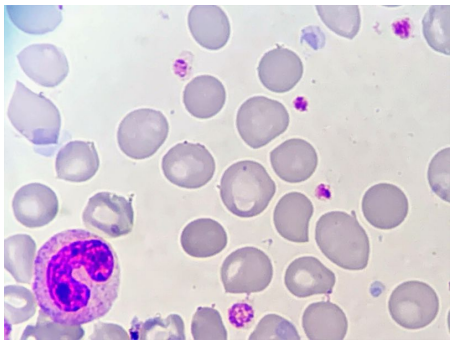
Training of Mask R-CNN pipeline was done using Quadro GV100 GPU with 32 GB VRAM. Here we discuss the initial configuration of the training pipeline which was provided by the package. The physical memory of the computer was 64 GB, hence, 3 images of the RV-PBS dataset of resolution 4032x3024 were passed to the GPU per step. The dataset has 10 classes and 1 background, hence



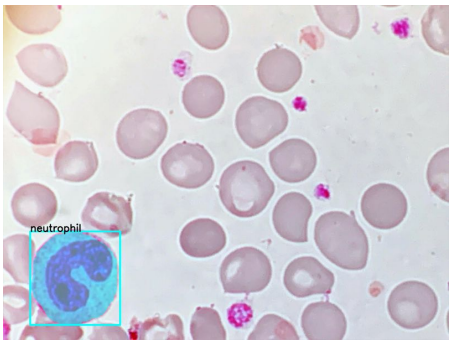
(a) Original image of Basophil cell from the test set.



(b) Mask R-CNN correctly detecting and segmenting the image as Basophil cell.



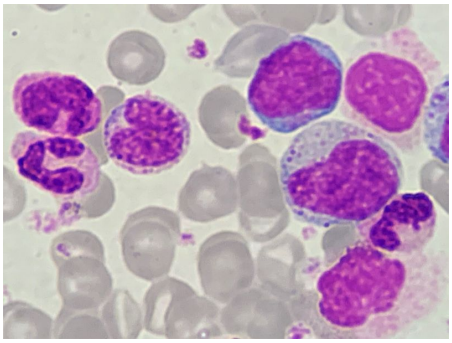
(c) Original image of Band cell from the test set.



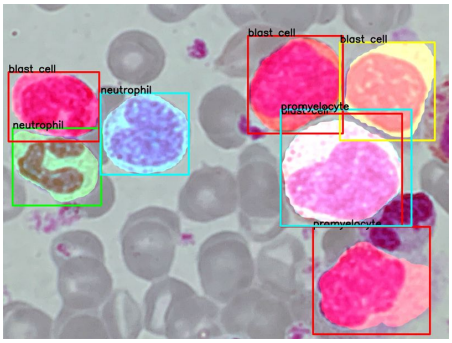
(d) Mask R-CNN correctly segmenting the cell, but incorrectly detecting the cell image as Neutrophil cell.

Figure 15: The outputs got from the Mask R-CNN pipeline for the test set of RV-PBS dataset, detecting and segmenting images.

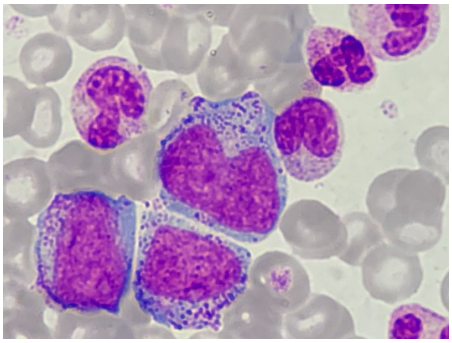
11 classes per pixel were detected. Steps per epoch were set to 500, whereas the validation step was set to 30. The confidence of detection was set to 0.7, which means if the Mask R-CNN model was confident above 70% that an instance of an object is present, only then it will detect it. For the first 200 epochs, only the Network heads were trained, i.e., the pre-trained convolutional blocks were not updating the weights, whereas the next 200 epochs fine-tuned the entire Mask R-CNN network. The pipeline used Stochastic Gradient Descent (SGD) (Robbins, 2007) optimizer with a learning rate of 0.001. The momentum of SGD was set to 0.9 during training.



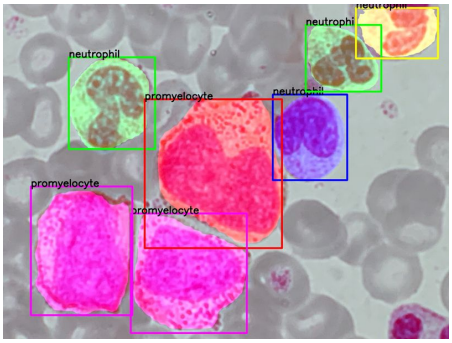
(a) Unannotated slide containing different cells.



(b) Mask R-CNN detecting and segmenting cells from the image.



(c) Unannotated slide containing different cells.



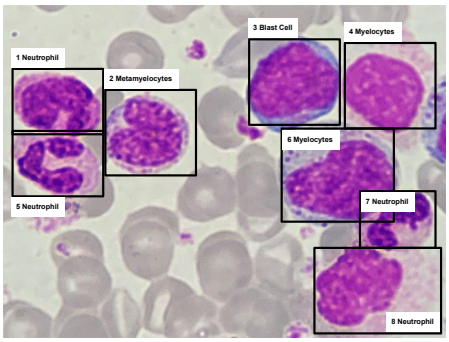
(d) Mask R-CNN detecting and segmenting cells from the image.

Figure 16: The outputs got from the Mask R-CNN pipeline for unannotated slides of the RV-PBS dataset.

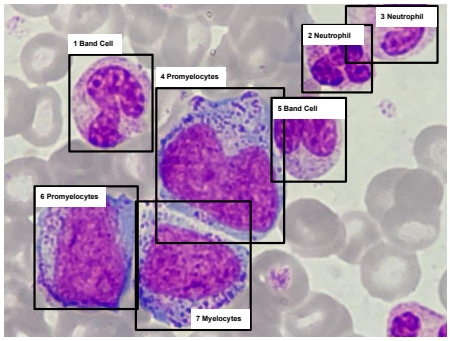
The loss of Mask R-CNN (He et al., 2017) can be formulated as:

$$L_{total} = L_{cls} + L_{box} + L_{mask} \tag{1}$$

The graph of Mask R-CNN during training and validation is shown in Figure 14. We show the Box Loss (L_{box}) in Figure 14a, the Class Loss (L_{cls}) in Figure 14b, the Mask Loss (L_{mask}) in Figure 14c and the Total Loss (L_{total}) in Figure 14d.



(a) Annotated version of Slide shown in Figure 16a.



(b) Annotated version of Slide shown in Figure 16c.

Figure 17: The annotated version of the unseen mixed slide of the RV-PBS dataset shown in Figure 16, is annotated for testing the effectiveness of Mask R-CNN for out-of-distribution images.

4.2.2. Results

Mask R-CNN shows promising results in masking the cells present in the RV-PBS dataset. We see some images are correctly detected, as shown in Figure 15b. Some are incorrectly detected, as shown in Figure 15d. This is because of the lack of a massive amount of data for training. We assume that training about 5000 images will give excellent results in detecting cells using Mask R-CNN only. Even though the training was conducted using about 580 images which contained at max 1-2 instances of cells per slide that too in the center of the images, Mask R-CNN successfully produced masks for this distribution of slides which had much more amount of cells different from the training distribution. From Figure 16b and 16d, we can see that the cells were completely masked except for one cell in Figure 16b. The pipeline correctly segregated different instances of the cells present in the slides of the RV-PBS dataset.

