Stacked Vector-Quantized Variational Autoencoders for Unsupervised Pretraining and Classification of White Blood Cells

David Yuchen Wang[a]

[a]*National University of Singapore, E1237235@u.nus.edu*

Abstract

This project proposes a Stacked Vector-Quantized Variational Autoencoder (SVQVAE) model for the task of classification on the WBC dataset of microscopic white blood cells. The proposed model utilizes a stack of Vector Quantized Variational Autoencoders to first perform high compression of the image space into a lower dimensional embedding space. This is done in an unsupervised manner by training the model on reconstruction on an unlabeled dataset. Then, the classification task is performed from the embedding space by further finetuning with a labeled dataset. The model is able to take advantage of unlabeled data for pretraining through learning effective reconstruction. Further experiments show the model also has the capabilities to learn from a very small subset of data, as well as enhancing performance when additional masked data is provided.

1. Introduction

Many medical diagnoses and treatments rely on the accurate detection and identification of various blood cells. In particular, the ability to differentiate and count the different white blood types play an important role in the decisions related to the treatment of disease [6]. In practice, manual and automated techniques are utilized to classify blood cells, including microscopic evaluation, automated hematology devices, and electro-optical analyzers [6]. However, these methods are often expensive and require specialized personnel to operate. In recent years, deep learning has emerged as a way to aid with medical image processing [14], and can provide a faster and cheaper alternative to blood cell identification tasks. In order to train high performing deep learning models, an important element is the availability of large amounts of high quality data. However, it is often difficult to access high quality labeled data for medical diagnosis [2]. Unlabeled data, on the other hand, is more accessible and can be easier to obtain. With these considerations, this project seeks to build a model that would be able to utilize large amounts of unlabeled data to condition more efficient finetuning on small labeled datasets. Another exploration is if the addition of extra information (through the use of segmentation masks) can aid the model in more effective learning.

1.1. Datasets

The pRCC [4] and CAM16 [8] datasets are used as unlabled data to pretrain the model. Afterwards, the Raabin-WBC [6] is used to further finetune the model and evaluate it's classfication accuracy on the 5 white blood cell types, Basophil, Eosinophil, Lymphocyte, Monocyte, and Neurophil. Furthermore, for the WBC dataset, annotated masks are given for a small subset of the images. This is included in an additional finetuning step to analyze any performance gains to the model when given this additional information. The whole dataset (WBC100) is further divided into the sub-datasets WBC50, WBC10, and WBC1, which are separately used to train the model. Details of the datasets [7] are shown in Figure 1 and Figure 2.

Class	WBC_100			WBC_50		WBC_10		WBC_1	
	Train		Validation	Train		Train		Train	
	data	mask	data	data	mask	data	mask	data	mask
Basophil	176	17	36	88	8	17	1	1	0
Eosinophils	618	61	126	309	30	61	6	6	0
Lymphocyte	2015	201	412	1007	100	201	20	20	2
Monocyte	466	46	95	233	23	46	4	4	0
Neutrophil	5172	517	1059	2586	258	517	51	51	5
Total#	8447	842	1728	4223	419	842	82	82	7

Figure 1: Statistics for each of the WBC100, WBC50, WBC10, and WBC1 datasets. All images are 575x575x3 pixels.

Class	CAM16				pRCC
	Train		Validation	Test	Train
	data	mask	data	data	data
normal	379	37	54	108	1419
tumor	378	37	54	108	
Total#	757	74	108	216	1419

Figure 2: Statistics for the CAM16 and pRCC datasets. CAM16 includes images of size 384x384x3 pixels and pRCC images are 2000x2000x3 pixels

1.2. Data Augmentation

In order to faciliate more effective learning, the data is augmented before the pretraining and finetuning step. Due to the variable sized images across all datasets, a standard size was chosen as 512x512x3 pixels for all images. All images in CAM16 are upscaled, all images in WBC are downscaled, and all images in pRCC are cropped. The motivation behind cropping was to preserve as much information as possible of the high fidelity images in pRCC dataset for use in the pretraining step. The mean and standard deviation of each channel of the image was computed for each dataset (WBC100, CAM16,

pRCC), and all images within the dataset were normalized before training. In the pretraining step, all images from pRCC and CAM16 are merged into a single dataset and fed to the model in batches. A random rotation between ±180 degrees was applied to all images. Images from pRCC were sampled with a 4x greater probability than CAM16, since the effective size of each pRCC image was much greater. By introducing this factor of 4, the hope was to sample more of the features within the high fidelity pRCC data.

