Self-supervised learning forclassification of anatomic featureson kidney biopsy whole slide images

Long, J.

Self-supervised learning for classification of anatomic features on kidney biopsy whole slide images

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
STATISTICS&DATA SCIENCE

Author :	Jingmin Long
Student ID :	s2501368
Supervisor :	Prof.Dr.Marta Fiocco,
	Prof.Dr.Fons Verbeek

Leiden, The Netherlands, August 20, 2022

Self-supervised learning for classification of anatomic features on kidney biopsy whole slide images

Jingmin Long

Snellius Building, Leiden Univeristy
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

August 20, 2022

Abstract

The technique of whole slide imaging (WSI) boosts the application of deep learning in medical imaging analysis and computational pathology. However, fully supervised learning stucks into bottlenecks due to the heavy reliance on manual annotations, which requires specific expertise and expensive cost. Self-supervised learning would be a potential solution, which is supervised by the signals generated from itself. It has been proved to perform as well as supervised learning on ImageNet in classification tasks. Yet, its performance on medical image classification is unexplored. This study verifies the effectiveness of four self-supervised learning to detect anatomic structures on kidney biopsy WSI, including SimCLR, MoCo, SwAV and Barlow Twins. In the pretext-task, these self-supervised learning algorithms are trained in 500 epochs with the same backbone architecture, ResNet-50, which is initialized by the weights pre-trained on ImageNet correspondingly. The evaluation protocol is a semi-supervised linear classifier, implemented by using multi-nomial logistic regression. The results of the classification task show the features extracted by the four algorithms all achieve good accuracy scores, higher than 85% with only 10% labels. Among them, SwAV outperforms the other algorithms from the perspective of overview and each class. Through this study, self-supervised learning algorithms exhibit the potential for more complex tasks related to renal pathology.

Acknowledgements

This thesis reports my work over the past months, in which about self-supervised learning and medical image analysis, and concludes my Master of Science in Statistics&Data Science at Leiden University. Medical imaging analysis is what I am curious about but quite challenging for me as a student in statistics. I put my perseverance to fill my knowledge gap between the two research fields, like learning microscopic image analysis, Linux OS commands, and how to annotate kidney WSI. In every movement I found it immensely rewarding working this interdisciplinary project.

I would not have been able to finish this project without my great support team. First I'd like to express my sincerest appreciation to my supervisors Prof.Dr.Marta Fiocco and Prof.Dr.Fons Verbeek for your expertise and patience. Your experienced and insightful guidance keep me on the clear track and from deviating away during this individual research. Then I would like to thank Dr.Jesper Kers and Dr.Junling He. You infect me with your innovation and enthusiasm about computational pathology. I would also like to thank the staffs in the statistics department, who helped me a lot with the administrative troubles I faced.

Lastly, I am grateful to my friends and family. To my friends, your continuous encouragement and help bring warm and joy to my life. My family would be proud of the milestone, but this would not be passed without your company.

Jingmin Long

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Whole slide imaging (WSI), also termed as virtual microscopy, refers to creating a single high-resolution digital image by scanning a complete microscope glass slide. [16] The emerging technique of WSI designs to emulate conventional light microscopy in a computer-generated manner, and enables the adoption of digital image processing in computational pathology tasks like objective prognosis, diagnosis and therapeutic response prediction. An impressive generic approach to tackle these complex tasks is deep learning, having advantages like automatic feature generation and scalability, especially beneficial for the high resolution WSI (e.g., $> 10{,}000 \times 10{,}000$ pixels on $40\times$). One of the typical categories in deep learning is supervised learning that refers to learn a function that can map an input object to a desired output supervised by given labels. However, concerning clinical research field, supervised learning hits the bottleneck in computer vision tasks, because it depends on exhaustive manual annotations of medical images, but the annotations demand specific expertise and laborious efforts on gigapixel high-resolution WSI.

Rather than being fed with input-output pairs, self-supervised learning (SSL) enables models to be supervised simply by data itself without massive number of labels, In light of how human learn to classify objects, self-supervised learning is introduced to establish generalized knowledge for deep learning models which approximates to 'common sense' for human, so that the machines can recognize the world without requiring massive amounts of teaching on every single task. The general technique is to build pre-training predictive models, predicting an unobserved and shuffled part of an input from an observed and ordered part of an input. In the tasks about natural language processing and natural image classification, the models pre-trained with self-supervised learning have demonstrated higher performance than the models merely trained in supervised learning manners.

Yet self-supervised learning is not promised to preserve the high performance on kidney biopsy WSI dataset, which is less heterogeneous in diverse classes than natural images. When the kidney is diseased, the pathological changes such as fibrosis and atrophy would derive many extra variation in patterns, making machines hard to distinct them. This thesis will examine if the features pre-trained on self-supervised learning could reduce the dependence on labels while keep good performance on downstream tasks. Four self-supervised learning algorithms will be investigated, including Simple framework for Contrastive learning (SimCLR), Momentum Contrastive learning (MoCo), Swapping Assignments between multiple Views of the same image (SwAV) and Barlow Twins (BarlowTwins). The downstream task is set as kidney anatomic structure classification, implemented by multinomial logistic regression. Varied proportion of labels are used to train the classification models, from 1%, 10%, 30%,..., to 90%, and the remaining data are used as the test set. We use the measurement metrics on the test set to evaluate and compare them, including accuracy scores of the whole model and precision, recall and f1 scores of each class. The results reveal the algorithms have excellent scores and accuracy even trained with simply 10% labels. Through the comparison of the four algorithms, we notice SwAV outperforms other algorithms from the perspective of every class. We could conclude that self-supervised learning would promote the leverage of the large amount of unlabelled medical imaging data. It has great potential to be proceeded with more complicated computer vision tasks pertaining to renal pathology based on kidney biopsy WSI.

1.2 Research Questions

The thesis examines the performance of self-supervised learning algorithms on classifying anatomic structures in kidney biopsy WSI. In particular, we focus on the performance of SimCLR, MoCo, SwAV and BarlowTwins on classifying of glomeruli, vessels and tubules. The research questions are comprised three parts. First, we generate feature representations for each sample. To intuit about the patterns, we apply t-distributed stochastic neighbor embedding (t-SNE) to visualize the features and check if there are clusters of specific structures. Second, we use a linear classifier implemented by multinomial logistic regression to evaluate the features representations. For the linear classifiers, the percentage of labels used for training is varied. We want to figure out the proportion of labels that is the most effective. Third, we compare the overall performance of the four algorithms and verify whether any of them are suitable for recognizing specific structures. We discuss the feasibility of assembling the algorithms and the possibility of proceeding to more complicated tasks like object detection and instance segmentation.

1.3 Thesis Structure

The remaining sections of the thesis are arranged as follows: in chapter 2, the necessary background knowledge about self-supervised learning, visualization and classification models is provided; in chapter 3, the experiments are elaborated, followed by their results in chapter 4; in chapter 5, conclusions and discussions are drawn.

