POLITECNICO DI TORINO

DIPARTIMENTO DI AUTOMATICA E INFORMATICA
Master Degree in Computer Engineering

MASTER THESIS

Unsupervised feature extraction using autoencoders for analyzing single-cell images from acute myeloid leukemia patients

Supervisors
Prof. Tatiana Tommasi
Dr. Carsten Marr
Ario Sadafi

Candidate
Raheleh Salehi

Academic year 2022-2023

Dedication

Words cannot express my gratitude to Dr. Carsten Marr and Dr. Tatiana Tommasi for their invaluable patience, support, and feedback. I also could not have undertaken this journey without members of the Marr lab, who generously provided knowledge and expertise. Additionally, this endeavor would not have been possible without the generous support from Ario Sadafi, who guides and supports me.

I am also grateful to my flatmates for their support and positive energy during this journey that impacted and inspired me. I would also like to thank Dr. Ali Danaee who supported me all these years when I moved abroad to start my new life.

Lastly, I would be remiss in not mentioning my family, especially my parents, and my sisters. Their belief in me has kept my spirits and motivation high during this process.

Abstract

Identification and classification of white blood cells in peripheral blood smears is a key step for the diagnosis of hematological malignancies. Different lab procedures, illumination, staining, and microscope settings are resulting in domain shifts, which hamper the reusability of machine learning methods when applied to data collected from different sites. In this thesis, we propose an autoencoder to extract unsupervised cross-domain features on three different datasets of single white blood cells. Using a Mask R-CNN architecture as a first step allows the autoencoder to focus on the relevant white blood cell and eliminate artifacts in the image. A simple random forest method is used to classify the extracted features of the single cells as a way to evaluate the quality of the features extracted by the autoencoder. We show that the random forest classifier trained on only one of the datasets can perform satisfactorily on the unseen datasets thanks to the rich features extracted by the autoencoder. In the cross-domain task, it outperforms published oracle models. According to the results, this unsupervised approach can be employed in more complicated diagnosis and prognosis tasks without the need to add expensive expert labels to unobserved datasets and it is expected that it will improve the applicability and reliability of machine learning algorithms in different centers and hospitals.

Contents

Chapter 1

Introduction

Hematopoietic malignancies such as leukemia rank among the leading causes of death and are a major obstacle to increase of life expectancy in countries worldwide during the past decade Deschler and Lübbert [2008]. Leukemia has different types which are based on histopathologic and histochemical findings in peripheral blood and bone marrow Buechner et al. [1985]. The main four types of Leukemia are chronic myelogenous leukemia (CML), chronic lymphocytic leukemia (CLL), Acute myeloid leukemia (AML), and acute lymphocytic leukemia (ALL). Chronic forms of leukemia affect middle-aged to older adults, while acute types mostly affect children and young adults. Based on data from SEER NCI [2021], it is estimated that there will be around 60,650 new cases of leukemia and an estimated 24,000 people will die of this disease in the next two decades. ALL and AML are among common childhood cancers and most often in older adults. It is slightly more common in men than women. A leukemia diagnosis can be challenging to process and it depends on age, gender, and some other parameters.
Cytomorphologists evaluate white blood cells under the microscope in blood or bone marrow smears for proper diagnosis. So far, this morphological analysis has not been automated and it requires trained experts checking blood smears under the microscope. Trained experts are expensive and the process is time-consuming. Computer-aided diagnosis systems are thus an essential element helping cytologists to speed up the process and have accurate performance. In recent years, deep learning approaches are more and more applied to biomedical tasks and are demonstrating their considerable capabilities, especially in image processing tasks, mostly due to hardware improvements and development of new algorithms. The promising ability of deep learning methods has supported them as a primary option for computer based diagnosis and recognition particularly in medical images. Since diagnosing hematological malignancies requires identification and classification of white blood cells in peripheral blood smears, classification based on deep learning techniques has received vast attention and recent works show the potential for automation of this medical task. For instance, Matek et al. [2019] have defined a highly accurate supervised approach based on convolutional neural networks ResNext Xie et al. [2017] architecture for the classification of white blood cells in blood smears of acute myeloid leukemia patients. They are using over 18,000 white blood cell images to train the model for the classification of the most important cell types with high accuracy based

on blood smears. In another work Matek et al. [2021], they suggest a CNN method for cell morphologies classification in bone marrow smears as well. Boldú et al. [2019] have suggested a predictive machine-learning approach for diagnosis of acute myeloid leukemia in peripheral blood cell images. They have developed a model that can identify different types of blasts (myeloid and lymphoid origin) and pathological promyelocytes and lymphocytes from normal mononuclear cells such as lymphocytes and monocytes. Their proposed model has two steps: First, the image is segmented, and features are extracted, this is called image processing in their pipeline.
Final recognition module includes a linear discriminant analysis (LDA). LDA is used for reducing the dimensionality of the features and to find a combination of features helping the classification. In the second step, the method predicts the patient's disease diagnosis based on the individual cells. In another work, Acevedo et al. [2021] suggested a machine-learning model for the automatic diagnosis of patients who are suffering from myelodysplastic syndrome, a preform of acute myeloid leukemia. The model is used two different neural network architectures, Vgg-16 and Inceptionv3. Firstly the model is trained on two neural networks and extracted the features, then these features were used to train a support vector machine classifier. In the second case, they suggested using the same networks as two end-to-end models for classification of the eight white blood cells classes. All of these studies have used data that are provided by a single center. However, there are many factors that can affect the microscopic images such as camera resolution, microscope settings, illumination, and staining protocols in laboratory procedures. Additionally, there are many cases where it is difficult to gather datasets that have all the required verification and diversity to train robust neural networks. These changes can affect model performance considerably. Datasets need to be annotated again and the models should be re-trained every time for every center. This challenge is called domain shift Sankaranarayanan et al. [2018]. Domain adaptation can be a solution to align different domains in the data. The aim is to train a model on one dataset as a source and get similar high performance on other target datasets. Based on the target domain data, domain adaptation methods can be categorized into the following:

- Supervised: Both the source domain and the target domains are fully annotated.
- Semi-Supervised: Both source and target domains are labeled partially, but unlabelled data in the source domain and target domains exists too.
- Unsupervised: Samples are not labeled in the target domains.

There are many learning setups that have been applied before under different names in domain adaptation methods. Dou et al. [2019] formalized a learning semantic feature space by incorporating global and local constraints in a supervised approach. They introduced two complementary losses to explicitly regularize the semantic structure of the feature space. Chen et al. [2020] have proposed a conduction synergistic alignment of both images and features for unsupervised domain adaptation method. In their method, they considered two perspectives of alignments by an adversarial learning regime with a shared feature encoder to get their mutual benefits for decreasing the gap between them in an end-to-end training. Finally, Ranzato and Szummer [2008] have published a standard

semi-supervised problem by ignoring the domain difference and considering the source instances as labeled data and the target ones as unlabeled data. Published methods we are aware of are altering the classifier by retraining on the new domain to change decision boundaries. Most of them require at least a few labels in the target domains.

1.1 Scientific contribution

In this thesis, we present an AutoEncoder-based Cell Feature Extractor (AE-CFE). It is a simple, economic, and robust approach for feature extraction single white blood cells. The proposed method is based on instance features that are extracted by a Mask R-CNN He et al. [2017] architecture as a white blood cell detector and feature extractor, then we propose an autoencoder to get the features of single white blood cells in digitized blood smears. Since the data is coming from different centers, we are defining a domain adaptation loss to decrease the discrepancy between the source distribution and the target distributions. The proposed approach is the first unsupervised two-staged autoencoder method for cross-domain feature extraction. While many domain adaptation methods have been published, there is no unsupervised feature extraction method that is able to work across different domains without requiring instance labels. We outperformed the published supervised methods on the unseen white blood cell datasets and offered a more robust decision support algorithm for diagnosing hematopoietic malignancies. The implementation is publicly available at https://github.com/marrlab/AE-CFE.

Chapter 2

Methods

2.1 Image classification

Image classification Guo et al. [2017] is a challenge in computer vision and the goal is to categorize the objects present in an image into a set of known predefined categories. Two general methods of classification are “supervised” and “unsupervised”. In unsupervised classification methods, a algorithm is using the specified characteristics of an image systematically during the image processing stage. An architecture that can automatically learn the underlying distribution of the data could answer questions about the data, for instance about correlations or clusters. The classification methods used are “image clustering” or “pattern recognition”. Supervised classification method is the process of visually selecting samples within the image and assigning them to pre-selected categories. A breakthrough in building models for image classification came with the discovery that a convolutional neural network (CNN) could be used to progressively extract higher- and higher-level representations of the image content. Instead of preprocessing the data to derive features like textures and shapes, a CNN takes just the image’s raw pixel data as input and "learns" how to extract these features, and ultimately infer what object they constitute.

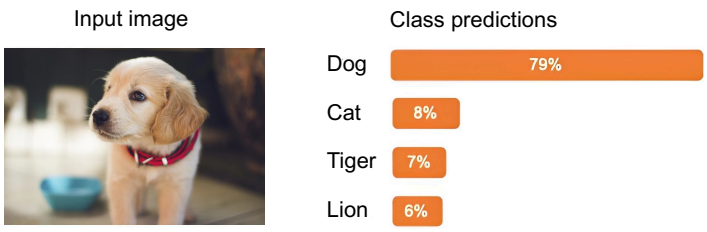

Figure 2.1. Example of image classification. The deep learning model returns classes along with the detection probability [figure from creative commons/rawpixel.com].