The purpose of Mask R-CNN was to correctly segregate different instances of cells for classification using a domain adaptation pipeline. Here, it successfully performs its job but the classification capability was not up to the mark as shown in Figure 16a, 17a, 16d and 16d. The annotated images for the RV-PBS dataset is shown in Figure 16c and Figure 17b. The results of the Mask R-CNN

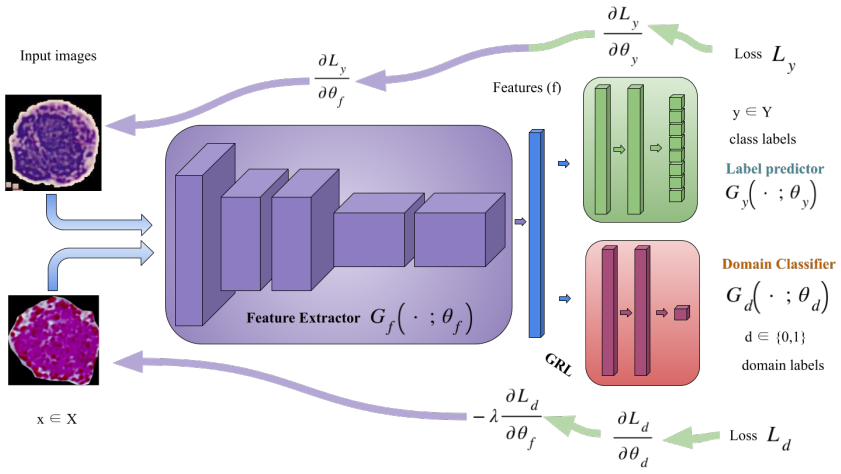


Figure 18: Unsupervised domain adaptation (Ganin & Lempitsky, 2015) pipeline suited to our problem. We use images from two domains, i.e., C-PBC (top left) and CRV-PBS datasets (bottom left) for training the domain adaptation model.

pipeline show the effectiveness of Mask R-CNN in segmenting different instances of cell images. If the training was done using more data, the classification would have been much better on even out-of-distribution images. This motivates us to develop the domain adaptation pipeline, which uses similar data of different domains to improve the existing classification accuracy of the classifiers.

4.3. Domain adaptation framework

It is becoming increasingly popular to re-train deep neural networks with pre-defined weights, such as ImageNet. Transfer Learning often uses pre-trained weights for the model, which was trained on a similar dataset, to fine-tune the deep neural network architecture. This way, the knowledge gathered previously through training will be used on a similar problem. Domain adaptation is a part of Transfer Learning, as shown in Figure 19.

4.3.1. Experiments

The domain adaptation pipeline is shown in Figure 18. The formulation of the domain adaptation model is discussed in Appendix 7. We use unsupervised

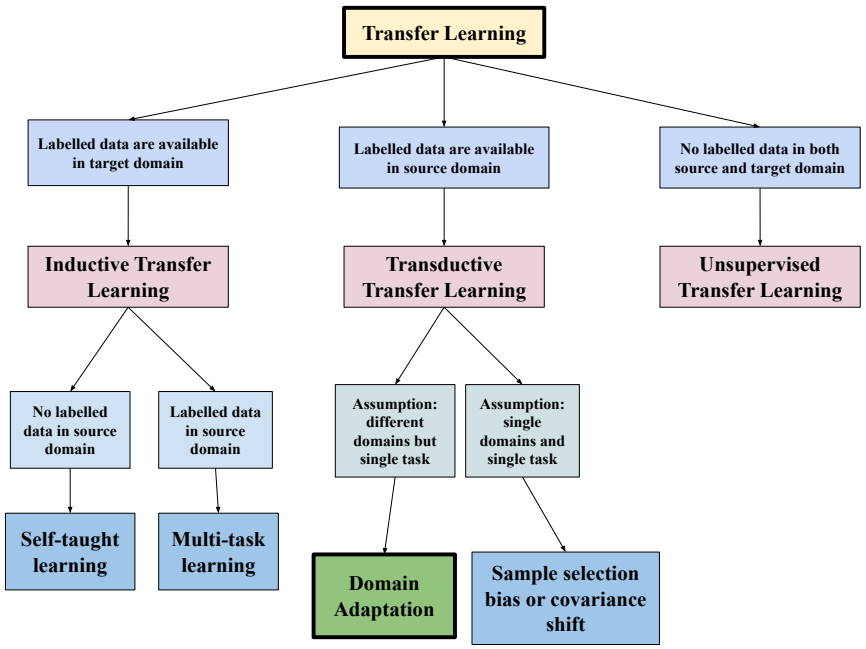
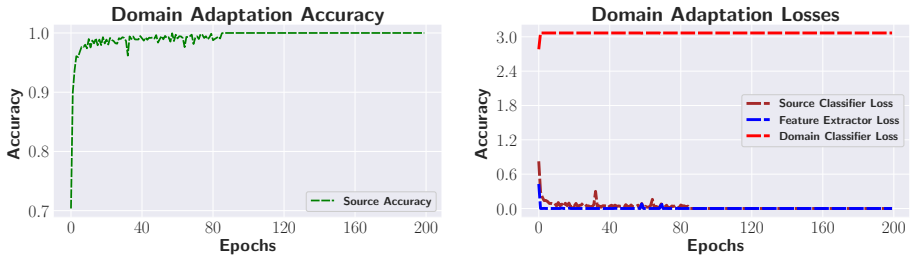


Figure 19: Flowchart showing the hierarchies of Transfer Learning (Pan & Yang, 2010). Domain adaptation is a subpart of Transfer Learning.

domain adaptation (Ganin & Lempitsky, 2015) for our purpose. The overall pipeline is shown in Figure 23. We initially used Mask R-CNN to extract the cells from the RV-PBS dataset along with getting the classes. The extracted cells are then passed to an initial classifier for screening. If the cells belong to the common classes, the cell image is passed through a domain adaptation module, which only uses a label classifier for this task. The Domain Classifier then outputs the final classification class, thus refining the classification of Mask R-CNN along with segmenting the classes.

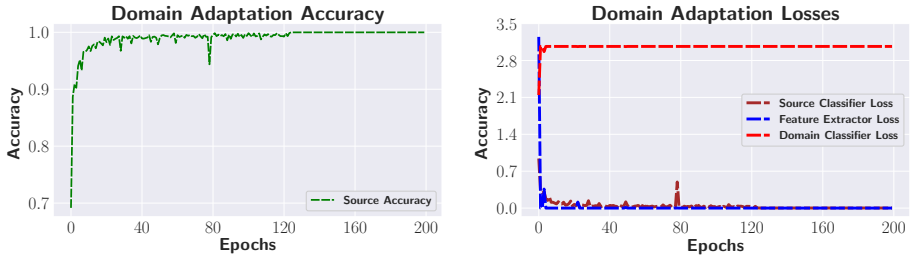
Since medical data prospect under heavy constraints (Willemink et al., 2020), the data has very little noise and variance. Hence, there is a very mere need for data augmentations, except for increasing the number of samples. While training the domain adaptation pipeline, we need to have a number of training samples in equal quantities from both domains. Hence, we apply a set of data augmentations to create the training dataset for domain adaptation. We use a probability vector



(a) Accuracy of Source Domain Images.

(b) Domain adaptation Losses.

Figure 20: Recorded Accuracy and Loss graph for domain adaptation with Source as RV-PBS and Target as C-PBC dataset. Losses are of Source Classifier (Categorical Crossentropy), Related Feature Extractor, and Domain Loss (Binary Crossentropy).

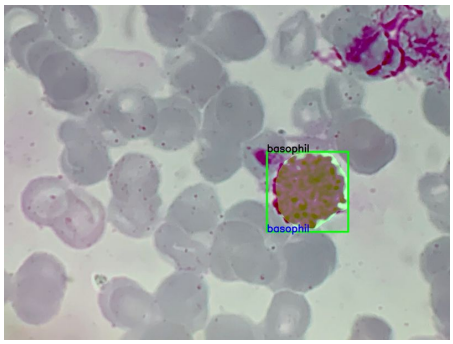


(a) Accuracy of Source Domain Images.