Chapter 2

Review of Methods

This chapter will provide the background knowledge for this project. Before covering the self-supervised learning algorithms, ResNet-50 is clarified in section 2.1. ResNet-50 is an important pre-requisite knowledge, used as the backbone of self-supervised architecture in this study. What follows is an in-depth overview of the four self-supervised learning investigated in this study in section 2.2. In section 2.3, t-SNE, the technology that suitable for visualizing high-dimensional features is introduced. At last, section 2.4 explains multinomial logistic regression, the classification model that is applied to evaluate the features extract by the self-supervised learning algorithms.

2.1 Backbone: ResNet-50

ResNet, abbreviated for 'Residual Network' introduced in 2015 by Kaiming He et al. [9] In the beginning, they speculate that additional layers in the deep neural networks should produce growing accuracy and performance, since more complicated features should be learned steadily with the additional parameters. Whereas, the fact is counterintuitive in the experiments of plain networks, that the performance of plain networks degrades after stacking more layers, due to the problems of gradients vanishing or gradients explosion. They address the degradation problem by introducing a deep residual learning framework, instead of counting on each few stacked layers directly fits a desired underlying mapping. Expressed in formula, the desired underlying mapping is denoted as $H(x)$ and the stacked nonlinear layers fitting another residual mapping is denoted as $F(x) = H(x) - x$. So the original mapping is recast into $F(x) + x$, realized by feedforward neural networks with 'shortcut connections', see Figure 2.1. [9] These blocks of residual learning overcome the problem of

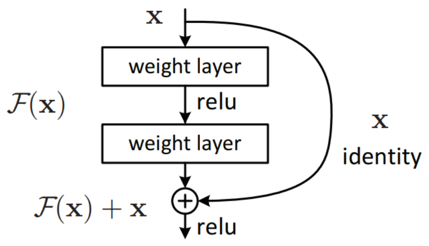

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.1: *A building block of residual learning [9]*

gradients vanishing by allowing this alternate shortcut path for the gradient to flow through, making it reasonable to construct deeper networks for better performance.

ResNet-50 is a 50-layer deep ResNet, see its architecture in Figure 2.2. From left to right, the elements are explained as follows:

- An input image with size 224×224.
- A 1-layer convolution with 64 kernels and a max pooling layer; each kernel is in size 7×7 and with stride 2.
- One convolution block repeated 3 times; each block encloses 3-layer convolution: 1×1,64 kernels, following 3×3,64 kernels and at last 1×1,256 kernels, all with stride 2.
- One convolution block repeated 4 times; each block encloses 128, 128, and 512 kernels, and the other setting are same with the blocks above.
- One convolution block repeated 6 times; each block encloses 256, 256, and 1024 kernels, and the other setting are same with the blocks above.
- One convolution block repeated 3 times; each block encloses 512, 512, and 2048 kernels, and the other setting are same with the blocks above.
- An average pooling and fully connected layer.

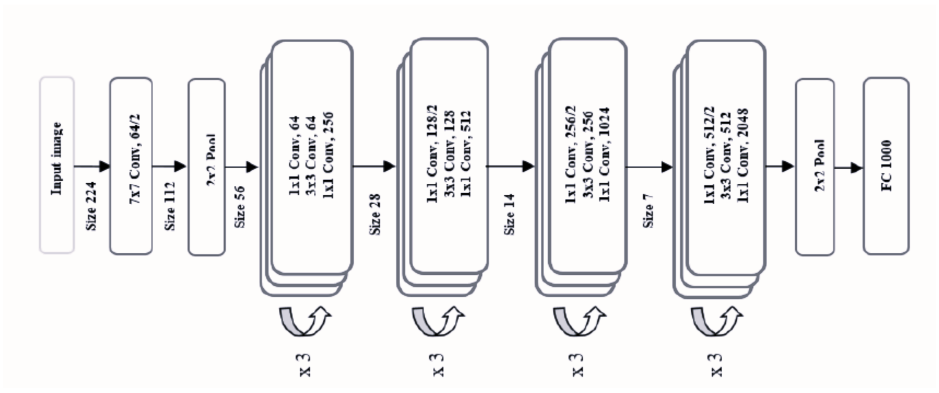


Figure 2.2: *The architecture of ResNet-50 [10]*

2.2 Head: MLP

Another structure used as the projection head of the self-supervised learning is multilayer perceptron, abbreviated as MLP. A MLP is a fully connected artificial neural network, comprised of one input layer, one output layer and many hidden layers, see Figure 2.3. All the hidden layers are followed by rectified linear unit (ReLU), an activation function defined as the positive part of its argument: $f(x) = max(0, x)$. MLP head could lower the rank of the feature matrix extracted by backbone, which is beneficial to calculate the contrastive loss, but poorer to represent the input image.

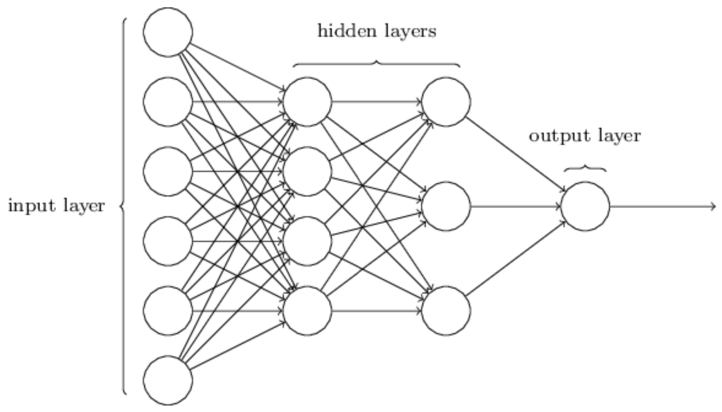


Figure 2.3: *A multilayer perceptron*

2.3 Self-supervised learning

In the pipeline of computer vision task, the task for pre-training is called 'pre-text task', and the task for fine-tuning is called 'downstream task'. Self-supervised learning gains popularity in pre-text task since its potential of reducing the cost of annotation on large-scale image datasets. Self-supervised learning establishes signals through predicting any distorted or unobserved parts or properties with the data itself, so as to evade the dependence on labels. One general architecture consists of an augmentation family, a siamese network, a head projector and objective loss function.

In this section, the four self-supervised architectures used in this study are clarified, including SimCLR, MoCo, SwAV, BarlowTwins. The four algorithms have various ways of generating pseudo labels: particularly, SimCLR and MoCo are based on contrastive instance learning, SwAV is based on contrastive clustering learning, Barlow Twins is based on redundancy reduction learning.