For finetuning, all images from WBC were also augmented using a random rotation between ±180, while the test set only received normalization.

For finetuning with masks, the entire WBC dataset was used, and all images with masks were overlayed such that the original image pixels remained within the mask while all masked out areas were replaced with normalized random noise. An example of this augmentation is shown in Figure 3

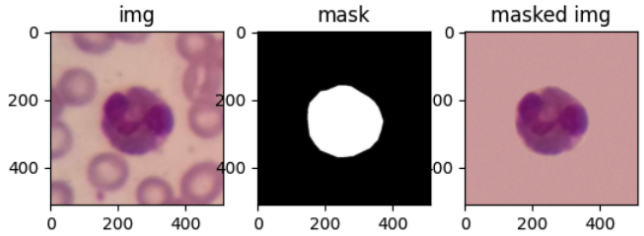

Figure 3: The masking data augmentation

2. Model Description

In order to effectively perform classification on images, deep learning models usually try to learn the underlying features within each image. This is often done through methods of first reducing the dimensionality of the image. A proven method for dimensionality reduction while still preserving features is the auto-encoder [1], which utilizes a neural network to learn a compact representation of the higher dimensional feature space. A scan of recent literature shows the success of many types of auto-encoders, with one of the most widely used being the Vector Quantized Variational Autoencoder (VQ-VAE) [10]. It's success has been seen in instances such as state of the art latent diffusion models, such as Stable Diffusion [11] and Wuerstchen [9]. Both of these models utilize a variant of the VQ-VAE to first compress an image into a latent space, and later reconstruct an image from the latent representation. The Wuerstchen [9] model takes this a step further, by first utilizing a VQGAN [3] to compress the image into a latent space, then utilizes a diffusion UNET to compress the latent into an even smaller dimensional space. The model is then able to train a text-guided generative model to generate latents at the smallest level, massively reducing training time. This latent can then be decoded twice to obtain a high fidelity image.

Motivated by this, this projects explores the possibility of utilizing a stack of VQ-VAEs to attempt to encode images to a very small latent space, which is many orders times smaller than the image space. If such a set of features can be learned effectively, then classification can be performed at the latent level, which will present a large decrease in the amount of redundant features. A non-linear, feed forward network can then be trained to classify the latent features. In this project, 3 VQ-VAEs are utilized in conjunction to encode a 512x512x3 image into a latent representation of size 4x8x8, and the model will be termed as a Stacked Vector Quantized Variational Autoencoder (SVQ-VAE).

2.1. VQ-VAE

The VQ-VAE [10] differs from traditional VAEs through the use of a codebook for vector quantization. The encoder $\mathcal{E}$ performs a non-linear mapping of the input space $\mathbf{x}$ to a latent space $\mathbf{z} = \mathcal{E}(\mathbf{x})$. The model also contains a learned codebook of K latent vectors of fixed size, $\mathbf{e}_k, k \in 1, ..., K$. The encoded vector is then quantized by taking the value of the closest vector in the codebook

$$\text{Quantize}[\mathcal{E}(\mathbf{x})] = \mathbf{e}_k, \quad \text{where } k = \text{argmin}_j \|\mathcal{E}(\mathbf{x}) - \mathbf{e}_j\| \quad (1)$$

The decoder $\mathcal{D}$ then decodes the latent vector back to the original space $\tilde{\mathbf{x}} = \mathcal{D}(\mathbf{e})$. The encoder and decoder are optimized via the objective function given in [10], where sg[·] refers to a stop gradient to the argument, $\mathbf{e}$ is the closest quantization vector in the codebook, and β is a hyperparameter:

$$\mathcal{L}(\mathbf{x}, \mathcal{D}(\mathbf{e})) = \|\mathbf{x} - \mathcal{D}(\mathbf{e})\|_2^2 + \beta \left\|\text{sg}[\mathbf{e}] - \mathcal{E}(\mathbf{x})\right\|_2^2 \quad (2)$$

The codebook vectors are learned via an exponential moving average, as described in [10]. The first term is defined as the reconstruction loss, which measures the mean squared error between the decoded image and the original image. The the second term is defined as the latent loss, which measures the mean squared distance between the each encoding to its closest quantization vector. The code implementation of the VQ-VAE in the project adapts the open-sourced implementation provided by [12].