2.2 Object detection

Object detection is a technique in computer vision to locate instances of objects in images or videos. When humans look at images or videos, they can recognize the instances of objects and detect their positions. The aim of using object detection algorithms is not only to classify the present objects in an image but also to locate where they are.

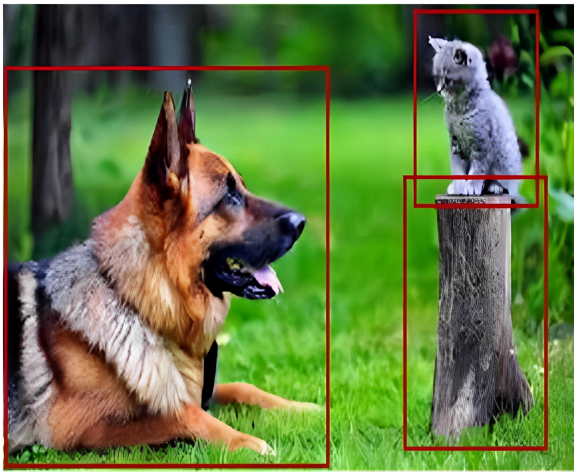

Figure 2.2. Exemplary object detection on an image. Three objects of Dog, Cat, and Trunk are detected [figure from https://www.boredpanda.com/].

There are two key steps in any object detection algorithm:

- To **detect** the instances of objects and their boundaries in the image.
- To **classify** the instances of objects into a set of known groups.

In general, there are two categories of deep object detectors: single-staged and two-staged methods. Single-staged methods are faster and more suitable for real-time scenarios. Redmon et al. [2016] for example is one of the single-staged object detectors. On the other hand, two staged approaches are more accurate in localization and recognition of the objects while requiring more resources and are slower in computation. In these methods first, the candidate bounding boxes are generated in the first stage, and in the second stage the candidate bounding boxes are refined and classified. R-CNN based methods, such as Faster R-CNN Ren et al. [2015] is an example of a two-staged object detector.

2.3 Instance segmentation

In instance segmentation algorithms are quite similar to instance detectors, while in addition to localization and classification of the objects, they provide binary masks indicating which pixels of an image belong to the detected objects. Mask R-CNN is one of the most widely used two staged instance segmentation algorithms.

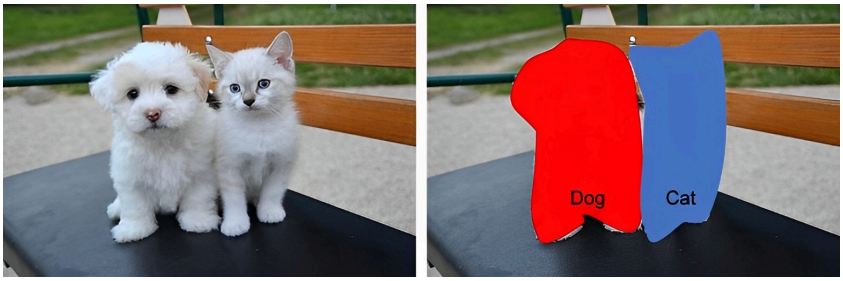

Figure 2.3. Exemplary instance segmentation on two images. The instances are segmented and categorized into the classes [figure from https://pixabay.com/] .

2.4 Mask R-CNN

Mask R-CNN He et al. [2017] is a Convolutional Neural Network (CNN) and one of the widely used state-of-the-art algorithms in instance segmentation tasks. It is an extension of Faster R-CNN Ren et al. [2015] which was developed previously for object detection. Mask R-CNN is able to segment instances found in an image and it focuses on detecting and segmenting every instance in the image. There are two stages in a Mask R-CNN model. The first stage consists of a feature extraction step over the whole image and a region proposal network (RPN) suggesting candidate bounding boxes all over the image. In the second stage extracted features are cropped out for every instance by RoIAlign to be analyzed by different heads of the network to classify and refine the bounding box, and instance segmentation. The feature extractor of the Mask R-CNN is based on a ResNet101 He et al. [2016] backbone with Feature Pyramid Network (FPN) Lin et al. [2017].

Feature Pyramid Network helps the recognition of objects at different scales. It takes a single-scale image and then returns a sized feature map at multi-levels and is calculated independently of the backbone of the convolutional neural network. There is a bottom-up pathway and a top-down pathway in the construction of the pyramid. The feedforward computation of the backbone ConvNet computes a hierarchy of feature maps in several scales for the bottom-up pathway. While in the top-down pathway, it samples spatially coarser but semantically stronger, hallucinated in higher resolution features. Then those features are raised via lateral connections from the bottom-up pathway. The task for lateral connections is to merge feature maps from the bottom-up pathway and the top-down pathway.

ResNet101 architecture is a convolutional neural network with 101 layers. A pretrained version of the network is trained on more than a million images from the ImageNet database. The image input size is 224 by 224 and the network has learned rich feature representations for a wide range of images.

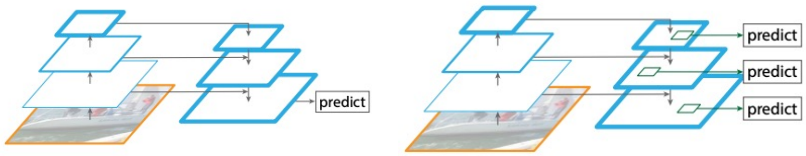

Figure 2.4. In the left figure, in a top-down method without the connections, the features are predicted on the finest level. On the right figure, it is a Feature Pyramid Network that has two pathways with predictions independently [figure from Lin et al. [2017]].

2.4.1 Loss function

Mask R-CNN is a combination of three different tasks including classification, bounding box, and segmentation mask. For each of those, a loss is defined and the overall loss of the model is the linear summation of the three:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{mask}} \tag{2.1}$$

Where $\mathcal{L}_{\text{cls}}$ is the log-likelihood loss for classification and $\mathcal{L}_{\text{box}}$ is bounding box regression loss, and $\mathcal{L}_{\text{mask}}$ is a mask branch output compared with the groundtruth. The classification loss is the classic negative log-likelihood $\mathcal{L}_{\text{cls}}(p, u) = -\log(p_u)$ for the true class u. Where p represents the predicted probability and u is the predicted value of the corresponding label. Bounding box regression loss Girshick et al. [2014] consists of the hyper-parameter λ to set the balance of the two tasks and it uses a smooth function.

$$\mathcal{L}_{\text{box}}(t^u, v) = \lambda \sum_{i\epsilon(x,y,w,h)} \text{smooth}_{L_1}(t^u - v) \tag{2.2}$$

$\mathcal{L}_{box}$ calculates the similarity of the t^u and v where t^u is predicted tuple for class u and v is a ground-truth bounding-box regression target. The smooth_{L1} function is a robust $\mathcal{L}_1$ loss that is sensitive to outliers.

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \tag{2.3}$$

The definition of $\mathcal{L}_{mask}$ is the average binary cross-entropy loss when considering u^{th} mask and the region is related to the ground truth class u.

$$\mathcal{L}_{\text{mask}} = -\frac{1}{m^2} \sum_{1 \le i,j \le m} \left[y_{ij} \log \hat{y}^u_{ij} + (1 - y_{ij}) \log(1 - \hat{y}^u_{ij}) \right] \tag{2.4}$$

where y is the label of the object (i, j) in the corrected mask for the bounding box of mxm size, and $\hat{y}^u_{ij}$ is the predicted mask value of the same cell corresponding to the ground-truth class u.

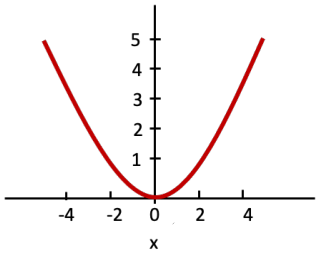

Figure 2.5. The plot of smooth$_{L1}$ loss [figure from Lin et al. [2017]].

2.4.2 Augmentation

Mask R-CNN is one of the critical components in our design because correct detection of white blood cell images and extraction of the relevant features is crucial for the next steps. To increase the detection accuracy of unseen data, we need to increase the dynamics in our training samples so that we obtain a robust model able to detect single cells and extract features in different illuminations, microscope settings, and other domain-specific characteristics of the data. Since it is hard to get more real data in medical or biological cases; data augmentation is a good solution to obtain more training data. Augmenting the data means that the number of existing samples is increased by generating new data points using a set of techniques without changing the sample labels Goodfellow et al. [2016], while the relevant information is retained in the images. The augmentation techniques that we used on the datasets are:

- **Random flips and mirroring**: The image is randomly flipped and mirrored on the horizontal and vertical axis. It is a perfect technique because the cells can have different orientations.
- **Affine rotation and scaling**: The image is randomly rotated or zoomed in and out. It is a good technique because there are three different datasets with different resolutions.
- **Changing color temperature**: The color temperature of the image is modified to a provided value.
- **Changing linear contrast**: The contrast of the image is adjusted by modifying pixel values to $127 + alpha \times (v - 127)$ where v is the original pixel value.
- **Random bilateral blur**: The image is blurred by a bilateral filter by setting the max distance parameter.