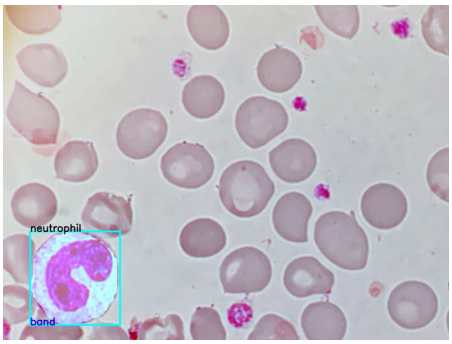
(b) Domain adaptation Losses.

Figure 21: Recorded Accuracy and Loss graph for domain adaptation with Source as C-PBC and Target as CRV-PBS dataset. Losses are of Source Classifier (Categorical Crossentropy), Related Feature Extractor, and Domain Loss (Binary Crossentropy).

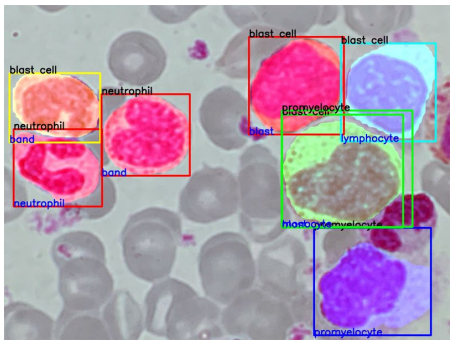
for performing a series of data augmentations. For this purpose, we choose an 11-length vector to decide whether a particular augmentation will happen, i.e., 1 or not, i.e., 0. This is just a simple design choice which can augment the dataset in 11 different ways at random. For example, a vector of (1,0,1,1,1,1,1,1,1,1,0) will decide that only the 2nd and last augmentation will not happen. We select this vector by giving an equal chance for each of the indices, i.e., while creating the vector, each cell might have a 0 or 1. There are various types of data augmentations that were applied to the CRV-PBS dataset. First, since the data of C-PBC had some background, we can give some random background from a series of backgrounds. We can set a zoom range between 0.5 to 1 to apply



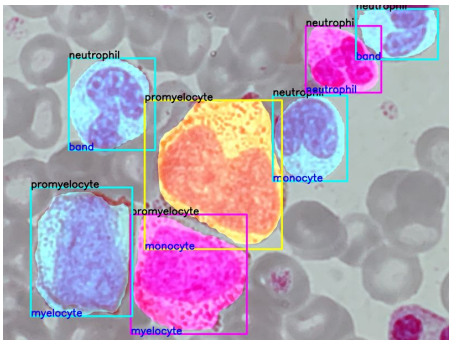
(a) Detected Basophil image by both Mask R-CNN and domain adaptation pipeline.



(b) Detected Neutrophil class by Mask R-CNN pipeline whereas Band Cell in domain adaptation pipeline. The original class is Band Cell.



(c) Unannotated images from RV-PBS dataset.



(d) Unannotated images from RV-PBS dataset.

Figure 22: The outputs got from Mask R-CNN and the domain adaptation pipeline combined. The detected class in black shows the Mask R-CNN output, whereas the detected class in blue shows the classification and domain adaptation pipeline's combined output.

to the image. We can flip the image horizontally, vertically, or both with a probability $\frac{1}{3}$. The camera captured a specific domain, and we applied a colour transformation to look more specific to that. Like for example, a fiery colour transform will make the slide look more yellowish, while a cool transform will make the slide look bluish, which was present in the stain. The colour transform is applied with a probability of $\frac{1}{6}$.

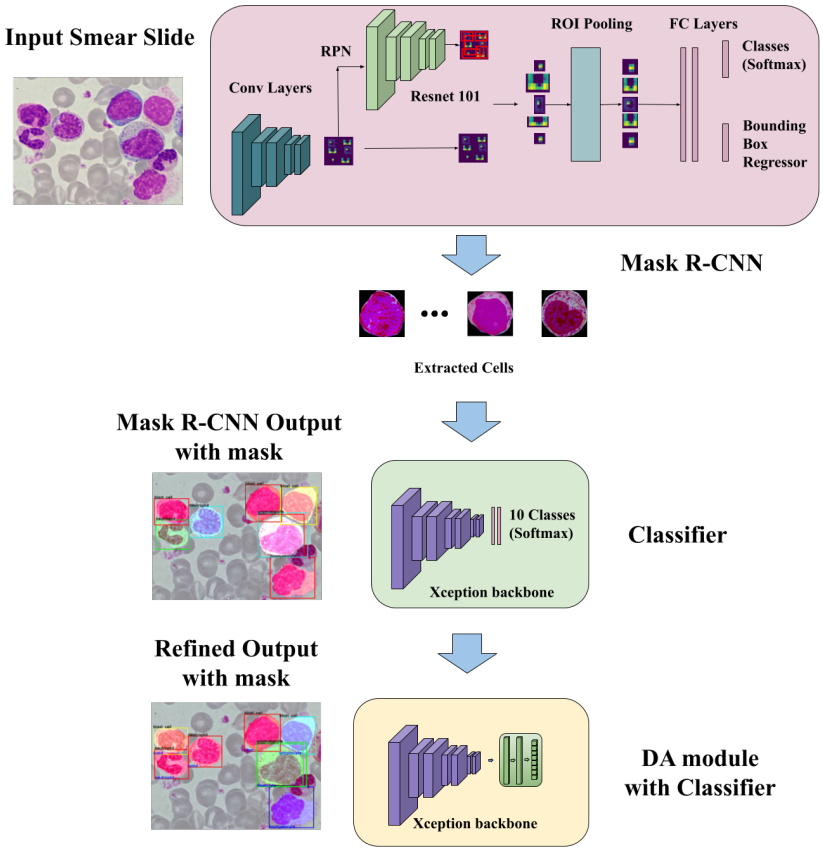


Figure 23: The overall pipeline of our work. We extract individual cells via Mask R-CNN from the RV-PBS dataset. First, we screen the classes by passing them through a classification model trained solely on the CRV-PBS dataset. If the detected image is in a common class, we re-verify the image using the domain adaptation pipeline for refining classification.

We can apply repeated Gaussian Blur to fade the image, which will simulate the out-of-focus for the cell on the camera. This is not so common in medical images, but we still wanted the algorithm to be robust to these perturbations. We changed the contrast and brightness to a certain extent with a certain probability of making different cell samples. We also rotated an image to a certain angle with a probability of $\frac{1}{4}$ in the range [0,90], [90,180], [180,270], and [270,360]. The common classes which were used for this study were: Basophil, Eosinophils, Lymphocytes, Myelocytes, Metamyelocytes, Monocytes, Neutrophil,

and Promyelocytes. These classes were present in both the CRV-PBS and the C-PBC datasets.

Remarks	**Precision (%)**	**Recall (%)**	**F1-score (%)**
Trained on the 8 common classes of C-PBC dataset	94.78	94.77	94.79
Above model when used to classify 8 common classes of CRV-PBS dataset	4.47	21.29	7.45
Trained on the 8 common classes of CRV-PBS dataset	82.64	82.25	81.74
Above model when used to classify common classes of C-PBC dataset	53.34	24.36	28.29
Model trained on mixed and classifying the 8 common classes of C-PBC dataset	95.21	95.45	95.37
Model trained on mixed and classifying on the 8 common classes of CRV-PBS dataset	83.98	83.73	84.34

Table 5: The eventual results summarized for the 8 Common Classes datasets and fine-tuning on the Xception model using a learning rate of 1e-04 and Adam Optimizer.