2.2. Architecture

Building off of the VQ-VAE architecture, this project combines multiple VQ-VAEs into a Stacked Vector Quantized Variational Autoencoder (SVQVAE) model. Three VQ-VAEs are used, and thus define 3 levels of the model. The architecture of each VQ-VAE is composed of an encoder and a decoder, which encode at two differing hierachies. The bottom encoder is composed of 3 convolutional blocks with kernels of size 4, 4, 3, stride of 2, 2, 1, and padding of 1, 1, 1 respectively. This is followed by multiple residual blocks each consisting of a convolutional layer with kernel size 3, padding 1, and a convolutional layer with kernel size 1. The bottom encoder encodes an input of shape C × N × N into an embedding of shape k × N/4 × N/4, where C is the number of channels of the input and k is the dimension of each quantization vector in the codebook. The top encoder further reduces the dimensionality by applying another 2 convolutions with kernel sizes 4, 3, stride of 2, 1, and padding of 1, 1 respectively to the bottom embedding, to

encode a k × N/8 × N/8 top embedding block. The use of two layers of differing embedding sizes enhances the model's ability to capture different levels of detail at each of the two levels [10]. For implementation of SVQVAE, the two embeddings are then merged, by first upsampling the bottom top embedding via a single transposed convolution with kernel size 4, stride of 2, and padding of 1, then stacked such that the final embedding is of size 2k × N/4 × N/4.

In the decoding step, the embedded encoding is passed through a series of residual blocks and transposed convolutions in the decoder to obtain a reconstruction of the original input.

The SVQVAE stacks three of these VQ-VAEs together, such that the embedding from one layer is then fed to the encoder of the next VQ-VAE, and then the next. For the datasets in the project, the SVQVAE encodes an image of size 512×512×3 into an embedding of 2k × 8 × 8.

The model architecture is shown in Figure 4.

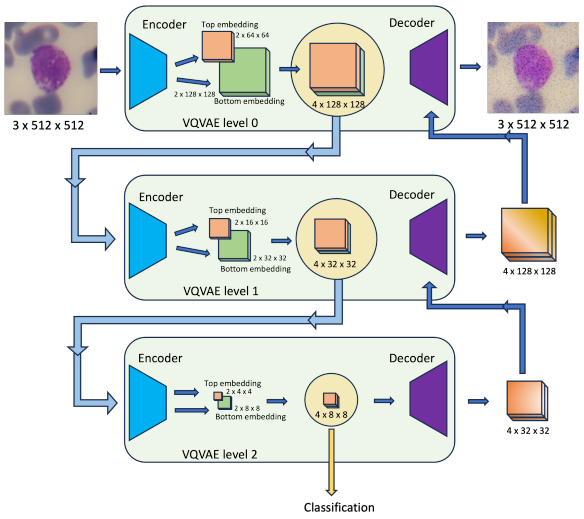

Figure 4: The SVQVAE architecture, with an input image dimension of 512x512x3 and a codebook vector dimension of 2.

3. Experimentation and Results

Various architectures and hyperparameters of the model were tested first through pretraining on the unlabeled dataset. Within the scope of these tests, the overall best model in terms of perfomance and size was selected for continued training on WBC datasets.

3.1. Pretraining

First, the model was pretrained from scratch on the unlabeled, joint pRCC and CAM16 dataset. The best performing SVQVAE model consists of 3 VQ-VAEs, each with 128 channels in the encoder/decoder, 2 residual blocks each in the encoder/decoder, 32 channels for each residual block, codebook vectors of dimension 2, 512 total codebook vectors, and decay of 0.99 for the codebook exponential update. The total number of parameters for this model is 3054644.