There are two different ways to augment datasets. Offline and online data augmentation. In offline augmentation during dataset generation, augmented images are saved on the disk and are repeated for every epoch. It is not recommended as there is a higher chance that the model overfits the data. While in online data augmentation, the image

is augmented randomly during the training, and each time the model sees a new image with a much lower chance of repetition. It is more efficient in storage but requires more CPU time, and there is less control over the augmentation process.

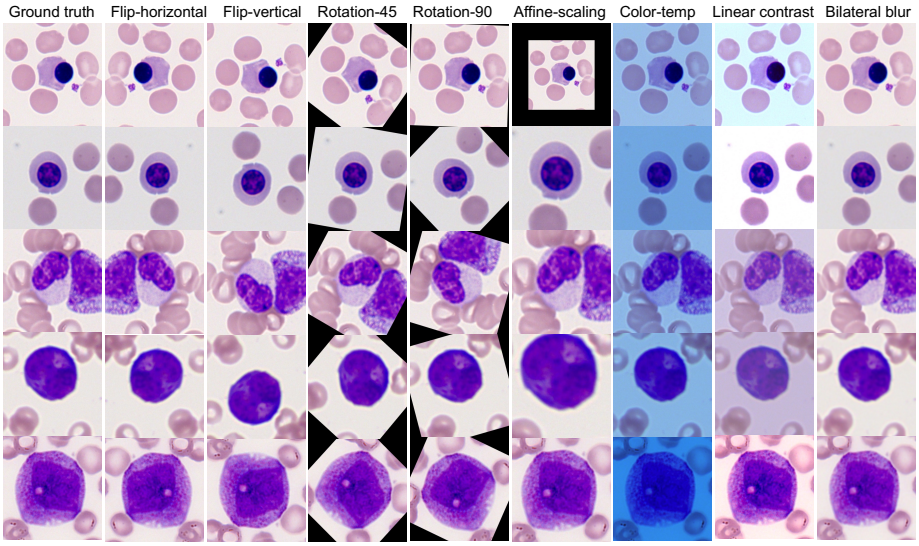

Figure 2.6. Some common augmentation methods applied on exemplary images from the datasets.

2.5 Autoencoder

Autoencoders are unsupervised neural network models that can be used for feature representation and dimension reduction Kramer [1991], Hinton and Salakhutdinov [2006]. The autoencoder can extract the features from unlabeled input data, encodes it to a low dimensional latent space, and then attempts to reconstruct the input data at the output layer Baldi [2012]. The common autoencoder consists of two components called encoder and decoder. The encoder tries to compress the input to a low dimensional representation, the decoder then reconstructs the input only based on that. An autoencoder implementation involves deciding about several hyperparameters. These hyperparameters are "latent space size", "number of layers", "number of nodes in each layer", and "loss function". The big challenge for autoencoders is learning a meaningful and concise latent space representation. Furthermore, the autoencoders are learning features specific to the training data, they can be a close but degraded representation.

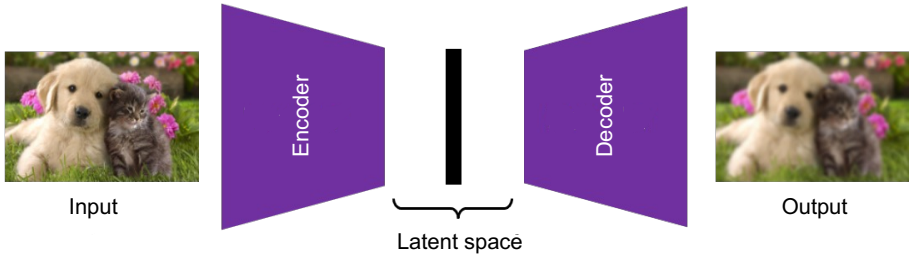

Figure 2.7. This is an example of a common autoencoder; the encoder takes an image as input, and compresses the input into a latent space dimension. Then the decoder tries to reconstruct the output from this representation [figure from https://www.rawpixel.com/].

A common autoencoder has two parts, encoder and decoder. In the simplest case,

$$z_i = f_{\text{encoder}}(I_i) \quad : \quad \forall I_i \epsilon D \tag{2.5}$$

Where z_i is the feature representation of the latent space in i^{th} the image from the dataset.

$$\hat{x} = f_{\text{decoder}}(z_i) \tag{2.6}$$

where $\hat{x}$ is the reconstructed images and f_{decoder} is a neural network to map the latent space representation z_i to reconstruct the input.
to minimize the reconstruction errors, reconstruction loss is defined:

$$\mathcal{L}(x, \hat{x}) = ||x - \hat{x}||^2 = ||x - \hat{\sigma}(f_{\text{decoder}}(\sigma(f_{\text{encoder}})))||^2 \tag{2.7}$$

where σ is the potentially activation function to be used.

2.6 Normalization techniques

Normalization techniques have been widely used in deep learning models, for better stabilization of the activiations of the hidden units and faster network convergence Shen et al.

[2021]. In recent years, a variety of normalization techniques have been proposed such as batch normalization, weight normalization, layer normalization, instance normalization, and group normalization (GN).

- **Batch Normalization** is a technique that tries to improve the training neural network performance by stabilizing the feature distribution. It standardizes the features for each mini-batch, by using the mean and variance to normalize features in a layer. The biggest issue when using batch normalization is the large batch size it requires to work correctly. Using batch normalization with a small batch size leads to a lower performance. Hence, since the GPU memory is limited one has to compromise between the batch size and the model size Santurkar et al. [2018].
- **Weight Normalization** is a technique to improve the conditioning of the optimization problem and speed up the convergence of stochastic gradient descent. It can work even for small mini-batches in a neural network. Implementing the weight normalization method is inspired by batch normalization's property of adding noise to the gradients. It is used to reparameterize each weight vector in terms of a parameter vector and scalar parameter for getting the desired output with respect to those parameters instead Salimans and Kingma [2016].
- **Layer Normalization**: While batch normalization normalizes each feature independently across the mini-batch. Layer normalization Ba et al. [2016] normalizes each of the inputs in the batch independently across all features. There is one axis corresponding to the batch and the other axis is for feature dimensions in this strategy.
- **Instance Normalization** It is another term of contrast normalization. It performs intensity normalization across the width and height of a single feature map of a single example. For example, Ulyanov et al. [2016] applied Instance Normalization instead of batch normalization on deep neural networks for image generation. The main difference between instance normalization and batch normalization. Instance normalization operates on a single sample of a channel. While batch normalization applies for the whole samples of a channel in the mini-batch.

2.6.1 Group normalization

The representation of feature distribution is critical in the latent space for different domains. To generalize the latent space representation with different domains, Group Normalization (GN) Wu and He [2018] is used in each layer of the encoder. GN is a combination of layer normalization and instance normalization strategies and it is suitable for both sequential and generative models. In this technique, the channels are divided into groups and the specific means and variance are calculated for each group to be normalized. GN calculation is not dependent on batch size and the accuracy stabilizes in a wide range of batch sizes. Many classical features like SIFT Lowe [2004] and HOG Dalal and Triggs [2005] have group-wise features and they include group-wise normalization. For instance, the result of the several spatial cells is a HOG vector which represents a normalized orientation histogram in each cell.

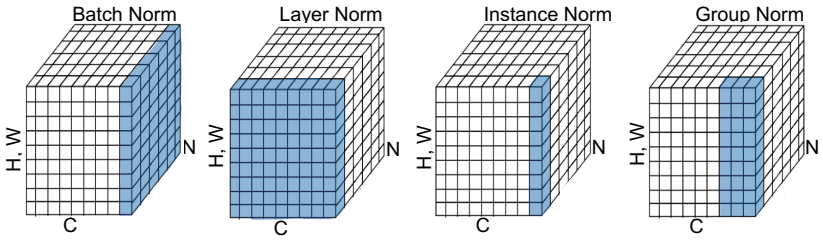

Figure 2.8. Visualization of different types of normalization techniques are used in deep learning approaches [figure from Wu and He [2018]].

2.7 Domain adaptation techniques

Gathering the datasets from different labs that might have different lab procedures, staining, lighting conditions, and camera viewpoints leads to domain shift in the data. Because of the domain shift, it is pretty difficult to train the model on a specific dataset and then apply it to another dataset for the same task that the model has not seen before. One of the solutions is exposing domain shifts in the optimization of the model to align different domains. In particular, we want to propose a feature extraction method that is invariable between different domains. In this thesis, one dataset is always considered to come from a single source, and the target domain can be more than one dataset. We have tested several domain adaptation methods to find the best for our problem.

2.7.1 Normal distribution alignment

The idea behind normal distribution alignment Sun and Saenko [2015] is very intuitive. Different representations coming from each dataset have their own specific distribution in the latent space of the autoencoder. Given a normal distribution with the mean matrice and the standard deviation as 0 and 1 respectively, not only the model tries to bring the euclidean distance closer by changing the value of the mean metrics but also tries to minimize the symmetric Kullback-Leibler divergence (KLD) for the difference between the covariance matrices. Domain adaptation loss is defined as follows:

$$\mathcal{L} = \sum_{k=1}^{k} \left\{ |D(\mu_k) - D(\mu_0)| + \frac{1}{2} \left[D_{KL}(s_0||s_k) + D_{KL}(s_k||s_0) \right] \right\} \tag{2.8}$$

With $D = D_1, ..., D_K$ being the datasets, and the μ_k is mean and s_k as softmax of covariance matrix of the embedded features of dataset D_K.