Remarks	Precision (%)	Recall (%)	F1-score (%)
Source: 8 common classes of C-PBC dataset **Target:** 8 common classes of CRV-PBS dataset Testing on 8 common classes of CRV-PBS dataset	78.89	78.31	77.11
Source: 8 common classes of C-PBC dataset **Target:** 8 common classes of CRV-PBS dataset Testing on 8 common classes of C-PBC dataset	95.64	95.77	95.89
Source: 8 common classes of CRV-PBS dataset **Target:** 8 common classes of C-PBC dataset Testing on 8 common classes of CRV-PBS dataset	86.71	85.67	84.89
Source: 8 common classes of CRV-PBS dataset **Target:** 8 common classes of C-PBC dataset Testing on 8 common classes of C-PBC dataset	93.11	93.21	93.37

Table 6: The eventual results by performing domain adaptation, summarized for the 8 Common Classes datasets and fine-tuning on Xception model using a learning rate of 1e-04 and Adam Optimizer.

4.3.2. Results

While testing domain adaptation, we need to train two times, i.e., using 8 common classes of the C-PBC dataset as the source and 8 common classes of the CRV-PBS dataset as the target and vice versa. In both cases, we need to find

the results of the model on the test set of both the dataset. First, we perform domain adaptation on the source containing 8 common classes of the C-PBC dataset and the target containing 8 common classes of the CRV-PBS dataset. When we evaluated the results on the testing set of the 8 common classes of the CRV-PBS dataset, we get precision, recall, and F1-score as **78.89%**, **78.31%** and **77.11%** respectively as shown in Table 6. When we test the same model on the 8 common classes of the C-PBC dataset, we see a slight increase in precision, recall, and F1-score for the 8 classes. The recorded values are **95.64%**, **95.77%** and **95.80%** respectively, as shown in Table 6. This is the best we can record for the 8 common classes combined. The training loss graph is shown in Figure 20. The loss shows the variation of losses of source classifier and feature extractor, which gets minimized as training progresses, and domain classifier, which gets maximized as training progresses. Similarly, the training accuracy captured by the source classifier during training is also shown in Figure 20. The test data received an accuracy of **96.32%** on 8 common classes of the C-PBC dataset whereas **82.34%** on 8 common classes of the CRV-PBS dataset.

When we perform domain adaptation using 8 common classes of the CRV-PBS dataset as the source and 8 common classes of C-PBC dataset as the target, we get the best-recorded results on the test dataset for the 8 common classes of the CRV-PBS dataset. The precision, recall and F1-score are **86.71%**, **85.67%** and **84.89%** respectively. The same model, when tested on 8 common classes of the C-PBC dataset, we get a precision, recall, and F1-score of **93.11%**, **93.21%** and **93.37%** respectively. Similar losses were recorded for this training in Figure 21. This shows domain adaptation does a good job of improving the classification of datasets containing common classes.

5. Dicsussion

To compare the classification accuracy of the different classifiers, we have performed a comparative study, as shown in Table 5. We trained the dataset on about 13050 images from the 8 common classes of the C-PBC dataset and

about 544 images from the 8 common classes of the CRV-PBS dataset without performing data augmentations. The split of the dataset was about 80% for training and 20% for testing. We trained the Xception model, which showed exceptional performance in both the datasets on 8 common classes of the C-PBC dataset, recording a precision of **94.78%**, recall of **94.77%** and F1-score of **94.79%**. Note that this is lesser than the metrics recorded when the model was trained on the C-PBC dataset, this is because of lesser data used in training the 8 classes of the same dataset. The model trained on 8 common classes of the C-PBC dataset when tested on 8 common classes of the CRV-PBS dataset gives lower results compared to doing vice versa. It seems like when a model is trained on 8 common classes of CRV-PBS data; it can classify the C-PBC dataset better than when the reverse is done. This may be because of the high-quality images of our dataset. For further study, we mix the data from both the datasets and perform an overall training which gives overall good results as shown in Table 5.

The results of the full pipeline are shown in Figure 22. We can see that the classification of the 8 common classes has improved using the domain adaptation pipeline. The detected class above the box in black color shows the initial class detected by the Mask R-CNN pipeline, whereas the detected class below the box in blue shows the result of the domain adaptation pipeline. We can see that the incorrect detection of Neutrophil is now correctly classified as a Band cell in Figure 22b. The case is unique in segmenting the mixed classes as shown in Figure 22c and Figure 22d. We see that a combination of both results will increase the performance in classifying the type of cell. In some worst cases, the domain adaptation pipeline might give wrong results when there is confusion about the labeling of the cells. We assume that training in more amount of data via Mask R-CNN will increase the segmentation and classification accuracy of the pipeline. This is novel research that builds a full pipeline from scratch to segment and detects all 10 classes of blood cell images according to Kolkata's demography. This work can be extended to count all the blood cell types by generating automated blood test reports with just minor modifications. The JSON data that is extracted from the model can get the count and types of cells

at a very reasonable speed, hence automating a major part of the healthcare industry.

5.1. Future scope

Because of the efficiency of Mask R-CNN in segmenting cells, we have developed a tool to point toward a folder containing several smear slides, and the tool will generate information regarding the type of cell and polygon masks of different cells present in the slide. In the future, we are planning to create an annotation tool that will leverage the masking capabilities of Mask R-CNN to automate the masking of newer slides. The annotator might have a minimum intervention to mask the out-of-distribution images. In this way, we can label a huge amount of slides in no time. The result of the JavaScript Object Notation (JSON) data generated is shown below:

```
{"data":
{"0":
{"filename":"IMG_4302.jpg", "height":3024,"width":4032,
"masks":
[
{"class_name":"neutrophil","score":0.9632735252,
"bounding_box": { "x1":22, "x2":740,
"y1":436,"y2":1169},
"vertices":[[397.0,1142.5], ... ,
[396.0,1141.5], [397.0,1142.5]]},


....


{"class_name":"neutrophil", "score":0.9132735252,
"bounding_box": {"x1":223, "x2":940,
"y1":936, "y2":1769},
"vertices":[[223.6,940.5], ... ,
[224.3,939.4], [223.6,940.5]]}
]
},
"1": {...}
...
}
}
```


A sample of JSON file generated using Mask R-CNN based tool

This data can be ported in any format. For our purpose, we have used CVAT annotation format in Extensible Markup Language (XML). In the future, we may create our own tool and it may have our own format for easy annotation. This is the future goal of this project, to create tools that will make the model learn new features actively with minimum human intervention.

5.2. Computational requirements and Complexities of the models

The analysis of model complexities in our study is meticulously presented in Table 7 and Table 8, considering a comprehensive set of statistical metrics. These metrics encompass the number of convolutional layers and fully connected (dense) layers, total, trainable, and non-trainable parameters (in millions), memory utilization (in Gigabytes), file size (in Megabytes), as well as average training and inference times indicated as mean $\pm$ standard deviation ($\mu \pm \sigma$ in seconds). It is noteworthy that all models underwent training on a singular NVIDIA GeForce RTX 3080 Ti GPU, implying potential variations in these statistics when utilizing alternative GPU configurations.

Table 7 presents the outcomes of our experiments (discussed in Sections 4.1.1 and 4.1), utilizing the PBC and C-PBC datasets to identify the most effective convolutional backbone for both datasets. The observations from Table 7 reveal that, as a rule, the convolutional backbone adopted from state-of-the-art (SOTA) models was chosen, with three dense layers following flattening, as discussed in Section 4.1. It's worth noting that the relationship between the number of parameters and training time is not strictly linear; while larger parameter counts tend to extend training durations, this dependency can be influenced by the internal connectivity patterns (Vento & Percannella, 2019) within the model. More intricate architectures may demand additional computational resources and training time, even when the parameter count is relatively lower than that of a simpler architecture. The number of trainable parameters, referring to the model's weights and biases requiring updates during training, plays a crucial role. Although the total parameter count remains constant, training time per epoch fluctuates based on the number of parameters undergoing updates. Generally, a frozen model variant consumes less time than a fine-tuned one that necessitates the entire model to be trained. In summary, the statistics in Table 1 highlight that models with more parameters tend to demand more time for both training and inference compared to models with fewer parameters.