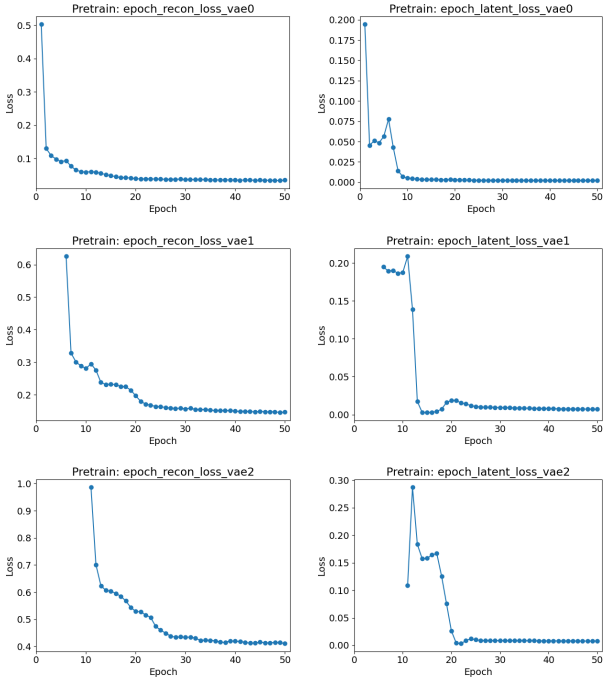

Figure 5: Plots showing the latent loss and reconstruction loss for each level of the SVQVAE during finetuning

During the pretraining stage, the loss objective for VQ-VAE (equation 2) is used to train each layer, with β set at 0.25. The model is trained in a staggered approach, using a staggering epoch of 5. First the reconstruction loss and latent loss is only calculated for the first VQ-VAE layer, and the weighted sum is back-propagated through the network. After training for the staggered number of epochs, the model is deepened by one layer, and the corresponding reconstruction and latent loss is calculated for that layer and back-propagated along with the reconstruction and latent losses for the previous layers. The pretraining reconstruction and latent losses for each layer of SVQ-VAE is shown in Figure 5.

The model was pretrained with a batch size of 24 for 50 epochs. AdamW optimizer is used with a learning rate of $1.5e^{-4}$, weight decay of 0.05, and betas (0.9, 0.95). A cosine annealing scheduler is used. These hyperparameters were chosen from reference to [5].

The model checkpoint after 50 is saved and used for the following training steps. The reconstructions achieved by the model from each level is visualized in Figure 6 for a random sample from each of the three datasets.

3.2. Finetuning

To enable the model to perform supervised classification, the smallest latent embedding from SVQVAE is flattened and fed through 3 feed forward layers, and then through a softmax. The features space is first fed through a layer of 2 × the flattened embedding dimension, then to a linear layer of size 64 and finally to a linear layer of size 5.

At each step, 3 separate loss functions are computed, the classification loss, which is the cross entropy between the predicted