2.7.2 Adversarial domain adaptation

Adversarial domain adaptation Tzeng et al. [2017] is another method to solve the domain shift problem by reducing the domain discrepancy between different datasets. The idea

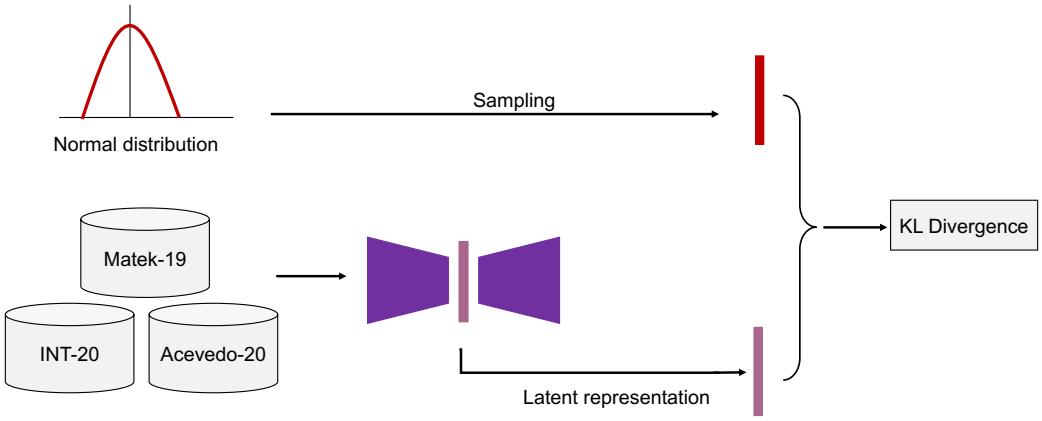

Figure 2.9. The normal distribution strategy is applied as a source distribution and the other datasets are aligned to the source distribution through the domain adaptation loss.

of the technique is based on Generative Adversarial Networks (GANs) that two networks called generators and discriminators are competing with each other. The generator is trying to generate fake outputs that look real while the discriminator tries to distinguish the fake from the real images. If the discriminator is able to recognize the fake example from the real one, the generator is punished with an adversarial loss.

$$\mathcal{L} = \beta\mathcal{L}_{\text{autoencoder}} + \beta\mathcal{L}_{\text{discriminator}} \tag{2.9}$$

$$\mathcal{L}_{\text{autoencoder}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{h}_i - h_i)^2 + 1 - \text{SSIM}(\hat{r}_i, r_i) \tag{2.10}$$

$$\mathcal{L}_{\text{discriminator}} = \text{NLLLoss}(x_{\text{real}}, x_{\text{fake}}) \tag{2.11}$$

NLLLoss is negative log likelihood loss. To describe the formula:

$$l(x_{\text{real}}, x_{\text{fake}}) = \{l_1, ..., l_N\}^T, l_n = -W_{x_{\text{fake}}} X_{n,x_{\text{fake}}} \tag{2.12}$$

Where x_{real} is source and x_{fake} is target, w is the weight and N is the batch size.

$$\mathcal{L}(x_{\text{real}}, x_{\text{fake}}) = \left\{ \begin{array}{ll} \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n}} l_n, & \text{if reduction} = \text{"mean"}; \\ \sum_{n=1}^{N} l_n, & \text{if reduction} = \text{"sum"}. \end{array} \right\} \tag{2.13}$$

2.7.3 Maximum mean discrepancy

The third domain adaptation technique we tried to apply the Maximum Mean Discrepancy (MMD) method Borgwardt et al. [2006] on the different distributions in the latent space

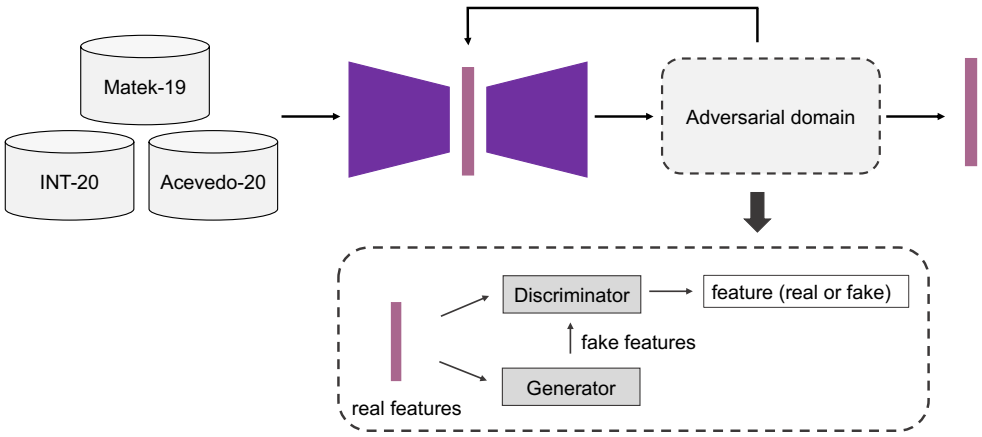

Figure 2.10. Overall view of an adversarial domain adaptation in our study.

to minimize the distance between the distributions. A mean squared error is used for mean matrices while a symmetrized Kullback-Leibler (KL) divergence is minimizing the difference between the covariance matrices.

To formulate MMD, having $D = \{D_1, ..., D_k\}$ as the set of all datasets, and mean matrix μ_k and s_k as softmax of the covariance matrix of any dataset D_k the MMD domain adaptation loss is calculated by:

$$\mathcal{L}_{\text{DA}} = \sum_{k=1}^{k} \left\{ \text{MSE}(\mu_k, \mu_0) + \frac{1}{2} \left[D_{\text{KL}}(s_0 || s_k) + [D_{\text{KL}}(s_k || s_0)] \right] \right\} \tag{2.14}$$

2.8 Classification methods

Classification is a supervised task in which machine learning algorithms learn how to predict a class for samples from the problem domain. To evaluate our unsupervised approach and the quality of the extracted features, we require a task to measure our performance on it. Trying to classify the instances in the dataset based on the extracted features of our model is the task we decided to use for evaluation. In binary classification, the trained classifier tries to classify the input data into only two different classes. For instance, the dataset consists of only healthy and unhealthy samples while in multilabel classification more than two classes exist. For example predicting the subtypes of a disease or diagnosis of different diseases.

2.9 Random forest

Random forest is a simple and powerful classification method used to classify various sub-samples of the dataset. It has several decision tree classifiers that fit on sample data

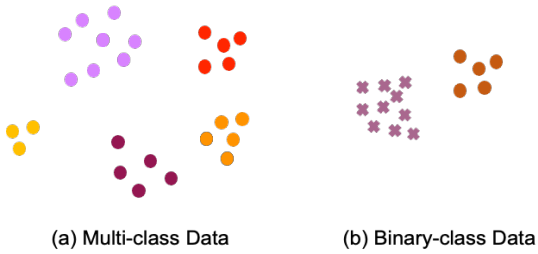

Figure 2.11. An example of multi-label and binary classification methods in machine learning approaches.

and uses averaging to improve the prediction accuracy. Several important parameters are defined for the model Breiman [2001] such as: number of estimators that are the number of the trees in the forest, maximum depth of each tree, the minimum samples split that are required to split an internal node and minimum samples leaf that are required to split an internal node.

2.10 Autoencoder-based cell feature extractor

We present an AutoEncoder-based Cell Feature Extractor (AE-CFE), which starts with a Mask R-CNN model to obtain features of single white blood cells in blood smears. This way there is a specific feature vector extracted for every detected cell instance with dimensionality of 2561414. An autoencoder receives the feature vector as input and a two-stage decoder modalities are developed. The decoder components involve a feature decoder and image decoder, that try to reconstruct the latent space representation to encoded features and the single cell images respectively. The encoder is a fully convolutional network and consists of 6 layers. To stabilize training independent from the batch size and to improve the model performance, a group normalization (GN) method is applied after each layer in the encoder as an alternative to batch normalization. For the encoder, part z_i is the extracted features obtained from the encoder by

$$z_i = f_{\text{enc}}(h_i; \theta) \tag{2.15}$$

where h_i is the input features from the Mask R-CNN, which is compressed by the autoencoder in the latent space and θ is the parameters in the encoder.

$$\hat{h_i} = f_{\text{feat}-\text{dec}}(z_i; \lambda) \tag{2.16}$$

$\hat{h_i}$ is reconstructed features by the feature decoder from the latent space α considered as the parameters in the feature decoder.
r_i is the reconstructed image by the image decoder according to the reconstructed features with defining β as parameters in image decoder training.

$$\hat{r_i} = f_{\text{img}-\text{dec}}(\hat{h_i}; \delta) \tag{2.17}$$

For optimizing our two-staged autoencoder we define a multi-task loss as:

$$\mathcal{L}_{\text{autoencoder}_{(\theta,\gamma,\delta)}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{h}_i - h_i)^2 + 1 - \text{SSIM}(\hat{r}_i, r_i) \tag{2.18}$$

Where the quality of the reconstructed features and the input features are calculated with a mean square error and for measuring the similarity of the reconstruction image x with the original single cell image y which is detected by Mask R-CNN, we define the structural similarity index measure (SSIM) Wang et al. [2004] as:

$$\text{SSIM(s, y)} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2.19}$$

where mean and variance of the images are defined as μ and σ respectively and c_1 and c_2 are constants for numerical stability.