2.3.1 SimCLR

SimCLR, abbreviated from 'a Simple framework for Contrastive Study of visual Representations', is a typical algorithm of contrastive learning proposed by Chen, Ting, et al. in 2020, which learns representations by maximizing agreement between individually augmented views of the same data example via a contrastive loss in the latent space, see its framework in Figure 2.4. [12]. The architecture of contrastive learning contains four elements, an augmentation family denoted as T, a base encoder denoted as $f(\cdot)$, a projection head denoted as $g(\cdot)$ and a contrastive loss function. Suppose x is an input image, two random augmentations t, t sampled from augmentation family T are applied to generate two augmented images: $\tilde{x}_i$ and $\tilde{x}_i$. Next the augmented images are fed through the base encoder $f(\cdot)$ and the projection head $g(\cdot)$ successively. During training, the outputs given by the projection head $g(\cdot)$, z_i and z_j, are used to maximize agreement with contrastive loss. When the training is finished, the projection head $g(\cdot)$ as well as its output z_i and z_j are abandoned. The base encoder $f(\cdot)$ are preserved to extract the features of input images. Each of the four elements are clarified as below:

- **An image augmentation family:** three simple augmentations are applied sequentially random cropping (with flip and resize), random

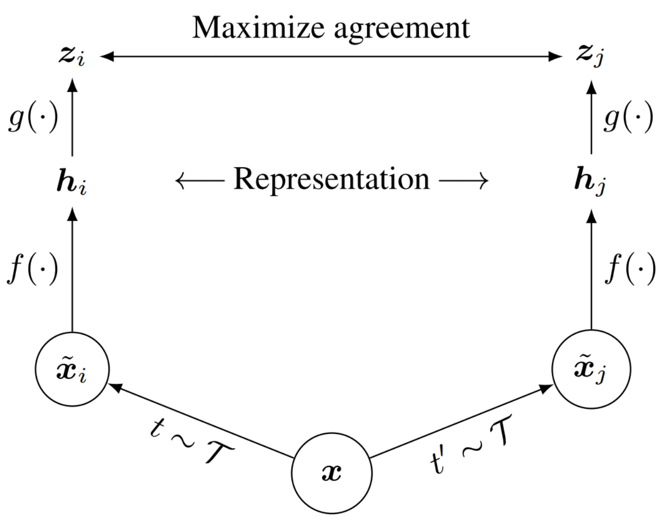


Figure 2.4: *A simple framework of contrastive learning [12]*

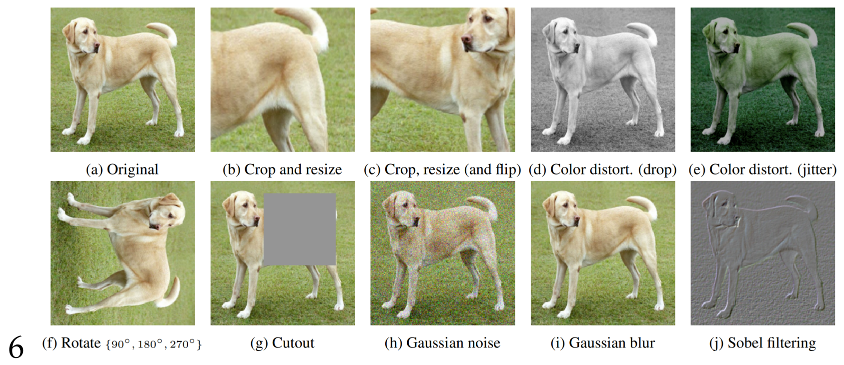

(a) Original
(b) Crop and resize
(c) Crop, resize (and flip)
(d) Color distort. (drop)
(e) Color distort. (jitter)
6
(f) Rotate {90°, 180°, 270°}
(g) Cutout
(h) Gaussian noise
(i) Gaussian blur
(j) Sobel filtering

Figure 2.5: *An example of the augmentation family [12]*

color distortions, and random Gaussian blur, see the example of the augmentations in Figure 2.5.

- **A base encoder:** ResNet-50 is adopted in this study for simplicity.
- **A projection head:** a MLP with one hidden layer.
- **The contrastive loss:** for each augmentation pairs, first a pairwise similarity is calculated; based on the similarity, the contrastive loss is converted. Suppose there are N input images in a batch, so there are N augmentations and $2N$ images in total. Expressed in formula, the pairwise similarity $s_{i,j}$ is:

$$s_{i,j} = \mathbf{z_i}^T \mathbf{z_j} / (\|\mathbf{z_i}\| \, \|\mathbf{z_j}\|) \tag{2.1}$$

where $i, j = 1, 2, ..., 2N$. Here the negative pairs are not sampled explicitly: all the other $2(N-1)$ samples are regarded as negative given a positive pair where $i = k$. Borrowing the idea from noise contrastive estimation (NCE) loss, the loss for a pair is calculated as the equation below and termed as NT-Xent (the normalized temperature-scaled cross entropy loss):

$$l_{i,j} = -log \frac{exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} 1_{k \neq i} exp(s_{i,k}/\tau)} \tag{2.2}$$

where $1_{k \neq i} \in 0, 1$ is an indicator function equal to 1 iff $k \neq i$, and τ is a temperature parameter. This contrastive loss function is also called infoNCE loss in later research. [3] Finally, the contrastive loss employed to update the SimCLR architecture is calculated as the average of the loss between positive pairs, expressed in the formula as:

$$L = \frac{1}{2N} \sum_{k=1}^{N} (l(2k-1, 2k) + l(2k, 2k-1)) \tag{2.3}$$

2.3.2 MoCo

MoCo, abbreviated from 'Momentum Contrast', is still a typical algorithm of contrastive learning proposed by He, Kaiming, et al in 2020. In MoCo, the global architecture appears akin to the classic contrastive learning, but the process of comparing positive and negative pairs is considered as a look-up problem of matching keys and query in dynamic dictionaries, see in Figure 2.6. Dynamic dictionary is in the sense that the keys are sampled randomly during the training of the encoder for keys. They hypothesize that more negative samples would result in better features, so they build a large dictionary while maintain the encoder for keys as consistent as possible. Driven by this motivation, they propose 'Momentum Contrast' with three essential tricks described below:

- **Dictionary as queue:** the keys queue up in the dictionary. Through the queue, the samples in the dictionary is progressively updated: when a new mini-batch are enqueued, the oldest are dequeued. This allows a large number of negative samples involved in the dictionary, but not constricted to the typical size of mini-batch. Additionally, this increases the consistency of the learning process, since the encoded keys of the oldest batch are outdated and the least consistent with the keys of the newest batches.

- **Momentum update:** though using queue enlarges the capacity of the dictionary, it impedes back propagation in the update of key encoder, because the gradients should propagate all samples in the queue. Copy the weights from query encoder to the key encoder is a naive but poor solution. They hypothesize that the abrupt change damage the consistency of the encoders. Thus, they propose 'a momentum update' to curb the damage. Denote the parameter of the key encoder f_k as θ_k and those of query encoder f_q as θ_q. θ_k is updated by:

$$\theta_k = m\theta_k + (1 - m)\theta_q \tag{2.4}$$

  Where $m \in [0, 1)$ is a momentum coefficient. With dictionary as queue, only θ_q can be updated by back propagation, thereby weighted average between θ_k and θ_q could improve the consistency. Verified by experiments, large m could evolve the key encoder progressively and make the queue more efficient, so $m = 0.999$ by default.