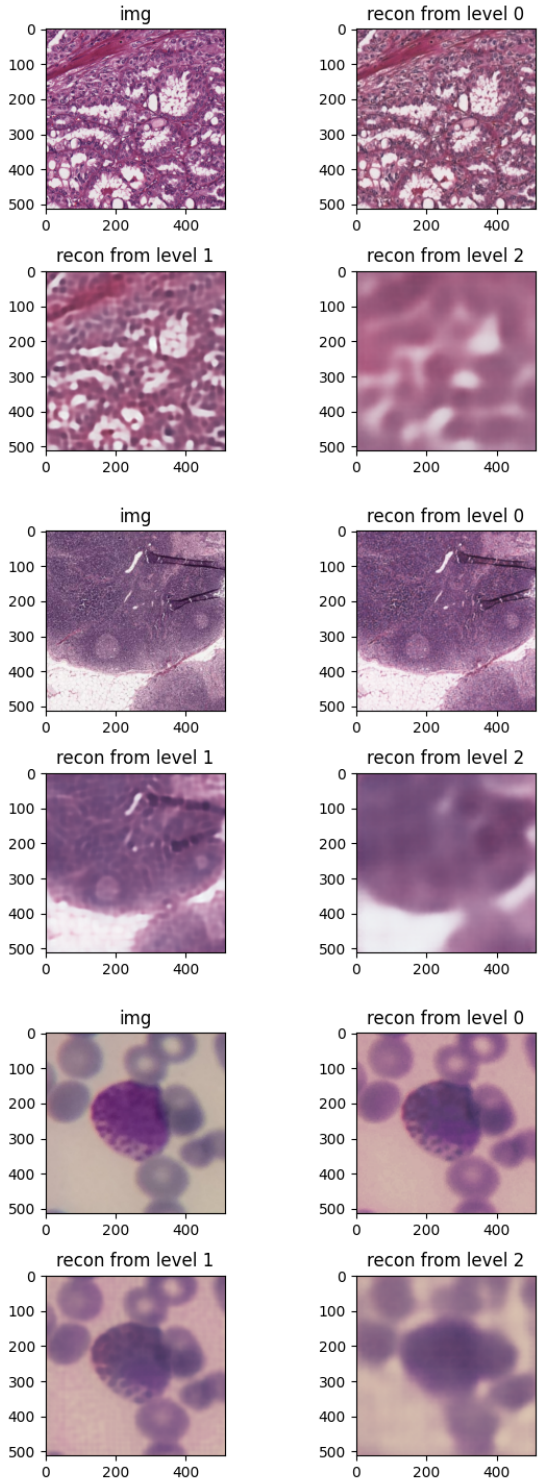

Figure 6: Visualization of the reconstructions produced by the model on a random image from each dataset after 50 epochs of pretraining on CAM16 and pRCC datasets. The random image is taken from CAM16 (top), pRCC (middle), and WBC100 (bottom)

labels and the ground truth, as well as the reconstruction and latent losses, which are defined for the smallest VQ-VAE level. The classification loss is backpropagated every step, while the reconstruction and latent losses are backpropagated every N epochs, with N = 2. Through experimentation over a few trials, this method proved to be the most effective in stablilizing and converging to a good training accuracy. Experiments were also carried out where all the losses were optimized each epoch, which resulted in slower convergence, as well as only optimizing classification loss, which resulted in the model being unable to learn.

Finetuning is performed on all subsets of the WBC dataset from the weights of the pretrained model. For finetuning, each sub-dataset (WBC100, WBC50, WBC10, WBC1) and classes are weighed such that there is an equal probability of sampling from each class. Finetuning was performed at 100 epochs with a batch size of 24. AdamW optimizer is used with a learning rate of $1.5e^{-4}$, weight decay of 0.05, and betas (0.9, 0.95), without a scheduler.

The resulting statistics and loss are summarized for WBC100 (Figure 9), WBC50 (Figure 10), WBC10 (Figure 11), and WBC1 (Figure 12).

The best training and testing accuracies of the finetuned model on each sub-dataset are summarized in Figure 7.

	Dataset			
	WBC 100	WBC 50	WBC 10	WBC 1
Training Accuracy (best)	97.7%	97.0%	97.9%	98.7%
Testing Accuracy (best)	97.3%	95.8%	94.4%	88.4%

Figure 7: Comparisons of the training and testing accuracies for each of WBC100, WBC50, WBC10, and WBC1 datasets. All were trained for 100 epochs and the best accuracy taken.

3.3. Additional Information

Given that masks are also provided for a portion of the WBC dataset, they are additionally utilized to see if model performance can be improved. All 4 datasets are finetuned from the pretraining checkpoint, this time on the mask dataset. At each step, the same loss is calcuated as in the finetuning step, except with the addition of a feature loss, which computes the mean square error between the latent embedding of the input image with masking $\mathbf{x}_m$, and without masking $\mathbf{x}$.

$$\mathcal{L}_{\text{feature}} = \text{MSE}[\mathcal{E}(\mathbf{x}_m), \mathcal{E}(\mathbf{x})] \tag{3}$$

The motoviation behind this loss is to give information to the network that only the features within the mask area are important towards classification.