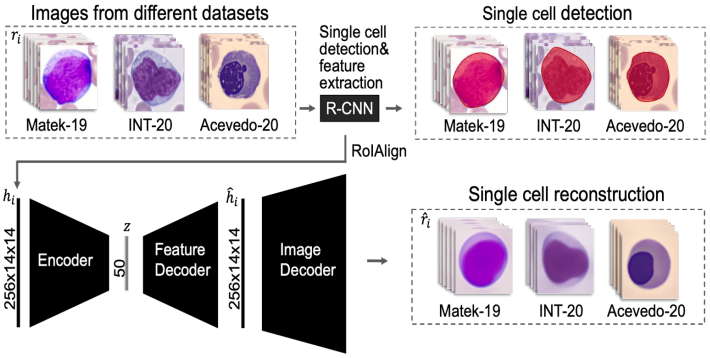

Figure 2.12. The proposed autoencoder method starts with the Mask R-CNN method which detected and extracted the single cell and feature from all over the datasets. The autoencoder consists of two decoders which are called Feature Decoder and Image Decoder separately to reconstruct features and images from the 50 bottleneck size [figure from Salehi et al. [2022]].

The domain adaptation loss introduced in section **Maximum Mean Discrepancy** (2.7.3) by constant coefficient β . To evaluate the overall training loss,

$$\mathcal{L}(\lambda, \varphi, \Omega) = \frac{1}{N}\sum_{i=1}^{N}(\hat{h}_i - h_i)^2 + 1 - \text{SSIM}(\hat{r}_i, r_i) + \beta\mathcal{L}_{\text{DA}} \tag{2.20}$$

Chapter 3

Experiments and results

In this chapter we explain the three datasets used in the experiments and further elaborate on the utilized hyper-parameters. Experiments are presented in detail, along with the results. The aim is to determine if morphological classification of white blood cells is possible from different datasets.

3.1 Datasets

We use three different datasets to evaluate our method:

Matek-19 dataset includes over 18,000 annotated white blood cells. This dataset is collected from 100 acute myeloid leukemia patients from the leukemia diagnostics laboratory at Munich University Hospital between 2014 and 2017. The dataset is categorized into 15 different classes with 400×400 pixels image dimensions for each image or 29×29 micrometers almost. This data is published and is publicly available Matek et al. [2019].

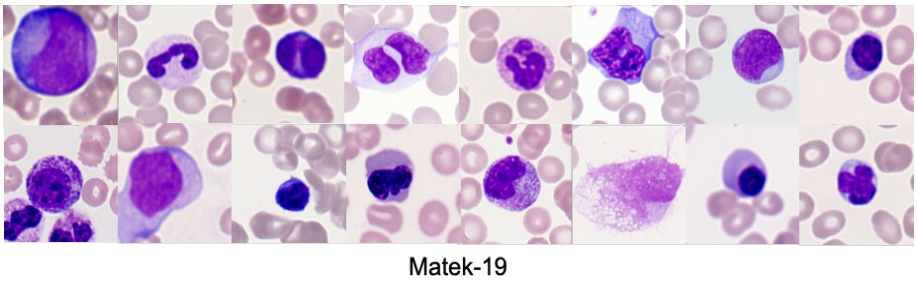

Figure 3.1. The example images of Matek-19 dataset, it has 15 different types of white blood cells classes.

INT-20 is an in-house dataset consisting of around 42,000 images in 18 different classes. Image dimensions are 288×288 in pixels or 25×25 micrometers.

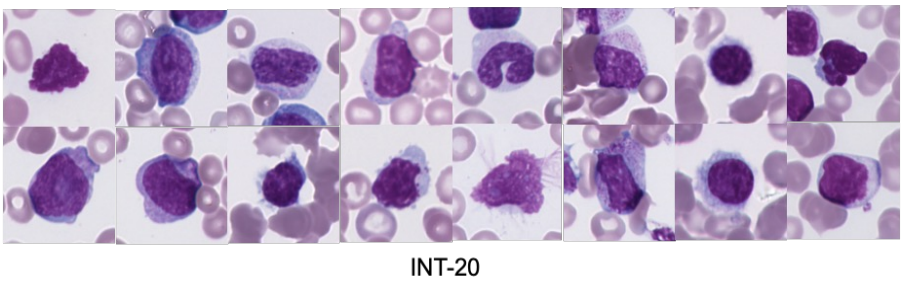

Figure 3.2. The example images of the INT-20 dataset, it repairs 18 different classes.

Acevedo-20 dataset has a total of 17,092 samples of individual normal cells coming from 8 different classes. The dataset is collected in the core laboratory at the Hospital Clinic of Barcelona and published by Acevedo et al. [2020]. Image dimensions are 360×363 pixels or 36×36.3 micrometers. Since each dataset has a different definition of

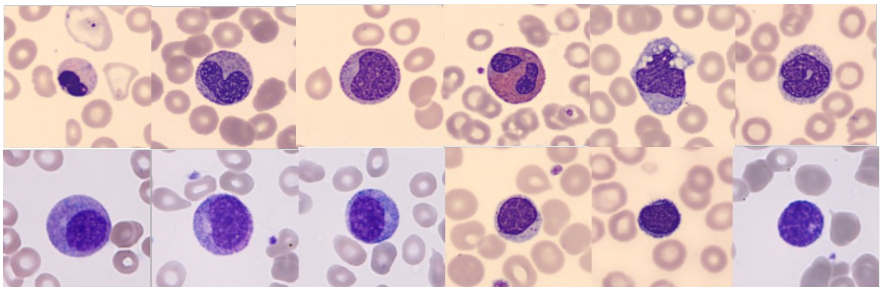

Figure 3.3. The example images of Acevedo-20 dataset from eights different classes.

classes, a medical expert helped us to categorize different labels into 13 commonly defined classes which are: basophil, eosinophil, erythroblast, myeloblast, promyelocyte, myelocyte, metamyelocyte, neutrophil banded, neutrophil segmented, monocyte, lymphocyte typical, lymphocyte atypical, and smudge cells. In Fig.3.4., sample distribution of commonly defined classes is shown in three datasets. The distribution is unbalanced and some classes are empty for the Acevedo-20 dataset.

3.2 Mask R-CNN dataset

The Mask R-CNN model is trained on a small separate dataset of around 1500 images annotated from the Matek-19 dataset. Since the annotated images did not need any expertise, it is extremely simple and affordable to annotate the images, increasing the applicability of our model. We annotated 1500 samples from different classes without any cell labels.

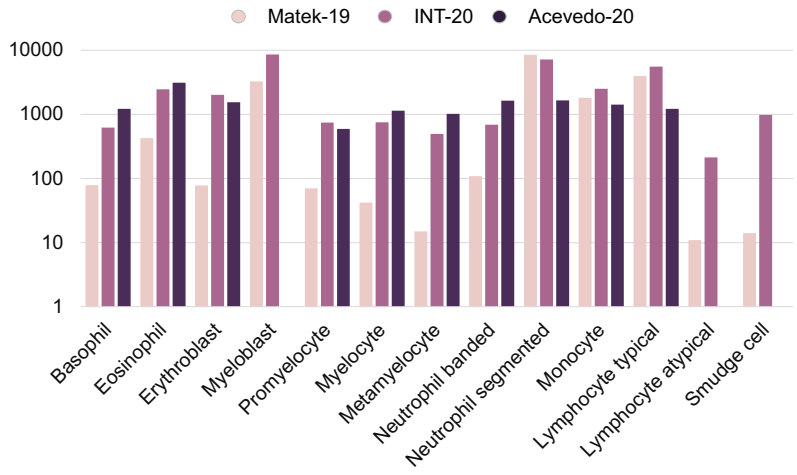

Figure 3.4. Distribution of samples between these 13 classes for different datasets [figure from Salehi et al. [2022]].

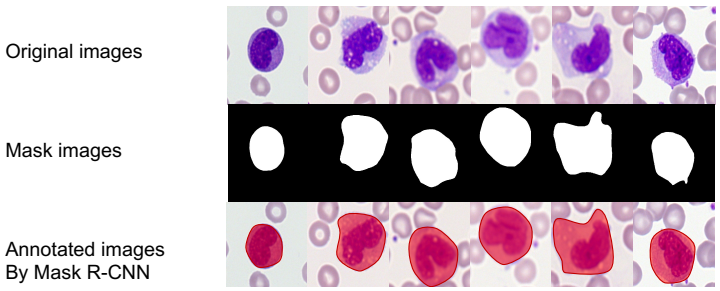

Figure 3.5. An example of segmented white blood cell images in the Matek-19 dataset.

3.2.1 Training Mask R-CNN

The implementation of Mask R-CNN on Python3, Keras, and TensorFlow is available. The model generates bounding boxes for the instances over the image, based on the ResNet101 Feature Pyramid Network (FPN) backbone.