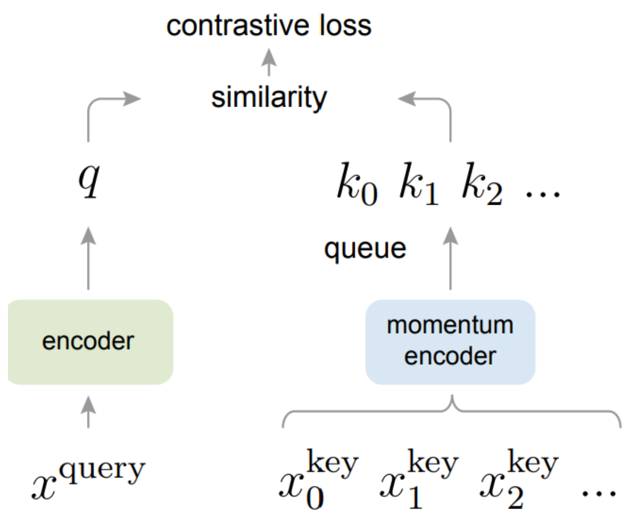


Figure 2.6: *The architecture of MoCo [5]*

The contrastive loss function still applies infoNCE loss, but it is adapted to compare between the keys and query. Mathematically, denote a query as q given by the query encoder $f_q(\cdot)$, a set of keys as $k_0, k_1, k_2, ...$ given by the key encoder $f_k(\cdot)$, the single key that matches q is k_+. The adapted infoNCE loss is:

$$L_q = -log\frac{exp(q \cdot k_+/\tau)}{\sum_{i=0}^{K} exp(q \cdot k_i/\tau)} \tag{2.5}$$

where τ is a temperature parameter. It is a softmax-like function of classifying q as k_+ over the sum of 1 positive and K negative samples.

After SimCLR released, MoCo published its second version 'MoCov2', which absorbs the tricks of SimCLR about augmentation and MLP head to improve its baseline. MoCov2 shares the same augmentation family, encoder and head projector architectures with SimCLR, but with a more effective batch learning style of key encoders, momentum contrast. The comparison of the two batch learning style is shown in Figure 2.7. We use MoCov2 in this study. The end-to-end batch learning style require higher time and memory cost than momentum contrast style.

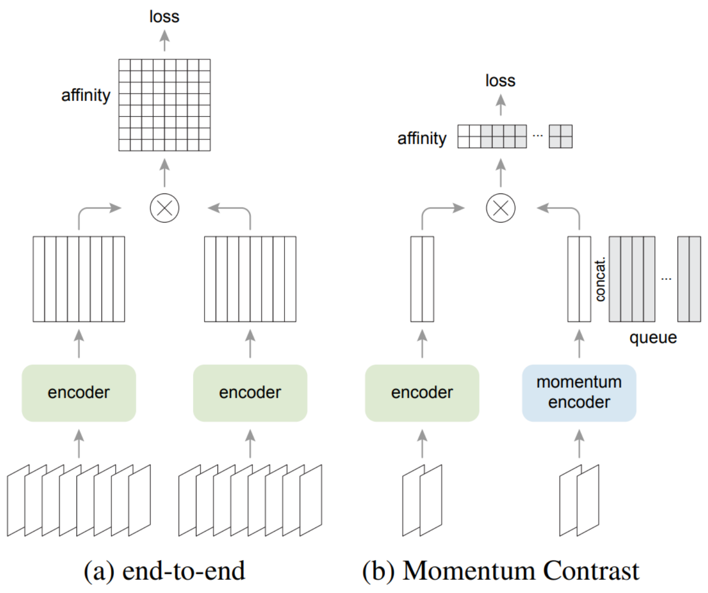


Figure 2.7: *A batching perspective of SimCLR and MoCov2 optimization mechanisms for contrastive learning. [6]*

2.3.3 SwAV

SwAV, abbreviated from 'Swapping Assignments between multiple Views of the same images', is one adapated contrastive learning algorithm proposed by Caron, Mathilde, et al in 2020. [13] Intead of pairwise feature comparisons in typical contrastive instance learning, SwAV learns features in an online clustering fashion which correlates the cluster assignments of multiple image views see Figure 2.8.

They defined a swapped prediction problem to solve the online clustering. Illustrated according to the right image in Figure 2.8, two augmen-

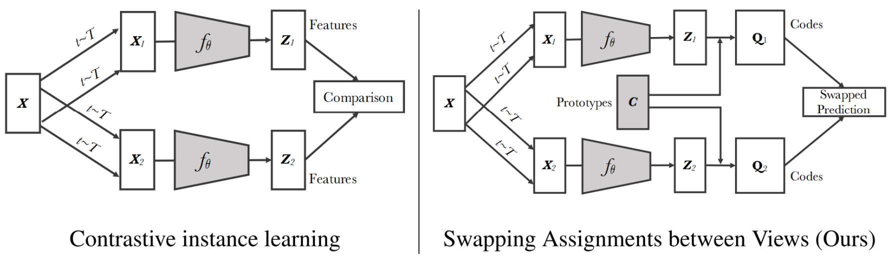


***Figure 2.8:** Contrastive instance learning (left) vs. SwAV (right) [13]*

tations of an image x_t and x_s are encoded as features z_t and z_s by encoders $f_{\theta(\cdot)}$. Then the features z_t and z_s are assigned to prototype vectors q_t and q_s by matching K prototypes $c_1, ..., c_K$ with the following loss function:

$$L(z_s, z_t) = l(z_s, q_t) + l(q_s, z_t) \tag{2.6}$$

where $l(q, z)$ represents the cross entropy loss between the codes and the probability activated by softmax function of the dot products of features z and prototype vectors c, expressed in the formula as below:

$$l(q_s, z_t) = -\sum_k q_s^{(k)} \log p_t^{(k)}, \text{ where } p_t^{(k)} = \frac{\exp(\frac{1}{\tau} z_t^T c_k)}{\sum_{k'} \exp(\frac{1}{\tau} z_t^T c_{k'})} \tag{2.7}$$

where τ is a temperature parameter. This loss function is jointly minimized with respect to the prototypes c and the parameters θ of encoder f_θ.

During the training, two vital innovation contribute to SwAV: online computation of codes and multi-crop augmentation strategy, explained as below:

- **Online computation of codes:** in online manner, the computation is carried within a mini-batch, but not passing over all the samples offline. Within a batch, the features are equally partitioned by the prototypes, which precludes trivial solutions that the features are all assigned to same codes. The number of prototypes is 3000 by default. When working with small batches, if the number of samples is smaller than the number of prototypes, the features in previous batches are retained to augment the size of samples for equal partition, but are discarded for training loss calculation.