The model is finetuned given the masks from each dataset with the same hyperparameters as the non-mask finetuning. The resulting best training and testing accuracies are summarized in Figure 8.

	Dataset (with mask)			
	WBC 100	WBC 50	WBC 10	WBC 1
Training Accuracy (best)	97.1%	97.5%	98.1%	100.0%
Testing Accuracy (best)	96.1%	96.5%	94.0%	90.7%

Figure 8: Comparisons of the training and testing accuracies for each of WBC50, WBC10, and WBC1 datasets. All were trained for 100 epochs with masking in the dataset and use of the feature loss. The best accuracy is taken.

4. Discussion and Future Work

This project investigated the use of a Stacked Vector Quantized Variational Autoencoder for pretraining and classification of the WBC white blood cell dataset. Results from the experiments show that the method is quite effective, with the capability to achieve above 97% accuracy on the WBC test set. The model is able to effectively leverage large amounts of unlabeled data to first pretrain, which allow it to be effective in self-supervised learning. Even with a high compression ratio of 3072 from image space to the smallest latent space ($512 \times 512 \times 3$ to $8 \times 8 \times 4$), the model is still able to extract meaningful features from the compressed space to accurately perform classification. For the scope of this project, extensive hyperparameter tuning was not performed, but it is very likely that model performance can improve with slight optimization of hyperparameters.

The model also shows good capability to learn on very small amounts of data without overfitting, as shown in the experiments with the WBC10 and WBC1 datasets. Especially for the case of WBC1, as seen from the confusion matrix in Figure 12, even with only a single example of Basophil in the training set, the model is able to generalize well and perform well on the test dataset. A hypothesis is that by training the model to perform reconstruction from such a small latent space, the model must learn a good set of general features, which also prevents overfitting in the classfication task. Thus, when these two losses are jointly optimized, the model can achive good performance from very few samples.

When additional information is introduced for the model, a slight improvement is noticed. This is seen from the comparison of Figure 7 and Figure 8. Training accuracy is higher for training on the masked data for WBC1, WBC10, and WBC50 datasets, while testing accuracy is higher for WBC50 and WBC1 dataset. This shows potential for the method to be further explored, such as designing a more sophisticated feature loss, such as comparing the probability distributions of the latent embeddings through metrics such as KL divergence [13]. Further optimizations are likely to be achieved through reweighing the loss weights and optimization of the losses following different schedules.

5. Summary and conclusions

Through experiments with the WBC, CAM16, and pRCC datasets, the SVQVAE model shows the potential for performing fast and accurate identification of images in the medical domain. The strengths of the model include its very small size of only 3.1M parameters, its capabilities to utilize large amounts of unlabeled data to perform pretraining, as well as the potential to optimize its performance through additional, currated data.

References

[1] D. Bank, N. Koenigstein, and R. Giryes. *Autoencoders*. 2021. arXiv: 2003.05991 [cs.LG].

[2] M. Benčević, M. Habijan, I. Galić, and A. Pizurica. *Self-Supervised Learning as a Means To Reduce the Need for Labeled Data in Medical Image Analysis*. 2022. arXiv: 2206.00344 [cs.CV].

[3] P. Esser, R. Rombach, and B. Ommer. *Taming Transformers for High-Resolution Image Synthesis*. 2021. arXiv: 2012.09841 [cs.CV].

[4] Z. Gao, B. Hong, X. Zhang, Y. Li, C. Jia, J. Wu, C. Wang, D. Meng, and C. Li. *Instance-based Vision Transformer for Subtyping of Papillary Renal Cell Carcinoma in Histopathological Image*. 2021. arXiv: 2106.12265 [cs.CV].

[5] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. *Masked Autoencoders Are Scalable Vision Learners*. 2021. arXiv: 2111.06377 [cs.CV].

[6] Z. M. Kouzehkanan, S. Saghari, S. Tavakoli, P. Rostami, M. Abaszadeh, F. Mirzadeh, E. S. Satlsar, M. Gheidishahran, F. Gorgi, S. Mohammadi, and R. Hosseini. "A large dataset of white blood cells containing cell locations and types, along with segmented nuclei and cytoplasm". In: *Scientific Reports* 12.1 (2022), p. 1123.