In this paper, we describe the GPU memory requirement, specifically addressing the memory needed to process a single batch of data in gigabytes (GBs).

We have developed a dedicated function to estimate the memory consumption of deep learning models when handling data in batch mode. Our function is implemented using the TensorFlow Keras backend and is designed to analyze model complexities comprehensively. It calculates memory usage for each layer's output shape, recursively accounts for internal sub-models, and provides detailed statistics, including trainable and non-trainable parameter counts. Additionally, it determines the size of a single numerical element and calculates the total memory usage in bytes, converting it into gigabytes. This functionality serves as an indispensable tool for both researchers and practitioners, offering valuable insights into the memory requirements of deep learning models. It can be instrumental in making informed decisions regarding resource allocation and optimization strategies across various computational contexts. It is noted that more complex models require much more memory footprint than that of lower complexity models.

In our study, we extend our experiments to encompass the CRV-PBS dataset, presenting the resulting statistics in Table 8. These findings are similar to those elucidated in Table 7, as detailed in Sections 4.1.1 and 4.1. In the context of selecting an optimal backbone for our Domain Adaptation pipeline, elaborated in Section 4.3, we have chosen the Xception model. This choice stems from its superior performance in classifying both datasets. Notably, the Xception model stands out with a lower number of convolutional layers compared to most models, resulting in reduced inference and training times, particularly when applied to the CRV-PBS dataset. Our approach strikes a balance, emphasizing a trade-off between reduced inference time for most models while maintaining accuracy and precision in prediction. The experimental results underscore the effectiveness of our model in achieving this balance, yielding an optimized, lower-complexity model with faster inference and training times than most models, all while enhancing classification prediction accuracy and precision.

Model Name	Dataset Name	Training Type	Total # of Conv Layers	Total # of Dense Layers	Total Params (M)	Trainable Params (M)	Non-Trainable Params (M)	Memory Usage (GB)	File Size (MB)	Train Epoch time (in s)	Inference time (in s)
Inception V3	C-PBC	fully trained	94	3	22.0694	22.0350	0.0344	2.8340	253.6688	119.7687 ± 9.8956	0.0644 ± 0.0390
	C-PBC	freezed	94	3	22.0694	0.2667	21.8028	2.8340	87.0698	117.4597 ± 5.2568	0.0744 ± 0.0555
	PBC	fully trained	94	3	22.0694	22.0350	0.0344	2.8340	253.6688	148.1968 ± 41.0319	0.0773 ± 0.0516
	PBC	freezed	94	3	22.0694	0.2667	21.8028	2.8340	87.0698	159.4549 ± 38.7494	0.0867 ± 0.0963
Inception ResNetV2	C-PBC	fully trained	244	3	54.5379	54.4773	0.0605	7.0000	626.9535	215.2427 ± 7.4374	0.0712 ± 0.0636
	C-PBC	freezed	244	3	54.5379	0.2011	54.3367	7.0000	211.5637	162.4180 ± 10.2264	0.1121 ± 0.0828
	PBC	fully trained	244	3	54.5379	54.4773	0.0605	7.0000	626.9653	223.7845 ± 0.7555	0.0729 ± 0.0535
	PBC	freezed	244	3	54.5379	0.2011	54.3367	7.0000	211.5637	135.6346 ± 6.2723	0.0783 ± 0.0603
NASNetLarge	C-PBC	fully trained	268	3	85.4374	85.2408	0.1967	17.6370	981.3203	462.1109 ± 1.4508	0.0765 ± 0.0731
	C-PBC	freezed	268	3	85.4374	0.5206	84.9168	17.6370	332.6796	189.3978 ± 1.0356	0.0736 ± 0.0137
	PBC	fully trained	268	3	85.4374	85.2408	0.1967	17.6370	981.3203	459.2430 ± 1.4518	0.0889 ± 0.0256
	PBC	freezed	268	3	85.4374	0.5206	84.9168	17.6370	332.6796	218.1738 ± 3.1188	0.0798 ± 0.0536
VGG16	C-PBC	fully trained	13	3	14.7847	14.7847	0.0000	2.3970	169.3445	151.5882 ± 0.6383	0.0461 ± 0.0099
	C-PBC	freezed	13	3	14.7847	0.0701	14.7147	2.3970	57.0247	87.7355 ± 0.4555	0.0462 ± 0.0096
	PBC	fully trained	13	3	14.7847	14.7847	0.0000	2.3970	169.3445	150.3752 ± 0.8648	0.0461 ± 0.0099
	PBC	freezed	13	3	14.7847	0.0701	14.7147	2.3970	57.0247	87.5087 ± 2.4485	0.0464 ± 0.0106
Xception	C-PBC	fully trained	40	3	21.1281	21.0736	0.0545	5.5510	242.1414	140.1736 ± 0.2560	0.0478 ± 0.0201
	C-PBC	freezed	40	3	21.1281	0.2667	20.8615	5.5510	83.0389	91.1776 ± 0.4915	0.0476 ± 0.0201
	PBC	fully trained	40	3	21.1281	21.0736	0.0545	5.5510	242.1414	140.3787 ± 0.3783	0.0475 ± 0.0177
	PBC	freezed	40	3	21.1281	0.2667	20.8615	5.5510	83.0389	90.0230 ± 1.3435	0.0474 ± 0.0199
Resnet101	C-PBC	fully trained	104	3	42.9248	42.8195	0.1053	8.9790	492.3575	202.0621 ± 0.5078	0.0554 ± 0.0357
	C-PBC	freezed	104	3	42.9248	0.2667	42.6582	8.9790	166.7687	111.0408 ± 1.2891	0.0559 ± 0.0327
	PBC	fully trained	104	3	42.9248	42.8195	0.1053	8.9790	492.3575	202.2891 ± 0.8241	0.0569 ± 0.0338
	PBC	freezed	104	3	42.9248	0.2667	42.6582	8.9790	166.7687	110.6711 ± 0.3967	0.0546 ± 0.0320
ViT	C-PBC	fully trained	0	19	21.6575	21.6575	0.0000	10.3453	242.48	154.34 ± 0.4521	0.1232 ± 0.0467
	C-PBC	freezed	0	19	21.6575	0.2667	21.3908	10.3453	162.23	102.33 ± 1.2134	0.0829 ± 0.0324
	PBC	fully trained	0	19	21.6575	21.6575	0.0000	10.3453	242.48	154.34 ± 0.4521	0.1202 ± 0.0578
	PBC	freezed	0	19	21.6575	0.2667	21.3908	10.3453	162.23	102.33 ± 1.2134	0.0874 ± 0.0864

Table 7: The assessment of model complexities for all models trained on the PBC and C-PBC datasets involves the application of state-of-the-art models with a batch size of 16 and a learning rate set at 1e-04 for evaluation on their respective datasets. Throughout, '**M**' represents millions, '**GB**' signifies Gigabytes, and '**s**' stands for seconds in our context. All the models were trained and tested using a single NVIDIA GeForce RTX 3080 Ti GPU.