  Mathematically, to map B feature vectors $Z = [z_1, ..., z_B]$ K into prototypes $C = [c_1, ..., c_K]$, the mapping are represented by codes $Q =$

$[q_1, ..., q_B]$, through maximizing the similarity between features Q and between the features Z and the prototypes C:

$$\max_{Q \in \mathcal{Q}} \mathbf{Tr}(Q^T C^T Z) + \varepsilon H(Q), \textit{ where } H(Q) = -\sum_{ij} Q_{ij} \log Q_{ij} \quad (2.8)$$

where H is an entropy function and ε is a smoothness parameter. By default, ε is setting at a low value because a strong entropy regularization parameter is likely to induce trivial solution. The prototypes C resemble the centers of each cluster. To enforce equal partition, the solution of matrix Q is modified to belong to the transportation polytope within mini-batches, see Figure 2.9, expressed in formula as:

$$\mathcal{Q} = \{Q \in \mathbb{R}_+^{K\times B} | Q\mathbf{1_B} = \frac{1}{K}\mathbf{1_K},\ Q^T\mathbf{1_K} = \frac{1}{B}\mathbf{1_B}\} \quad (2.9)$$

where $\mathbf{1_B}$, $\mathbf{1_K}$ denote the vectors of ones in dimension B and K. They

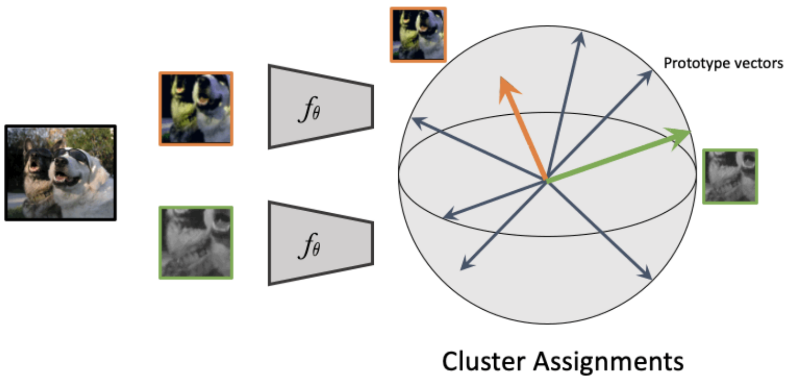


Figure 2.9: *Cluster assignments in SwAV [13]*

proved that in this online fashion, continuous codes work better than discrete codes. One probable reason is that discretion like rounding leads to an aggressive optimization and rapid convergence, hindering the sufficient training. Hence, they retain the soft code Q as the probability over set $\mathcal{Q}$, in forms of normalized exponential matrix:

$$Q* = \mathrm{Diag}(u)\exp(\frac{C^T Z}{\varepsilon})\mathrm{Diag}(v) \quad (2.10)$$

where u and v are renormalization vectors in $\mathbb{R}^K$ and $\mathbb{R}^B$ correspondingly, computed by a small number of matrix multiplications using the iterative Sinkhorn-Knopp algorithm.

- **Multi-crop augmentation strategy:** to balance the higher performance and heavier memory burden caused by increasing the number of crops, they propose a multi-crop strategy where V more low resolution crops are augmented in addition to two standard resolution crops, see Figure 2.10. These crops are augmented in the way same

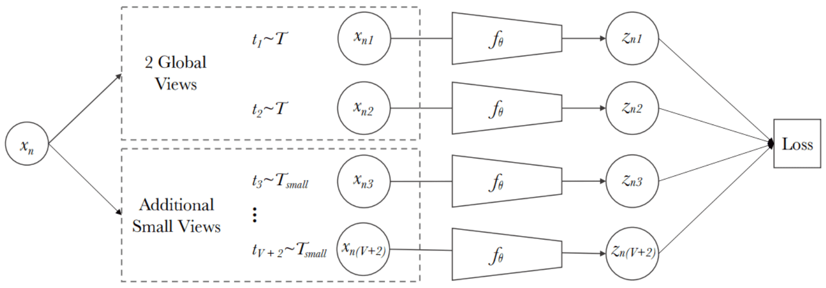


Figure 2.10: *The architecture of multi-crop strategy [13]*

with augmentation in SimCLR. Thus the total loss would be:

$$L(z_{t_1}, z_{t_2}, ..., z_{t_{V+2}}) = \sum_{i \in 1,2} \sum_{v=1}^{V+2} \mathbf{1}_{v \neq i} l(z_{t_v}, q_{t_i}) \tag{2.11}$$

2.3.4 Barlow Twins

Barlow Twins is a self-supervised learning proposed by Zbontar, Jure, et al. that applies redundancy-reduction principle, which is a principle first proposed in neuroscience. In 1961, neuroscientist H.Barlow proposed that highly redundant sensory inputs are recoded into a factorial code, explaining the organizations of visual system. In regards the research field of machine learning, it inspired many supervised and unsupervised learning algorithms. Likewise, Barlow Twins has a distinct objective function which aims to make the cross-correlation matrix of twin embeddings as close to the identity matrix as possible.[1] The global architecture of Barlow Twins is shown in Figure 2.11 More specifically, two augmented views Y_A and Y_B of an image X are made from augmentation family T. Two high-dimensional embeddings Z_A and Z_B are encoded by Y_A and Y_B with encoders f_θ respectively. Then Z_A and Z_B are mean-centered along the batch dimension before calculating loss function. The innovative loss function of Barlow Twins is:

$$L_{BT} = \sum_i (1 - C_{ii})^2 + \lambda \sum_i \sum_{j \neq i} C_{ij}^2 \tag{2.12}$$

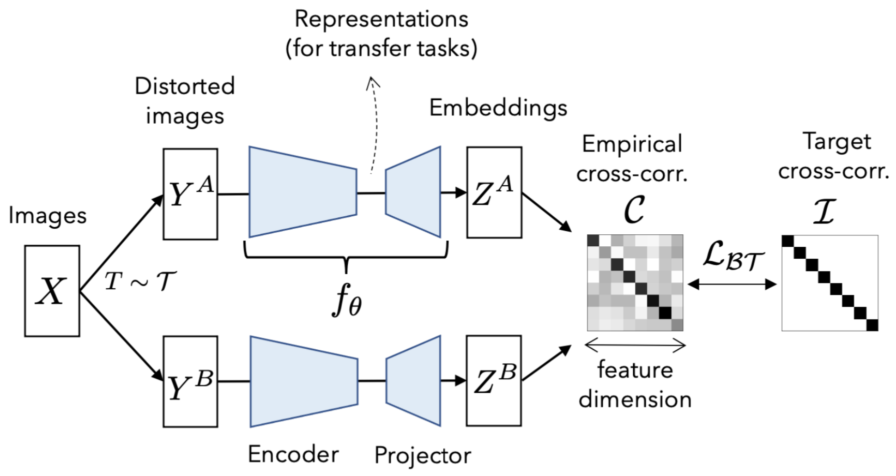


***Figure 2.11:** The architecture of Barlow Twins [1]*

where the former term $\sum_i (1 - C_{ii})^2$ represents the parts that should be invariant to distortion, and the latter term $\lambda \sum_i \sum_{j \neq i} C_{ij}^2$ represents the redundant parts that should be reduced. In the equation, λ is a positive constant that penalizes the off-diagonal elements in the correlation matrix C and the correlation matrix C is computed between two mean-centered high-dimensional embeddings Z^A and Z^B:

$$C_{ij} = \frac{\sum_b z^A_{b,i} z^B_{b,j}}{\sqrt{(\sum_b z^A_{b,i})^2} \sqrt{(\sum_b z^B_{b,j})^2}} \tag{2.13}$$

where b means the b-th mini-batch, and i or j means the dimension of the embedding z. As for the other implementation details, they are the same with the settings of SimCLR.