[7] L. H. Kuan. *CS5242*. School of Computing, National University of Singapore. 2023.

[8] G. Litjens, P. Bandi, B. Ehteshami Bejnordi, O. Geessink, M. Balkenhol, P. Bult, A. Halilovic, M. Hermsen, R. van de Loo, R. Vogels, Q. Manson, N. Stathonikos, A. Baidoshvili, P. van Diest, C. Wauters, M. van Dijk, and J. van der Laak. "1399 H&E-stained sentinel lymph node sections of breast cancer patients: the CAMELYON dataset". In: *Gigascience* 7.6 (2018). DOI: 10.1093/gigascience/giy065.

[9] P. Pernias, D. Rampas, M. L. Richter, C. J. Pal, and M. Aubreville. *Wuerstchen: An Efficient Architecture for Large-Scale Text-to-Image Diffusion Models*. 2023. arXiv: 2306.00637 [cs.CV].

[10] A. Razavi, A. van den Oord, and O. Vinyals. *Generating Diverse High-Fidelity Images with VQ-VAE-2*. 2019. arXiv: 1906.00446 [cs.LG].

[11] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV].

[12] rosinality. *vq-vae-2-pytorch*. https://github.com/rosinality/vq-vae-2-pytorch. GitHub repository. 2023.

[13] J. Shlens. *Notes on Kullback-Leibler Divergence and Likelihood*. 2014. arXiv: 1404.2000 [cs.IT].

[14] B. Sistaninejhad, H. Rasi, and P. Nayeri. "A Review Paper about Deep Learning for Medical Image Analysis". In: *Computational and Mathematical Methods in Medicine* 2023 (2023). Article ID 7091301, pp. 1–10. DOI: 10.1155/2023/7091301.

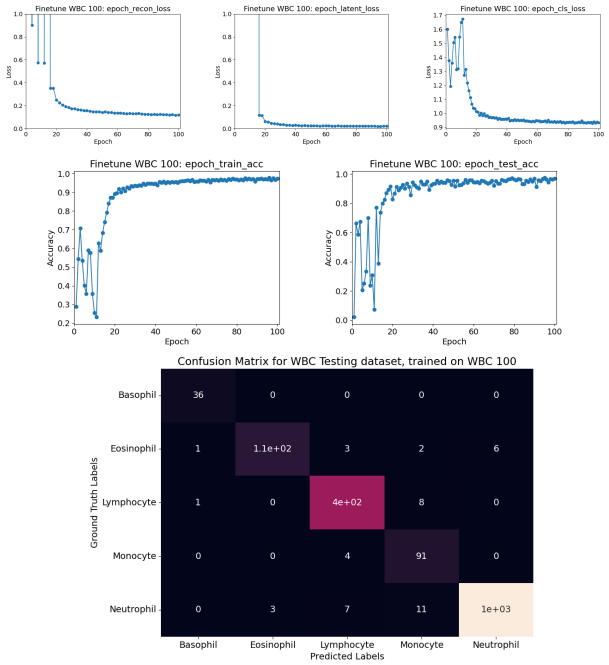

Figure 9: Statistics for finetuning of WBC100. Tops shows the reconstruction loss (left), latent loss (middle), and classification loss (right). Middle shows training accuracy (left) and testing accuracy (right). Bottom shows the confusion matrix computed for the WBC test set on the best checkpoint