3.2.2 Setup and hyperparameters

The Mask R-CNN is trained for 26 epochs with a learning rate of 0.001 and Adam optimizer. To segment the white blood cells perfectly on the other datasets, we applied several different augmentation techniques which are explained in section 2.4.2 and 50 percent of the samples are augmented during the training in an online augmentation regime. The Mask R-CNN is reaching an mAP of 0.89 and 85% of the cells in datasets are detected successfully leading to 65,693 segmented single cells out of a total of 77,363 images coming from the three datasets.

3.3 Bottleneck size: A thorough study

In order to have a thorough evaluation of the results we are studying the performance of the model both qualitatively and quantitatively.

3.3.1 Qualitative evaluation

In all experiments for qualitative evaluations, firstly we trained Mask R-CNN with a ResNet50 backbone and then another Mask R-CNN model with a ResNet101 backbone. We conducted 8 experiments with the two models. Here is the list of every experiment in details:

- **ResNet50-MSE**: We use the whole image and MSE loss function for both decoders (Figure 3.6).
- **ResNet50-BB-MSE**: Images are cropped based on cell bounding boxes and we use MSE loss function for feature decoder and reconstructed image decoder (Figure 3.7).
- **ResNet50-SSIM**: Use the whole images and use only MSE loss function for feature decoder and SSIM loss function for reconstructed images (Figure 3.8).
- **ResNet50-BB-SSIM**: Images are cropped based on cell bounding boxes and use only MSE loss function for feature decoder and SSIM loss function for reconstructed images (Figure 3.9).
- Same 4 experiments repeated with a **ResNet101 backbone**.

We decided to firstly use qualitative results to select the best bottleneck size so we analyzed the performance of our method in the above experiments in different bottleneck sizes. Here, the reconstructed images compare for different bottleneck sizes in above experiments. A smaller bottleneck size leads to a lower quality of reconstruction due to the problem in passing very small amounts of information. A higher bottleneck leads to reconstruction of noisy red blood cells around the image which is also not desirable.

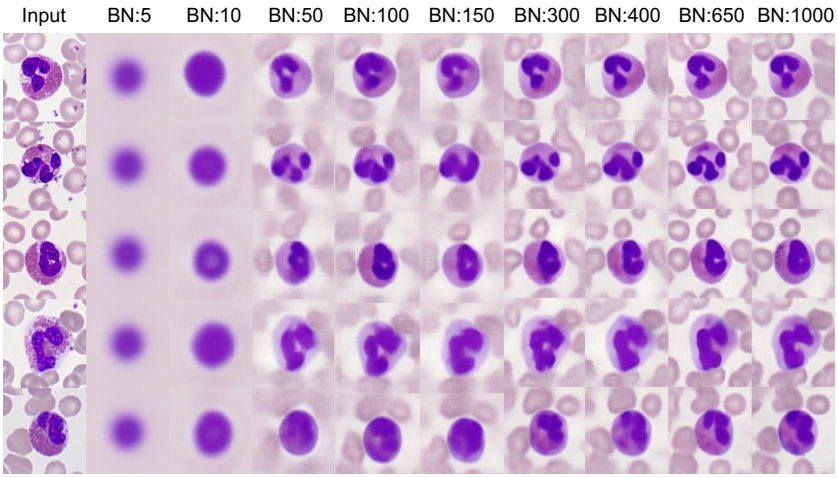

Figure 3.6. Example with different bottleneck size (BN) from 5 to 1000 of reconstructed images on **ResNet50-MSE**.

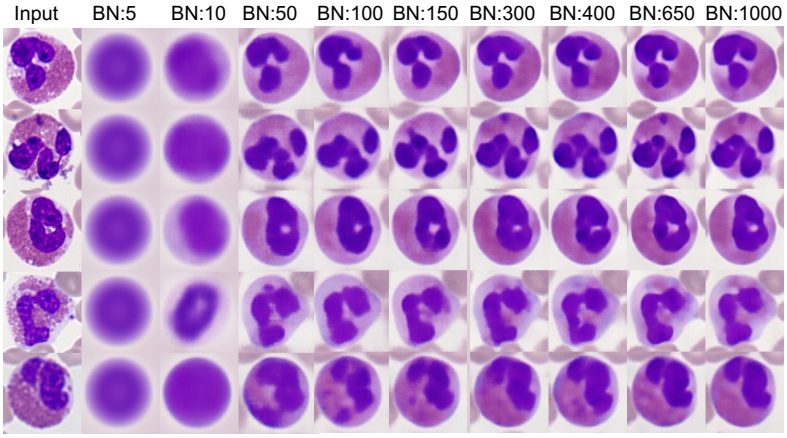

Figure 3.7. Example of reconstructed images on **ResNet50-BB-MSE**.

3.3.2 Quantitative evaluation

All experiments done in qualitative evaluations are studied in the quantitative evolutions as well. To quantitatively calculate the quality of the extracted features by our experiment models, the random forest model is trained on the extracted features trying to classify single white blood cells of the Matek-19 dataset. The Matek-19 dataset is split between the train set and the test set at 80% and 20% respectively. The results are provided in Table 3.1 for different bottleneck sizes of the autoencoder.

Looking at the qualitative and quantitative results, **ResNet101-BB-SSIM** successfully eliminated irrelevant information while the dimension of the latent space is chosen

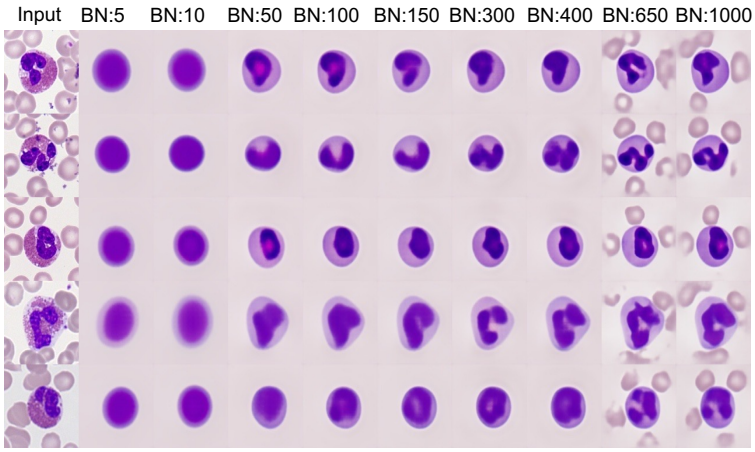

Figure 3.8. Example of reconstructed images on **ResNet50-SSIM**.

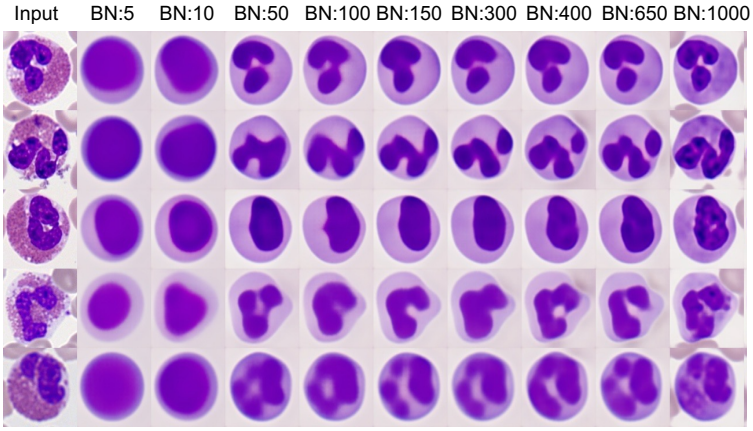

Figure 3.9. Example of reconstructed images on **ResNet50-BB-SSIM**.

to be **50** due to the simplicity to solve domain shift and being able to retain enough information.

3.4 Model architecture: An ablation study

To study the effectiveness of every component in our method we have designed an ablation study with five baselines:

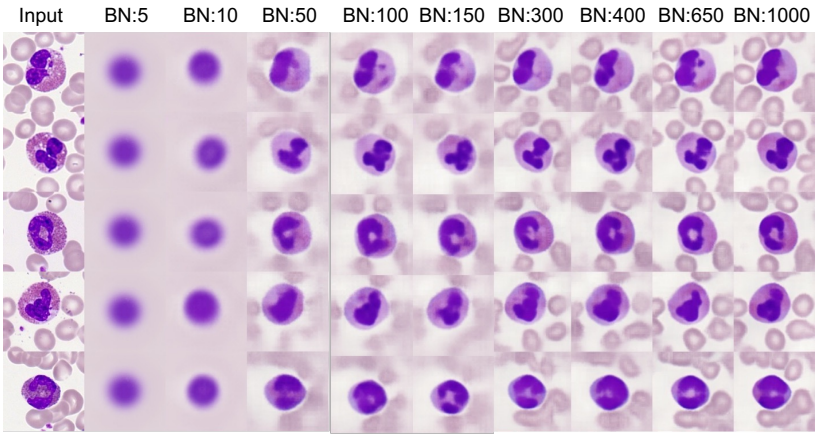

Figure 3.10. Example of reconstructed images on **ResNet101-MSE**.