2.4 Visulization: t-SNE

Before evaluation, we want to visualize the high dimensional features on a low dimensional manifold to explore the hidden patterns of the features by human observation intuitively. t-SNE is an effective choice for dimensionality reduction of non-linear data, which is capable to interpret complex polynomial relationship between features. t-SNE, abbreviated from t-distributed stochastic neighbor embedding, is a common statistical method to visualize high-dimensional data in two or three dimensions evolved from stochastic neighbor embedding algorithms (SNE), proposed by Van der Maaten, Laurens, and Geoffrey Hinton in 2008. [14]. t-SNE algorithm calculates pairwise similarity metrics between samples both in high dimension and low dimension, and then make their distribution as similar as possible by means of minimizing Kullback-Leibler (KL) divergence.

We introduce SNE first before t-SNE. In SNE, the similarity between two data points are converted by Euclidean distances using softmax function. The similarity of datapoint x_j to datapoint x_j in high dimension is regarded as a conditional probability $p_{j|i}$, given by:

$$p_{j|i} = \frac{\exp(-\left\| x_i - xj \right\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\left\| x_i - xj \right\|^2 / 2\sigma_i^2)} \tag{2.14}$$

where σ_i is the Gaussian variance centered on datapoint x_i. $p_{i|i} = 0$ because the similarity of the same datapoint is 0. Similarly, for the counterparts datapoints y_j to datapoint y_j in low dimension, the conditional probability $q_{j|i}$ is given by:

$$q_{j|i} = \frac{\exp(-\left\| y_i - yj \right\|^2)}{\sum_{k \neq i} \exp(-\left\| y_i - yj \right\|^2)} \tag{2.15}$$

where Gaussian variance centered on datapoint y_i is set as $\frac{1}{\sqrt{2}}$ and $q_{i|i} = 0$. The objective cost function C applies KL divergence, which is a statistical measure of the dissimilarity between to probability distribution, given by:

$$C = KL(P|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{2.16}$$

where P and Q are the joint probability distribution of the datapoints in high and in low dimension respectively. The minimization of C with respect to y_i is calculated with gradients:

$$\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j) \tag{2.17}$$

It can be inferred that: large $p_{j|i}$ and small $q_{j|i}$ will lead to high cost; and small $p_{j|i}$ and large $q_{j|i}$ will lead to low cost. The latter is problematic because it means the cost is low when two datapoints are far in high dimensions and close after mapped into low dimensions.

t-SNE improve SNE by two means. First it is more reasonable to make the probability distribution symmetric by averaging the conditional probability distribution:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}, \quad q_{ij} = \frac{q_{i|j} + q_{j|i}}{2N} \tag{2.18}$$

Second, a student t-distribution is employed in low dimension map instead. This ensures that modeling dissimilar datapoints by means of large pairwise distances, and modeling similar datapoints by means of small pairwise distances.[14] In more detail, q_{ij} is changed to:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}} \tag{2.19}$$

Based on the new q_{ij}, the gradient of KL divergence is:

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1} \tag{2.20}$$

2.5 Evaluation: multinomial logistic regression

If no specific downstream task is assigned, a common evaluation protocol is to train a linear classifier on top of the features extracted by self-supervised learning. Usually, the classifier is implemented by multi-nomial logistic regression and semi-supervised learning is employed to fine-tune on a proportion of labels. [12] Multinomial logistic regression is an extension of logistic regression to multi-class problems, implemented by a set of independent binary logistic regression.

First logistic regression is briefly introduced. Logistic regression is widely used to model binary dependent variables, such as vote or do not vote, success or failure. It has its basis in the odds of a 2-level outcome of interest, where odds is the probability of success divided by the probability of failure. Mathematically, denote a response variable y has two outcomes 0 and 1 and a predictor variable matrix x, the odds of y is:

$$odds(y = 1) = \frac{P(y = 1|x)}{1 - P(y = 1|x)} = \frac{P(y = 1|x)}{P(y = 0|x)} \tag{2.21}$$

The log odds function of y, which is also the inverse of standard logistic function is defined as:

$$\ln(odds(y = 1)) = \beta^T x \tag{2.22}$$

where β is the coefficient vector. Inverse it back, the standard logistic regression function, also the sigmoid function, is:

$$\begin{aligned} P(y=1|x,\beta) &= \frac{1}{1+e^{-\beta^T x}} \\ P(y=0|x,\beta) = 1 - P(y=1|x,\beta) &= \frac{1}{1+e^{\beta^T x}} \end{aligned} \tag{2.23}$$

The two equations can be equally compacted to joint distribution:

$$\begin{aligned} P(y|x,\beta) &= (\frac{1}{1+e^{-\beta^T x}})^y (1-\frac{1}{1+e^{-\beta^T x}})^{1-y} \\ &= h_\beta(x)^y (1-h_\beta(x))^{1-y} \end{aligned} \tag{2.24}$$

Based on that, the cost function of a single sample is taking a negative logarithmic function, expressed in formula:

$$cost(h_\beta(x),y) = -y^{(i)} \times log(h_\beta(x)) - (1-y^{(i)}) \times log(1-h_\beta(x)) \tag{2.25}$$

where the superscript (i) indicates the i-th sample. For a dataset or batch contains m samples, the total cost function is their sum:

$$J(\beta) = -\frac{1}{m}\sum_{i=1}^{m}[-y^{(i)} \times log(h_\beta(x)) - (1-y^{(i)}) \times log(1-h_\beta(x))] \tag{2.26}$$

In machine learning, the minimization of cost function is usually implemented by gradient descent, which iterates on every parameter β_j in coefficient vector β:

$$\begin{aligned} \beta_j \leftarrow \beta_j - \lambda\frac{\partial J(\beta)}{\partial \beta_j}&, \\ where \quad \frac{\partial J(\beta)}{\partial \beta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\beta(x^{(i)}) - y^{(i)})x_j^{(i)}& \end{aligned} \tag{2.27}$$

where λ means the hyperparameter learning rate. [4]

Extensively, in multi-nomial logistic regression for K possible classes, the K_{th} class is chosen as the base with coefficient $\beta_K = 0$, so $K-1$ classes are separately regressed against the K_{th} class, expressed in formula:

$$\begin{aligned} \ln \frac{Pr(Y_i = 1)}{Pr(Y_i = K)} &= \beta_1 \cdot X_i \\ \ln \frac{Pr(Y_i = 2)}{Pr(Y_i = K)} &= \beta_2 \cdot X_i \\ &... \\ \ln \frac{Pr(Y_i = K-1)}{Pr(Y_i = K)} &= \beta_{K-1} \cdot X_i \end{aligned} \tag{2.28}$$

Adding up the equations, $Pr(Y_i = K)$ can be solved by;

$$Pr(Y_i = K) = \frac{1}{1 + \sum_{j=1}^{K-1} e^{\beta_j \cdot X_i}} \tag{2.29}$$

and generally for $Pr(Y_i = k)$ where $k = 1, ..., K-1$:

$$Pr(Y_i = k) = \frac{e^{\beta_k \cdot X_i}}{1 + \sum_{j=1}^{K-1} e^{\beta_j \cdot X_i}} \tag{2.30}$$

And each of the binary linear classifiers is solved in the way same with standard logistic regression.