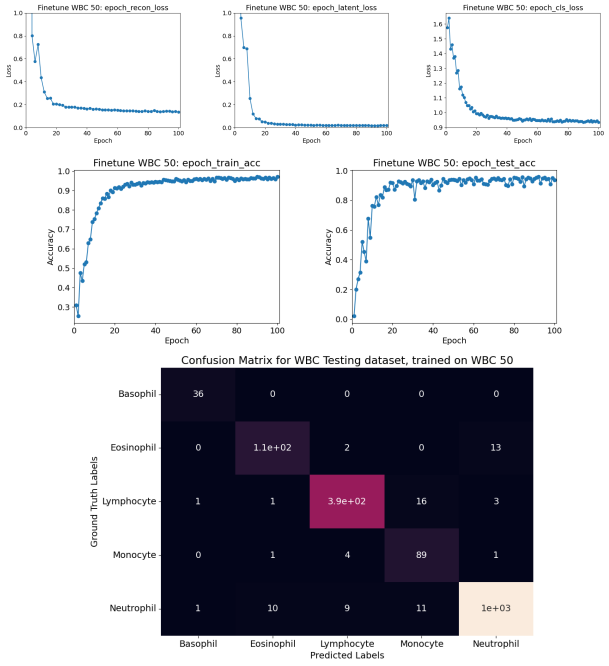

Figure 10: Statistics for finetuning of WBC50. Tops shows the reconstruction loss (left), latent loss (middle), and classification loss (right). Middle shows training accuracy (left) and testing accuracy (right). Bottom shows the confusion matrix computed for the WBC test set on the best checkpoint

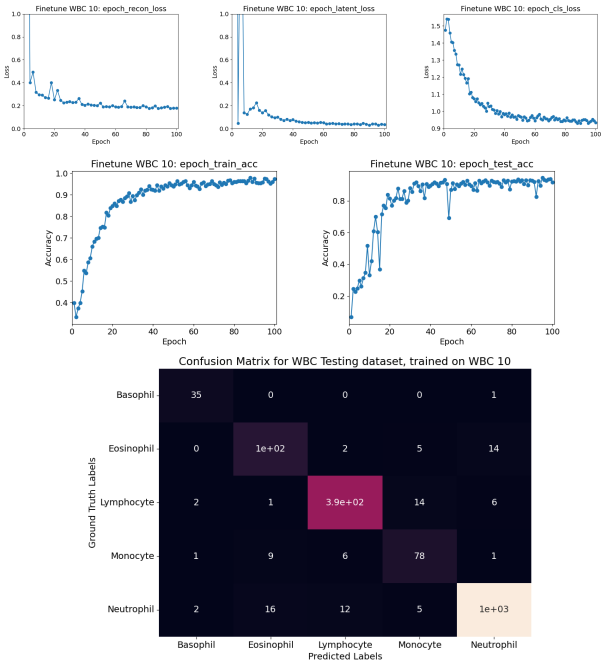

Figure 11: Statistics for finetuning of WBC10. Tops shows the reconstruction loss (left), latent loss (middle), and classification loss (right). Middle shows training accuracy (left) and testing accuracy (right). Bottom shows the confusion matrix computed for the WBC test set on the best checkpoint

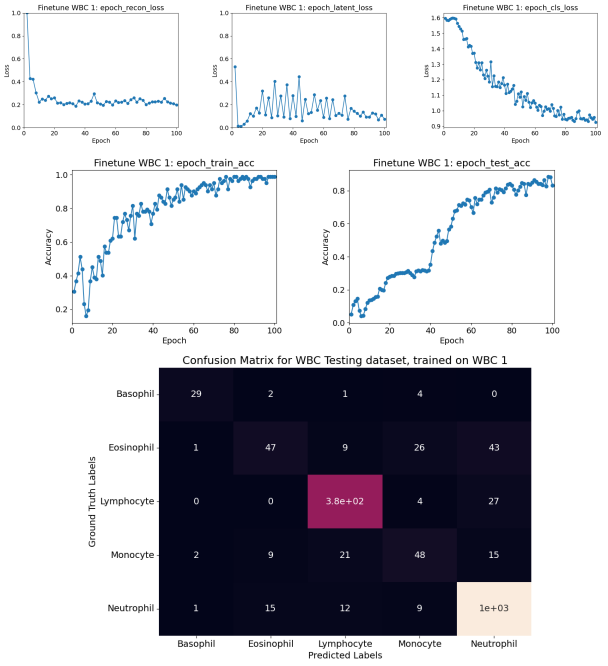

Figure 12: Statistics for finetuning of WBC1. Tops shows the reconstruction loss (left), latent loss (middle), and classification loss (right). Middle shows training accuracy (left) and testing accuracy (right). Bottom shows the confusion matrix computed for the WBC test set on the best checkpoint