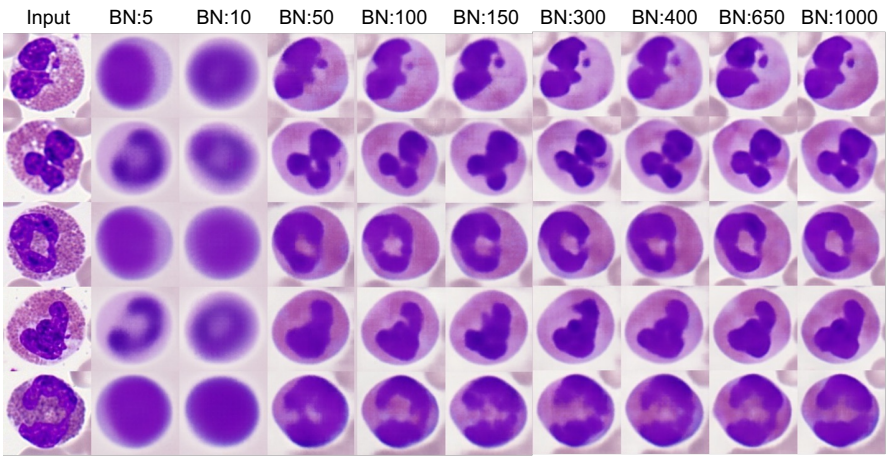

Figure 3.11. Example of reconstructed images on **ResNet101-BB-MSE**.

1. **ResNet-RF**: Using a pretrained ResNet101 He et al. [2016] architecture on ImageNet dataset Deng et al. [2009] to extract features and classify them with a random forest.

2. **R-CNN-RF**: Trained a random forest classification model on the instance features extracted by the Mask R-CNN architecture which is trained on single cell detection task.

3. **AE-RF**: A random forest classification trained on a similar autoencoder which is trained on our datasets with no domain adaptation technique.

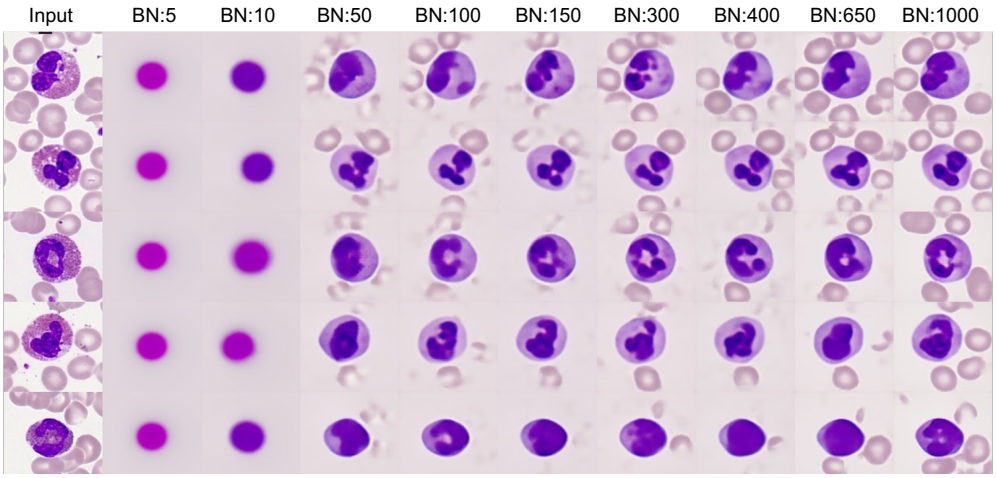

Figure 3.12. Example of reconstructed images on **ResNet101-SSIM**.

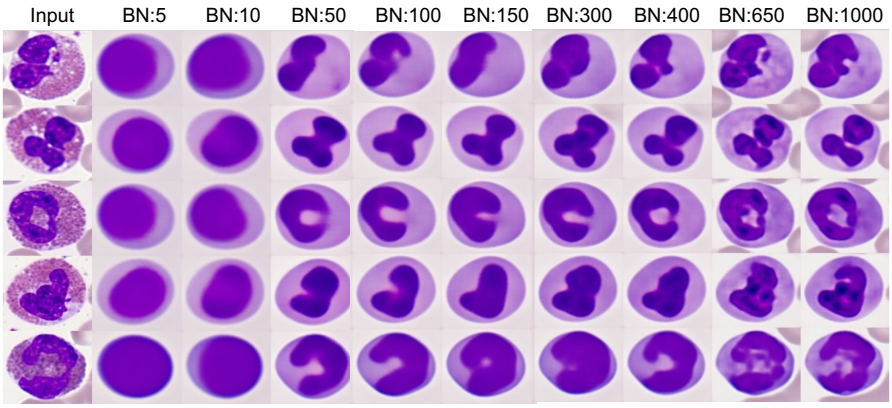

Figure 3.13. Example of reconstructed images on **ResNet101-BB-SSIM**.

4. **ND-AE-DA**: A normal distribution domain adaptation is applied on features extracted by the two-staged autoencoder.

5. **AAE-DA**: Trained the two-staged autoencoder on the adversarial domain adaptation on features extracted.

In all experiments for quantitative evaluations, we trained a random forest model on the extracted features from the AE-CFE model. The goal is to try to classify the single cell features into one of the 13 defined classes.

Table 3.1. Try to classify a random forest classifier on different autoencoder models in which a Mask R-CNN is trained on ResNet101 or ResNet50.

	AE-MSE		**AE-SSIM**		**AE-MSE-BB**		**AE-SSIM-BB**	
BN	ResNet101	ResNet50	ResNet101	ResNet50	ResNet101	ResNet50	ResNet101	ResNet50
5	0.46	0.46	0.46	0.72	**0.81**	0.46	0.78	0.72
10	0.59	0.66	0.52	0.72	0.74	0.75	**0.79**	0.78
50	0.81	0.79	0.82	0.81	**0.85**	0.84	**0.85**	0.84
100	0.82	0.79	0.82	0.82	0.86	0.84	**0.87**	0.85
150	0.82	0.80	0.83	0.83	0.86	0.84	**0.87**	0.85
300	0.82	0.80	0.82	0.83	0.85	0.83	**0.87**	0.85
400	0.82	0.81	0.81	0.83	0.86	0.84	**0.87**	0.86
650	0.84	0.77	0.82	0.82	0.87	0.82	**0.87**	0.85
1000	0.81	0.88	0.80	0.83	0.84	0.82	**0.86**	0.86

3.4.1 ND-AE-DA: Normal distribution alignment

The model tries to align the three distributions by help of a reference normal distribution through minimizing the euclidean distance between the means and symmetric Kullback-Leibler (KL) divergence for the difference between the covariance matrices.

Setup and hyperparameters

In this experiment, the reference normal distribution is the source and all datasets are considered as target, loss function calculates the similarity between source and targets in every iteration.

3.4.2 AAE-DA: Adversarial domain adaptation

To implement the proposed adversarial domain adaptation, there is a discriminator network to distinguish the fake from the real samples. In our setting, the adversarial discriminator tries to look at the latent feature vectors generated by the autoencoder and recognize the dataset it comes from. If the discriminator is able to distinguish, an adversarial loss is imposed on the autoencoder. The discriminator consists of 3 layers and all intermediate layers use the ReLU activation functions and there is not any set for the output of the discriminator.

Setup and hyperparameters

In this study, two Adam Optimizers are used during the training, an Adam optimizer for generator and discriminator with a learning rate of 0.001 and 0.001 respectively NLLLoss loss is used for the discriminator and adversarial loss is calculated by a combination of MSE and SSIM loss function

3.4.3 AE-CFE: Maximum mean discrepancy

For implementation of maximum mean discrepancy loss, Matek-19 is defined as the source, and the other datasets are considered as targets and loss function calculates the difference between the source and targets in every iteration.

Setup and hyperparameters

Training is done with use of an Adam optimizer for 150 epochs with a learning rate of 0.001 on three NVIDIA A100-SXM4-40GB GPUs with a total batch size of 1500 (500 on each GPU).

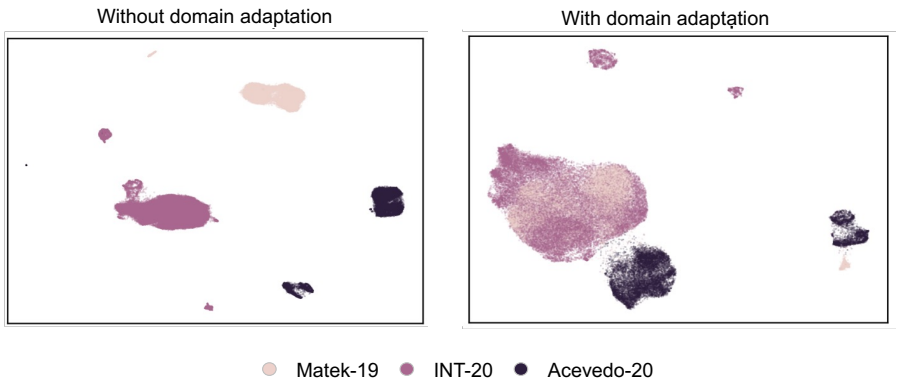

Figure 3.14. UMAP embedding of AE-CFE method before and after domain adaptation method. After using the domain adaptation loss, there is a uniform distribution in the representation space [figure from Salehi et al. [2022]].