Model Name	Total # of Conv Layers	Total # of Dense Layers	Total Params (M)	Trainable Params (M)	Non-Trainable Params (M)	Memory Usage (GB)	File Size (MB)	Train Epoch time (in s)	Inference time (in s)
Inception V3	94	3	22.0695	22.0351	0.0344	2.8340	253.6691	57.3908 ± 0.8968	0.1043 ± 0.0864 1
Inception ResNetV2	244	3	54.5379	54.4774	0.0605	7.0000	626.9542	152.2778 ± 1.1853	0.2043 ± 0.3543
NASNetLarge	268	3	85.4375	85.2408	0.1967	17.6370	981.3203	245.2935 ± 1.0038	0.2442 ± 0.0342
VGG16	13	3	14.7848	14.7848	0.0000	2.3970	169.3453	100.7890 ± 0.6179	0.2033 ± 0.0192
Xception	40	3	21.1282	21.0737	0.0545	5.5510	242.1432	105.0321 ± 2.3296	0.1522 ± 0.0689
Resnet101	104	3	42.9249	42.8196	0.1053	8.9790	492.3583	157.7575 ± 3.1522	0.2447 ± 0.1200
ViT	0	19	21.6575	21.6575	0.0000	10.3453	242.48	98.7612 ± 0.7352	0.2763 ± 0.4154
Domain Adaptation Xception model	40	6	28.3379	28.2834	0.0545	1.8070	324.6068	134.6777 ± 0.4488	0.1007 ± 0.0493

Table 8: The assessment of model complexities for all models trained on the CRV-PBS dataset entails the utilization of state-of-the-art architectures. These models were tested with a common batch size of 16 and a learning rate of 1e-04, tailored to their respective datasets. Notably, the Domain Adaptation model was trained on a combined dataset comprising the 8 common classes of CRV-PBS and PBC cropped. It's important to note that '**M**' represents millions, '**GB**' signifies Gigabytes, and '**s**' stands for seconds in our context. All the models were trained and tested using a single NVIDIA GeForce RTX 3080 Ti GPU.

6. Conclusion

In this study, we created a novel dataset comprising peripheral blood smears for benchmarking instance segmentation pipelines. Through investigations, we have noticed that Mask R-CNN alone cannot provide high classification accuracy, so we use innovative techniques such as domain adaptation to increase the performance of classification. Since the creation and collection of blood smear data are very expensive, we rely on imbalanced and fewer source data and a similar target dataset for this task. The convolution backbone for domain adaptation has been selected by performing a comparative study using transfer learning, which shows excellent results in both datasets. Using domain adaptation successfully enhances the classification performance of the Mask R-CNN pipeline. Our statistical experiments examining computational complexities reveal that our Domain Adaptation model excels in achieving a harmonious equilibrium among key computational resources, including the number of parameters, training time, and inference time. Importantly, this balance is achieved without compromising on the model's ability to deliver notably superior accuracy and precision in classifications compared to alternative models.

The initial comparative study on classifying the PBC dataset shows we can surpass the state-of-the-art by using whole slides rather than cropped cells. Using transfer learning helps to get impressive performance with relatively less amount of data. This will help the medical practitioners and doctors to cater to a large group of individuals in the time of emergency. The current research provides the outline of the creation of a larger dataset that can get deployable results even by the use of Mask R-CNN. The only limitation of this work is the current metrics record precision of 86.71%, a recall of 85.67%, and an F1-score of 84.89% even after domain adaptation. Medical sectors need stronger confidence in the detection of cell types for automating the process of blood tests. Increasing the dataset size will help to resolve this issue, for which we are planning to add more data soon. We are also planning to release the data for a benchmarking challenge which will test the precision, recall, and F1-score of different novel

methods. The future scope of this work lies in the creation of an annotation software that can annotate a diverse domain of peripheral blood smear data automatically with minimum human intervention.

Acknowledgement(s)

The authors are thankful to the editors and anonymous reviewers for their rigorous comments that helped to improve the quality of this paper significantly. The authors are also grateful to Swathy Prabhu Mj, Ramakrishna Mission Vivekananda Educational and Research Institute, for arranging machines with Asus RTX 2080 Ti (12 GB VRAM) and Quadro GV100 (32 GB VRAM) GPUs with 64 GB RAM to hasten the research. The authors are also thankful to Dripta Mj for his suggestions.

Disclosure statement

- Conflict of interest/Competing interests - The authors declare no competing interests.
- Ethics approval - The data is ethically cleared for study and future publication.
- Consent to participate - Waiver for consent was approved by the ethical committee.
- Consent for publication - The waiver for consent was approved by the ethical committee.
- Availability of data and materials: We will make the data available upon publication of this paper, on the following URL `https://github.com/Jimut123/RV-PBS`. This data should not be used for commercial purposes.
- Code availability - The code will be available at the following URL upon publication: `https://github.com/Jimut123/cellseg`. All the models

and tools created during this study will also be made available in the same URL.

- Authors' contributions (CRediT) - **Jimut Bahan Pal**: Conceptualization, Investigation, Methodology, Software Validation, Formal analysis, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization. **Aniket Bhattacharyea**: Investigation, Methodology, Software Validation, Data Curation. **Debasis Banerjee**: Data Curation, Supervision, Project administration. **Br. Tamal Maharaj**: Supervision, Project administration, Resources, Writing - Review & Editing.

Funding

There was no funding available for this study.

7. Appendices

Appendix A. Evaluation metrics

In this work, the following evaluation metrics are used:

$$\text{Accuracy (AC)} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$\text{Recall or Sensitivity (SE)} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{Precision (PC)} = \frac{TP}{TP + FP} \tag{4}$$

$$\text{F1-Score} = \frac{2 \times (PC \times SE)}{PC + SE} \tag{5}$$

Where TP is True Positive, TN is True Negative, FP is False Positive and FN is False Negative. The binary Cross-Entropy loss function was used in the Domain Classifier. It can be formulated as:

$$L_{y'}(y) := -\frac{1}{N}\sum_{i=1}^{N}(y'_i \log(y_i) + (1 - y'_i)\log(1 - y_i)) \tag{6}$$

Where y_i is the predicted class and y'_i is the original class value (log = ln, natural log). Categorical Cross-Entropy loss function can be written as :

$$L_y(y') := -\frac{1}{N}\left(\sum_{i=1}^{N} y'_i \cdot \log(y_i)\right) \tag{7}$$

Where y_i is the predicted class, and y'_i is the true value of a label predicted, generally in one-hot-encoded form.

Appendix B. Unsupervised Domain Adaptation

Here is a brief discussion of Ganin's (Ganin & Lempitsky, 2015) Unsupervised domain adaptation. Let us consider input x passed to the domain adaptation pipeline. Here, $x \in X$ and $y \in Y$ are input images and corresponding label pairs. We can consider two distributions, i.e., 8 common classes of CRV-PBS as source $S(x, y)$ and 8 common classes of C-PBC as Target $T(x, y)$ distribution. Since we are taking the common classes from both datasets, the images might differ by some domain shift. The primary aim of the domain adaptation pipeline is to predict Y from X. During training, we take a huge number of samples from the target distribution, which helps the classification model to improve classification performance on the source distribution.

Since there are two domains of similar data, we have an extra parameter to consider, i.e., domain d,

$$d_i = \begin{cases} 0, & \text{if } x_i \in \text{Source domain, } y_i \in \text{Y are known} \\ 1, & \text{if } x_i \in \text{Target domain, } y_i \in \text{Y are necessarily not known} \end{cases}$$

The task is to predict the target domain's input samples in test time. For every input sample $x \in X$, we can use a Feed Forward Deep Learning Convolutional Neural Network to predict $y \in Y$ and its corresponding domain labels $d_i \in \{0, 1\}$. From Figure 18 we get a $\mathbb{D}$ dimensional feature vector $f \in R^{\mathbb{D}}$ by passing x through G_f. Here, we have used the Xception (Chollet, 2017) model's convolutional backbone as G_f for extracting features, since it performed relatively well in classifying the cropped cell images of both domains.