Chapter 3

Experiments

Several experiments with various configurations were conducted to assess the performance of the introduced self-supervised learning algorithms on kidney biopsy WSI. The flowchart of the experiments is given in Figure 3.1. The experiments follow three steps: data pre-processing clarified in section 4.1, training based on self-supervised learning clarified in section 4.2, evaluation based on classification clarified in section 4.3.

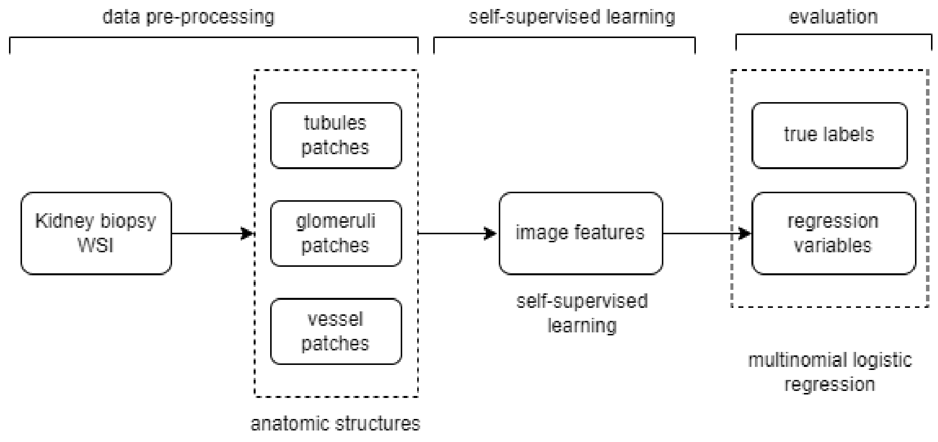


Figure 3.1: *Flowchart of the experiments*

3.1 Data Pre-processing

All the kidney biopsy WSI data is anonymized and collected in Leiden University Medical Center (LUMC). In the kidney WSI, we are interested in recognizing three essential anatomic structures: glomeruli, tubules and vessels. One example of the three anatomic structures in a healthy region is shown in Figure 3.2. We cut the WSI in 20x magnification into 256 $\times$

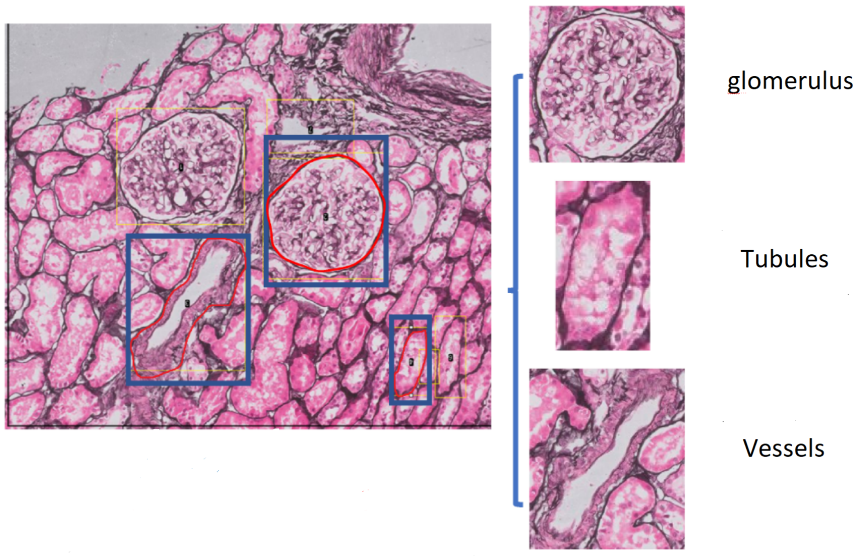


Figure 3.2: *Anatomic structures of kidney*

256 patches and labelled them according to the structures of the largest proportion in a patch. There are 19456 patches, including 5192 glomeruli patches, 3600 tubule patches and 10664 vessel patches. See examples of the patches of glomerulus, tubules, and vessels in Figure 3.3, Figure 3.4 and Figure 3.5 respectively. It is observed that the structures of vessels have the most complex patterns among them, and some structures become less distinguishable when they are diseased.

3.2 Self-supervised learning

The training part of the experiments is conducted by applying the four self-supervised learning algorithms on the patches without labels generated from the kidney biopsy WSI dataset mentioned above. For each input of 256 $\times$ 256 image patch, an output of 2048 $\times$ 1 vector feature will be obtained. (The output feature is extracted at the last layer of backbone but not MLP head.) The settings of parameters of each algorithm are given in section 3.2.1. The initialization of weights of the architecture is described in section 3.2.2.

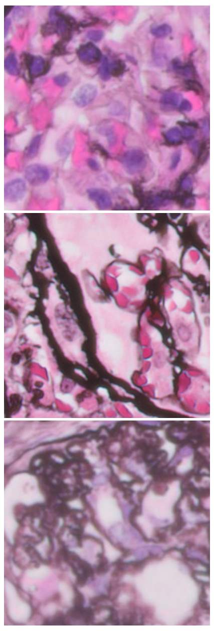


Figure 3.3: glomerulus

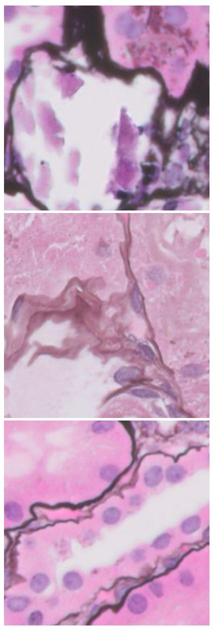


Figure 3.4: tubules

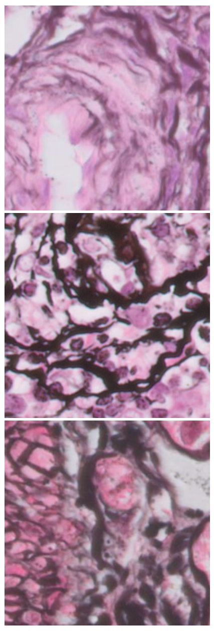


Figure 3.5: vessels

3.2.1 Parameters

The settings of the parameters for the four algorithms mostly follow the default configuration in their corresponding papers, see Table 3.1. The explanation of the parameters is followed.

Table 3.1: Parameter Settings of Self-supervsed learning Algorithms

	SimCLR	**MoCo**	**SwAV**	**BarlowTwins**
epochs	500	500	500	500
backbone	ResNet-50	ResNet-50	ResNet-50	ResNet-50
MLP heads	2048×2048 2048×128	2048×2048 2048×128	2048×2048 2048×128	2048×8192 8192×8192 8192×8192
batchsize	64	32	64	64
optimizer	SGD	SGD	SGD	SGD
learning rate	Cosine annealing	Multi-step	Cosine annealing	Cosine annealing

- **Epochs:** We set the training epochs 500 for the four algorithms, because it is enough for convergence while not wasting computing resources.

- **Backbone:** The backbones of the four algorithms are all ResNet-50.

- **Heads:** The heads of Barlow Twins are 3-layer MLP in shape $2048 \times 8192 \times 8192 \times 8192$. The heads of ther other three algorithms are 2-layer MLP in shape $2048 \times 2048 \times 128$.