Here, The different oracle methods are compared with our results. All the experiments share the main hyperparameters and the training procedure:

- In all experiments, the models are trained for 150 epochs using Adam Optimizers with learning rate of 0.001 on three NVIDIA A100 SXM4 40 GB CPUs.
- The whole datasets are splitted on 80% and 20% as train set and test set respectively. Then a random forest model is trained on one dataset and tested on the test set of all three datasets. We repeat these experiments for all datasets. The mean and standard deviation of accuracy is reported over 5 runs.

In Table 3.2 we compare these oracle methods with the quality of features extracted from our model. For two of the baselines (ResNet-RF and R-CNN-RF) cross-domain evaluations were inaccurate and accuracy was close to zero and random forest was not able to classify any sample correctly. Next in Table 3.3, we compared our method with the oracle methods which are specifically trained for the classification of the datasets.

Matek et al. Matek et al. [2019] have published their ResNext architecture and trained model weights. Additionally a similar ResNext model is trained on each of the datasets to compare and evaluate our method. The results in both tables show that both of the oracles fail on the unseen dataset while the random forest method trained on AE-CFE feature vectors is performing far better on unseen datasets.

Table 3.2. The accuracy percentage of a random forest model trained on AE-CFE model with five other feature extraction models as baselines: ResNet 101 trained on ImageNet, extraction features with Mask R-CNN, an autoencoder trained on instance feature vector, a normal distribution domain adaptation trained on extracted features, and adversarial domain adaptation on trained on features extraction.

Trained on	Tested on	ResNet-RF	R-CNN-RF	AE-RF	ND-AE-DA	AAE-DA	AE-CFE
Matek-19	Matek-19	62.5±2.8	60.5.± 0.8	86.0±0.04	83±0.4	**87.5± 0.8**	83.7±0.5
	INT-20	0	0	46.8±0.2	40±0.21	31.4± 0.31	**48.4±0.2**
	Acevedo-20	0	0	20.1±0.1	8.4±0.8	8.6± 0.4	**21.9±0.4**
INT-20	Matek-19	0	0	47.2±3.4	30.0±0.1	63.9±0.2	**73.2±0.1**
	INT-20	45.2±1.1	46.09±0.4	**69.1±0.4**	66.3±0.3	66.8±0.4	65.6±0.5
	Acevedo-20	0	0	4.6±0.6	15.7±0.9	17.7±0.7	**31.8±0.4**
Acevedo-20	Matek-19	0	0	39.4±1.4	15.2±0.4	39.4±0.6	**45.1±0.5**
	INT-20	0	0	9.7±0.3	16.0±0.6	17.7±0.7	**21.0±0.5**
	Acevedo-20	37.1±0.8	35.9±1.1	**67.2±0.7**	40.0±0.2	64.3±0.1	65.2±0.5

Table 3.3. Percentage of accuracy of a random forest model which is trained on AE-CFE features with 3 other oracle methods specifically trained for each of the datasets.

Trained on	Tested on	ResNext	Matek et al.	AE-CFE
Matek-29	Matek-19	-	**96.1**	83.7±0.5
	INT-20	-	29.5	**48.4±0.2**
	Acevedo-20	-	8.1	21.9±0.4
INT-20	Matek-19	49.6±6.3	-	**73.2±0.1**
	INT-20	88.7±1.5	-	65.6±0.5
	Acevedo-20	16.9±1.6	-	**31.8±0.4**
Acevedo-20	Matek-19	7.3±3.1	-	**45.1±0.5**
	INT-20	8.1±1.4	-	**21.0±0.5**
	Acevedo-20	**85.7±2.4**	-	65.2±0.5

Chapter 4

Discussion

Computer-aided diagnosis systems are rapidly entering in the health care landscape and have advanced to a level of maturity that permits them to be used under real-life conditions to help human decisions in some case of medical and healthcare environments. The diagnosis of many pathologies, such as leukaemia, infectious diseases or other hematopoietic malignancies disorders need the identification and classification of subtypes of white blood cells Topol [2019]. In this thesis, we investigated a method to focus only on white blood cell images because surrounding objects like red blood cells, are irrelevant and can greatly affect the performance of any model for any subsequent task, such as leukemia subtype diagnosis. For instance, the density of red blood cells that are around the white blood cells in the image can mislead the model into categorizing samples based on anemic features rather than the cytomorphological white blood cell properties. This makes the Mask R-CNN a critical element in our design. It is focusing on instance features that are cropped out by the region of interest instead of the whole image or features in the background. Since the feature vector is sparse and we aim on extracting meaningful features, a two-staged autoencoder is considered for the next step.
Still, there is a question: What happens if cells in new dataset e.g. from a different clinics, are considerably different? Mask R-CNN achieves an 85% detection rate for single white blood cells in three different datasets while it is trained on only one of them, making it a reliable single cell detector for the first stage. Annotation of the white blood cells in images for training the Mask R-CNN is cheap, simple, convenient, and fast, thus Mask R-CNN performance can be improved easily. However, due to the domain shift in the datasets, our features should remain invariant between different datasets making them useful for any subsequent model trained on the features.
Training diagnostic models can be challenging, requiring expensive and not always available medical experts so having domain invariant features is another critical task. We have tackled this problem using an Maximum Mean Discrepancy (MMD) loss trying to make features as similar as possible across different domains. For classification, using a random forest model which is trained on features extracted by our method does not perform well compared to oracle models that are specifically trained on the source domains, but its performance is by far superior in cross-domain scenarios. This is partly because of our design in domain adaptation loss that forces the distributions from three different datasets

to get closer to each other. Choosing the small feature vectors with minimum sparsity allows usage of these features in many different applications.

Our AuroEncoder-based Cell Feature Extractor (AE-CFE) approach which is a part of this master thesis is published in a paper for the **2022** edition of the Conference on Medical Image Computing and Computer Assisted Intervention - **(MICCAI)**.

Future works in this context can extend the approach and address some of its shortcomings: using additional features from cell nuclei can improve the extracted features quality. Additionally so far the domain shift is only assumed to be present between different datasets, so handling possible domain shift presents within dataset would be desirable. Furthermore, applying a feature extraction method to some diagnosis scenarios would better exhibit its capabilities. And finally trying different methods of domain adaptation are just some of the exciting directions to explore in the future.

Bibliography

Andrea Acevedo, Anna Merino, Santiago Alférez, Ángel Molina, Laura Boldú, and José Rodellar. A dataset of microscopic peripheral blood cell images for development of automatic recognition systems. *Data in brief*, 30, 2020.

Andrea Acevedo, Anna Merino, Laura Boldú, Ángel Molina, Santiago Alférez, and José Rodellar. A new convolutional neural network predictive model for the automatic recognition of hypogranulated neutrophils in myelodysplastic syndromes. *Computers in Biology and Medicine*, 134:104479, 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

Laura Boldú, Anna Merino, Santiago Alférez, Angel Molina, Andrea Acevedo, and José Rodellar. Automatic recognition of different types of acute leukaemia in peripheral blood by image analysis. *Journal of Clinical Pathology*, 72(11):755–761, 2019.

Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

SA Buechner, Chin-Yang Li, and WP Su. Leukemia cutis. a histopathologic study of 42 cases. *The American journal of dermatopathology*, 7(2):109–119, 1985.

Cheng Chen, Qi Dou, Hao Chen, Jing Qin, and Pheng Ann Heng. Unsupervised bidirectional cross-modality adaptation via deeply synergistic image and feature alignment for medical image segmentation. *IEEE transactions on medical imaging*, 39(7):2494–2505, 2020.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Barbara Deschler and Michael Lübbert. Acute myeloid leukemia: epidemiology and etiology. *Acute Leukemias*, pages 47–56, 2008.

Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. *Advances in Neural Information Processing Systems*, 32, 2019.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Tianmei Guo, Jiwen Dong, Henjian Li, and Yunxing Gao. Simple convolutional neural network on image classification. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pages 721–724. IEEE, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Christian Matek, Simone Schwarz, Karsten Spiekermann, and Carsten Marr. Human-level recognition of blast cells in acute myeloid leukaemia with convolutional neural networks. *Nature Machine Intelligence*, 1(11):538–544, 2019.

Christian Matek, Sebastian Krappe, Christian Münzenmayer, Torsten Haferlach, and Carsten Marr. Highly accurate differentiation of bone marrow cell morphologies using deep neural networks on a large image data set. *Blood, The Journal of the American Society of Hematology*, 138(20):1917–1927, 2021.

Surveillance Research Program NCI. Seer* explorer: An interactive website for seer cancer statistics, 2021.

Marc'Aurelio Ranzato and Martin Szummer. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*, pages 792–799, 2008.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Raheleh Salehi, Ario Sadafi, Armin Gruber, Peter Lienemann, Nassir Navab, Shadi Albarqouni, and Carsten Marr. Unsupervised cross-domain feature extraction for single blood cell image classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 739–748. Springer, 2022.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3752–3761, 2018.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

Yang Shen, Julia Wang, and Saket Navlakha. A correspondence between normalization strategies in artificial and biological neural networks. *Neural Computation*, 33(12): 3179–3203, 2021.

Baochen Sun and Kate Saenko. Subspace distribution alignment for unsupervised domain adaptation. In *BMVC*, volume 4, pages 24–1, 2015.

Eric Topol. *Deep medicine: how artificial intelligence can make healthcare human again*. Hachette UK, 2019.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

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