During training, we try to minimize the loss of G_f and G_y, i.e., feature extractor and label predictor. This is done by passing images from both the domains, i.e., $S(x, y)$ and $T(x, y)$ to G_f. We try to make these two distributions, i.e., source, $S(f) = \{G_f(x; \theta_f) \| x \in S(x)\}$ and target, $T(f) = \{G_f(x; \theta_f) \| x \in T(x)\}$ as close to each other. We try to maximize the loss of the Domain classifier, i.e., Categorical Cross-Entropy loss, as discussed in Appendix 7. This helps the model get more confused about the domains in which the images belong, hence the model learns domain invariant features θ_f. The total loss can be formulated as:

$$\begin{aligned} E(\theta_f, \theta_y, \theta_d) = & \sum_{\substack{i=1..N \\ d_i=0}} L_y \left(G_y(G_f(x_i; \theta_f); \theta_y), y_i \right) - \\ & \lambda \sum_{i=1..N} L_d \left(G_d(G_f(x_i; \theta_f); \theta_d), y_i \right) \qquad (8) \\ = & \sum_{\substack{i=1..N \\ d_i=0}} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1..N} L_d^i(\theta_f, \theta_d) \end{aligned}$$

Here, since the label predictor $L_y(.,.)$ is a classifier, hence the loss is multinomial. Similarly, $L_d(.,.)$, the domain classifier loss is logistic. L_y^i and L_d^i are the corresponding loss functions that are evaluated at the i^{th} training example. We are seeking the parameters $\hat{\theta}_f$, $\hat{\theta}_y$, $\hat{\theta}_d$ that deliver a saddle point of the function:

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg \min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d) \qquad (9)$$

$$\hat{\theta}_d = \arg \max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d)\,. \qquad (10)$$

At the saddle point, θ_d of the domain classifier minimizes the domain classification loss, while parameters θ_y minimize the label predictor loss. Discriminative features are learned by the label predictor. It maximizes the domain classification loss learning domain invariant features. The overall architecture of this process can be seen in Figure 18.

Standard optimizers such as Stochastic Gradient Descent can be used as optimizers. The stationary saddle point is a part of the following stochastic updates, where μ may be considered as the learning rate, which may vary over time:

$$\theta_f \quad \longleftarrow \quad \theta_f \; - \; \mu \left(\frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f} \right) \tag{11}$$

$$\theta_y \quad \longleftarrow \quad \theta_y \; - \; \mu \frac{\partial L_y^i}{\partial \theta_y} \tag{12}$$

$$\theta_d \quad \longleftarrow \quad \theta_d \; - \; \mu \frac{\partial L_d^i}{\partial \theta_d} \tag{13}$$

A special layer known as Gradient Reversal Layer (GRL) can achieve the reduction as shown in Equation 11. The layer is inserted between the feature extractor and the domain classifier, as shown in Figure 18. The meta-parameter is not updated by back-propagation. During forward propagation, GRL acts as an identity transform. During back-propagation, the GRL multiplies the gradient by -λ and sends it to the previous layer. Hence the total loss can be formulated as 14

$$\begin{aligned} \tilde{E}(\theta_f, \theta_y, \theta_d) = & \sum_{\substack{i=1..N \\ d_i=0}} L_y \left(G_y(G_f(x_i; \theta_f); \theta_y), y_i \right) + \\ & \sum_{i=1..N} L_d \left(G_d(R_\lambda(G_f(x_i; \theta_f)); \theta_d), y_i \right) \end{aligned} \tag{14}$$

After learning the label predictor, $y(x) = G_y(G_f(x; \theta_f); \theta_y)$ can predict inputs from both the domains.

References

Acevedo, A., Alférez, S., Merino, A., Puigví, L., & Rodellar, J. (2019). Recognition of peripheral blood cell images using convolutional neural networks. *Computer Methods and Programs in Biomedicine*, *180*, 105020. URL: `https://www.sciencedirect.com/science/article/`

pii/S0169260719303578. doi:https://doi.org/10.1016/j.cmpb.2019.105020.

Acevedo, A., Merino, A., Alferez, S., Molina, A., Boldu, L., & Rodellar, J. (2020). A dataset for microscopic peripheral blood cell images for development of automatic recognition systems. Mendeley Data, V1, doi: 10.17632/snkd93bnjr.1.

Adewoyin, A., & Nwogoh, B. (2014). Peripheral blood film - a review. *Annals of Ibadan Postgraduate Medicine*, *12*, 71 – 79.

Al-qudah, R., & Suen, C. Y. (2020). A survey on peripheral blood smear analysis using deep learning. In Y. Lu, N. Vincent, P. C. Yuen, W.-S. Zheng, F. Cheriet, & C. Y. Suen (Eds.), *Pattern Recognition and Artificial Intelligence* (pp. 725–738). Cham: Springer International Publishing.

Beucher, S. (1994). Watershed, hierarchical segmentation and waterfall algorithm. In J. Serra, & P. Soille (Eds.), *Proceedings of the 2nd International Symposium on Mathematical Morphology and Its Applications to Image Processing, (ISMM) 1994, Fontainebleau, France, September 1994* (pp. 69–76). Kluwer volume 2 of *Computational Imaging and Vision*. URL: https://doi.org/10.1007/978-94-011-1040-2_10. doi:10.1007/978-94-011-1040-2_10.

Bringay, S., Barry, C., & Charlet, J. (2006). Annotations for the collaboration of the health professionals. *AMIA - Annual Symposium proceedings. AMIA Symposium*, (pp. 91–5).

Chadaga, K., Prabhu, S., Bhat, K. V., Umakanth, S., & Sampathila, N. (2022). Medical diagnosis of COVID-19 using blood tests and machine learning. *Journal of Physics: Conference Series*, *2161*, 012017. URL: https://doi.org/10.1088/1742-6596/2161/1/012017. doi:10.1088/1742-6596/2161/1/012017.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1800–1807). doi:10.1109/CVPR.2017.195.

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. URL: https://proceedings.neurips.cc/paper/2016/file/577ef1154f3240ad5b9b413aa7346a1e-Paper.pdf.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248–255). doi:10.1109/CVPR.2009.5206848.

Deshpande, N. M., Gite, S., & Aluvalu, D. R. (2021). A review of microscopic analysis of blood cells for disease detection with ai perspective. *PeerJ Computer Science*, *7*.

Deshpande, N. M., Gite, S., Pradhan, B., Kotecha, K., & Alamri, A. M. (2022). Improved otsu and kapur approach for white blood cells segmentation based on lebtlbo optimization for the detection of leukemia. *Mathematical biosciences and engineering : MBE*, *192*, 1970–2001.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=YicbFdNTTy.

Endeshaw, T., Gebre, T., Ngondi, J., Graves, P. M., Shargie, E. B., Ejigsemahu, Y., Ayele, B., Yohannes, G., Teferi, T., Messele, A., Zerihun, M., Genet, A., Mosher, A. W., Emerson, P. M., & Richards, F. O. (2008). Evaluation of light microscopy and rapid diagnostic test for the detection of malaria under operational field conditions: a household survey in ethiopia. *Malaria Journal*, *7*, 118 – 118.

Ganin, Y., & Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In F. Bach, & D. Blei (Eds.), *Proceedings of the 32nd International Conference on Machine Learning* (pp. 1180–1189). Lille, France: PMLR volume 37 of *Proceedings of Machine Learning Research*. URL: https://proceedings.mlr.press/v37/ganin15.html.

Guan, H., & Liu, M. (2022). Domain adaptation for medical image analysis: A survey. *IEEE Trans. Biomed. Eng.*, *69*, 1173–1185. URL: https://doi.org/10.1109/TBME.2021.3117407. doi:10.1109/TBME.2021.3117407.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 2980–2988). doi:10.1109/ICCV.2017.322.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778). doi:10.1109/CVPR.2016.90.

Hoffman, J. F. (2016). Biconcave shape of human red-blood-cell ghosts relies on density differences between the rim and dimple of the ghost's plasma membrane. *Proceedings of the National Academy of Sciences*, *113*, 14847–14851. URL: https://www.pnas.org/content/113/51/14847. doi:10.1073/pnas.1615452113. arXiv:https://www.pnas.org/content/113/51/14847.full.pdf.

Hsu, H.-K., Hung, W.-C., Tseng, H.-Y., Yao, C.-H., Tsai, Y.-H., Singh, M., & Yang, M.-H. (2019). Progressive domain adaptation for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Ibtehaz, N., & Rahman, M. S. (2020). Multiresunet: Rethinking the u-net architecture for multimodal biomedical image segmentation. *Neural Networks*, *121*, 74–87.

Johnson, J. M., & Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *J. Big Data*, *6*, 27. URL: https://doi.org/10.1186/s40537-019-0192-5. doi:10.1186/s40537-019-0192-5.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio, & Y. LeCun (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. URL: http://arxiv.org/abs/1412.6980.

Kouzehkanan, Z. M., Saghari, S., Tavakoli, S., Rostami, P., Abaszadeh, M., Mirzadeh, F., Satlsar, E. S., Gheidishahran, M., Gorgi, F. A., Mohammadi, S., & Hosseini, R. (2022). A large dataset of white blood cells containing cell locations and types, along with segmented nuclei and cytoplasm. *Scientific Reports*, *12*.

Kukar, M., Guncar, G., Vovko, T., Podnar, S., Cernelc, P., Brvar, M., Zalaznik, M., Notar, M., Moskon, S., & Notar, M. (2021). Covid-19 diagnosis by routine blood tests using machine learning. *Scientific Reports*, *11*.

Labati, R. D., Piuri, V., & Scotti, F. (2011). All-idb: The acute lymphoblastic leukemia image database for image processing. In B. Macq, & P. Schelkens (Eds.), *18th IEEE International Conference on Image Processing, ICIP 2011, Brussels, Belgium, September 11-14, 2011* (pp. 2045–2048). IEEE. URL: https://doi.org/10.1109/ICIP.2011.6115881. doi:10.1109/ICIP.2011.6115881.

Lee, S. M. W., Shaw, A., Simpson, J. L., Uminsky, D. T., & Garratt, L. W. (2021). Differential cell counts using center-point networks achieves human-level accuracy and efficiency over segmentation. *Scientific Reports*, *11*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (pp. 740–755). Cham: Springer International Publishing.

Linden, M., Ward, J. M., & Cherian, S. (2012). 19 - hematopoietic and lymphoid tissues. In P. M. Treuting, & S. M. Dintzis (Eds.), *Comparative Anatomy and Histology* (pp. 309–338). San Diego: Academic Press. URL: https://www.sciencedirect.com/science/article/pii/B9780123813619000196. doi:https://doi.org/10.1016/B978-0-12-381361-9.00019-6.

Liu, S., & Deng, W. (2015). Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* (pp. 730–734). doi:10.1109/ACPR.2015.7486599.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer Vision – ECCV 2016* (pp. 21–37). Cham: Springer International Publishing.

Murez, Z., Kolouri, S., Kriegman, D., Ramamoorthi, R., & Kim, K. (2018). Image to image translation for domain adaptation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4500–4509). doi:10.1109/CVPR.2018.00473.

Pal, J. B. (2022). Holistic network for quantifying uncertainties in medical images. In A. Crimi, & S. Bakas (Eds.), *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (pp. 560–569). Cham: Springer International Publishing.

Pal, J. B., & Mj, D. (2023). Improving multi-scale attention networks: Bayesian optimization for segmenting medical images. *The Imaging Science Journal*, *71*, 33–49. doi:10.1080/13682199.2023.2174657.

Pal, J. B., & Paul, N. (2021). Classifying chest x-ray covid-19 images via transfer learning. In *2021 Ethics and Explainability for Responsible Data Science (EE-RDS)* (pp. 1–8). doi:10.1109/EE-RDS53766.2021.9708580.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*, 1345–1359. doi:10.1109/TKDE.2009.191.

Pandey, P., P, P. A., Kyatham, V., Mishra, D., & Dastidar, T. R. (2020). Target-independent domain adaptation for wbc classification using generative latent search. *IEEE Transactions on Medical Imaging*, *39*, 3979–3991. doi:10.1109/TMI.2020.3009029.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 779–788). doi:10.1109/CVPR.2016.91.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc. volume 28. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

Robbins, H. (2007). A stochastic approximation method. *Annals of Mathematical Statistics*, *22*, 400–407.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, & A. F. Frangi (Eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (pp. 234–241). Cham: Springer International Publishing.

Schlemper, J., Oktay, O., Schaap, M., Heinrich, M., Kainz, B., Glocker, B., & Rueckert, D. (2019). Attention gated networks: Learning to leverage salient regions in medical images. *Medical Image Analysis*, *53*, 197–207. URL: https://www.sciencedirect.com/science/article/

pii/S1361841518306133. doi:https://doi.org/10.1016/j.media.2019.01.012.

Sekachev, B., Manovich, N., Zhiltsov, M., Zhavoronkov, A., Kalinin, D., Hoff, B., TOsmanov, Kruchinin, D., Zankevich, A., DmitriySidnev, Markelov, M., Johannes222, Chenuet, M., a andre, telenachos, Melnikov, A., Kim, J., Ilouz, L., Glazov, N., Priya4607, Tehrani, R., Jeong, S., Skubriev, V., Yonekura, S., vugia truong, zliang7, lizhming, & Truong, T. (2020). opencv/cvat: v1.1.0. URL: https://doi.org/10.5281/zenodo.4009388. doi:10.5281/zenodo.4009388.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2818–2826). doi:10.1109/CVPR.2016.308.

Thomas, E., Pawan, S. J., Kumar, S., Horo, A., Niyas, S., Vinayagamani, S., Kesavadas, C., & Rajan, J. (2021). Multi-res-attention unet: A cnn model for the segmentation of focal cortical dysplasia lesions from magnetic resonance images. *IEEE Journal of Biomedical and Health Informatics*, *25*, 1724–1734. doi:10.1109/JBHI.2020.3024188.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc. volume 30. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vento, M., & Percannella, G. (Eds.) (2019). *Computer Analysis of Images and Patterns - 18th International Conference, CAIP 2019, Salerno, Italy, September 3-5, 2019, Proceedings, Part II* volume 11679 of *Lecture Notes in Computer Science*. Springer. URL: https://doi.org/10.1007/978-3-030-29891-3. doi:10.1007/978-3-030-29891-3.

Wang, Q., Bi, S., Sun, M., Wang, Y., Wang, D., & Yang, S. (2019). Deep learning approach to peripheral leukocyte recognition. *PLOS ONE*, *14*, 1–18. URL: https://doi.org/10.1371/journal.pone.0218808. doi:10.1371/journal.pone.0218808.

Willemink, M. J., Koszek, W. A., Hardell, C., Wu, J., Fleischmann, D., Harvey, H., Folio, L. R., Summers, R. M., Rubin, D., & Lungren, M. P. (2020). Preparing medical imaging data for machine learning. *Radiology*, (p. 192224).

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (pp. 818–833). Cham: Springer International Publishing.

Zheng, X., Wang, Y., Wang, G., & Liu, J. (2018). Fast and robust segmentation of white blood cell images by self-supervised learning. *Micron*, *107*, 55–71.

Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., & Liang, J. (2018). Unet++: A nested u-net architecture for medical image segmentation. In D. Stoyanov, Z. Taylor, G. Carneiro, T. Syeda-Mahmood, A. Martel, L. Maier-Hein, J. M. R. Tavares, A. Bradley, J. P. Papa, V. Belagiannis, J. C. Nascimento, Z. Lu, S. Conjeti, M. Moradi, H. Greenspan, & A. Madabhushi (Eds.), *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support* (pp. 3–11). Cham: Springer International Publishing.

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. doi:10.1109/CVPR.2018.00907.