- **Batchsize:** The batchsize of MoCo is 32, while that of the other algorithms is 64.

- **Optimizer:** The optimizers of the four algorithms are all stochastic gradient descent (SGD). SGD is a stochastic approximation of gradient descent optimization. Standard gradient descent optimizes the objective function in an iterative way: $w = w - \eta \bigtriangledown Q(w)$, where η is the learning rate and $Q(w)$ is the total loss values on the dataset or the batch. But stochastic gradient descent calculates and updates the derivative from every sample as: $w = w - \eta \bigtriangledown Q_i(w)$, where $Q_i(w)$ is the loss value of the i-th sample in the dataset or the batch. Compared with gradient descent, SGD requires less computation memory and speed up the convergence. [11]

- **Learning rate:** The learning rate schedule of MoCo is multi-step, while that of the others is cosine annealing.
  Multi-step learning rate schedules the decay of the learning rate with a certain multiplicative factor when a milestone, which means a certain number of epochs, is reached. The certain number of epochs is a hyperparameter. In this training, the decay factor is set at 0.1, and the starting learning rates are set at 0.03 and 0.003. The milestones are 120 before learning rate reaches 0.003 and 160 after then correspondingly. When learning rate decays to 0.0003, it keeps at this constant.
  With Cosine annealing schedule, the learning rate starts with a large value and decreases to minimum value rapidly, and then restart from large value and decreases rapidly again. Specifically, Cosine annealing schedules the learning rate as: $\eta_t = \eta^i_{min} + \frac{1}{2}(\eta^i_{max} - \eta^i_{min})(1 + cos(\frac{T_{cur}}{T_i}\pi))$, where η^i_{max} and η^i_{min} are the range of learning rate, T_{cur} and T_i are the current epoch since the restart and the epoch since the last linear warm-up finished respectively. [15] The range of the learning rate for BarlowTwins is [0.002, 0.2], for the other algorithms is [0, 4.8].

3.2.2 Initialization

Wherever possible, neural network should be trained and fine-tuned with a pre-trained model, since pre-training is much more efficient than training from scratch. Though there are very few pre-trained models in the field of medical imaging, it has been identified that using even a few early layers from a pre-trained ImageNet model can improve both the speed of training and final accuracy of medical imaging models. [8] Following that, the weights of the four algorithms are initialized by the pre-trained weights of the corresponding algorithms on ImageNet, which is a large public natural image database designed for visual object recognition research. The top-1 accuracy of the pre-trained weights on ImageNet of the four algorithms is listed in Table 3.2. Compared with the accuracy using supervised learning, the four self-supervised learning algorithms have comparable performance. Using the weights trained on ImageNet to initialize the weights for this medical imaging task could make the training more robust and fast to converge.

Table 3.2: *Accuracy of the pre-trained weights on ImageNet*

	supervised	**SimCLR**	**SwAV**	**MoCo**	**Barlow Twins**
Top-1 acc.	76.5%	69.3%	75.3%	66.4%	73.2%

3.2.3 Environments

For the software environment used in the experiments, Python is the main language and Vissl is the main library. Vissl is produced by Facebook, which implements the state-of-the-art self-supervised learning approaches on top of Pytorch. For the hardware environment, we use 3 graphics processing unit (GPU) and their models are Quadro RTX 6000.

3.3 Classification

Within each self-supervised architecture, we retain the features of the image patches at the last layer of the ResNet-50 backbone to do evaluation, so the MLP heads are discarded. First, before quantifying the evaluation of the algorithms, we use t-SNE to visualize the vector features of the image patches in two dimensions, with setting perplex=50.0 and random initialization. To evaluate the features learned by self-supervised learning, a linear classifier with multinomial logistic regression is employed to classify the features on top of the ResNet-50 backbone of the self-supervised learning architecture. We use $1\%, 10\%, 30\%, 50\%, 70\%, 90\%$ of the labels to supervise the training of the classification models respectively and predict the labels of the rest of the data. To improve the robustness of the results, for each certain percentage of labels, the training of the classification models is repeated 5 times with 5 different random sampled dataset. The evaluation metrics of the prediction on the test set is used for quantifying the performance of the features extracted by the self-supervised learning algorithms. As for evaluation metrics, the mean and standard deviation of the accuracy scores of the models and the precision, recall and F1 scores for each class are calculated. Accuracy score of the classification model is calculated by the fraction of correct predictions, formulated as:

$$accuracy = \frac{\#correct\ predictions}{\#total\ predictions} \tag{3.1}$$

where # means 'the number of'. In multi-nomial classification, supposing a confusion matrix oriented as row for truth and column for prediction, precision and recall for the i-th class C_i are defined as following:

$$precision_i = \frac{C_{ii}}{\sum_j C_{ji}}, \quad recall_i = \frac{C_{ii}}{\sum_j C_{ij}} \tag{3.2}$$

Precision is the ratio that the correct prediction of C_i out of all the prediction of C_i, and recall is the ratio that the correct prediction of C_i out of all the true C_i F1 score for the i^{th} class C_i are defined as the harmonic mean of precision and recall:

$$F1_i = \frac{precision_i \cdot recall_i}{precision_i + recall_i} \tag{3.3}$$

Chapter 4

Results

The results obtained by the experiments are discussed in this chapter. After the features obtained, they are visualized by t-SNE and trained by multi-nomial logistic regression.

For each self-supervised algorithm, the features are extracted at the last layer of its ResNet-50 backbone, at 0, 100, 200, ..., 500 training epochs. Specifically, at epoch=0, the features are extracted using the weights pre-trained on ImageNet. Firstly, the 2-dimensional visualization of its features based on t-SNE are produced. To display the points clearly, we randomly generated 2500 samples to draw the t-SNE plot, consisting 456 tubules, 1363 vessels and 681 glomeruli. In the plots, the points for tubules, vessels and glomeruli are colored by blue, green and red respectively. Secondly, multinomial logistic regression models are applied to the features and the corresponding measurement metrics are given. The measurement metrics include accuracy scores of the whole model, precision, recall and f1 scores of each class.

Moreover, to examine the capability of 'self-supervised' for each algorithm, the percentages of the labels used for training the linear classifiers are varied among 1%, 10%, 30%, 50%, 70% and 90%. To compare the efficiency of the percentage of labels, we define the marginal benefits as the maximum scores a model can obtain for additional percentage of labels in training. We regard the model with the percentage of labels that has the maximum marginal benefit as the optimal model.

4.1 SimCLR

First, the 2-dimensional t-SNE plot of the features at different epochs trained by SimCLR is presented, see Figure 4.1. At epoch=0, when only the weights pre-trained on ImageNet are applied, the points of glomeruli and tubules are already clustered respectively, and the points of vessels spread around without mixing up with the points of other classes too much. It indicates that the weights pre-trained on ImageNet accelerate the convergence speed. At epoch=100, the mixed area between the clusters of glomeruli, the clusters of tubules and the surrounding points of vessels are reduced further. From then, there is no significant change in the clusters. It can be speculated that the features extracted by SimCLR at are capable of recognizing the three classes.

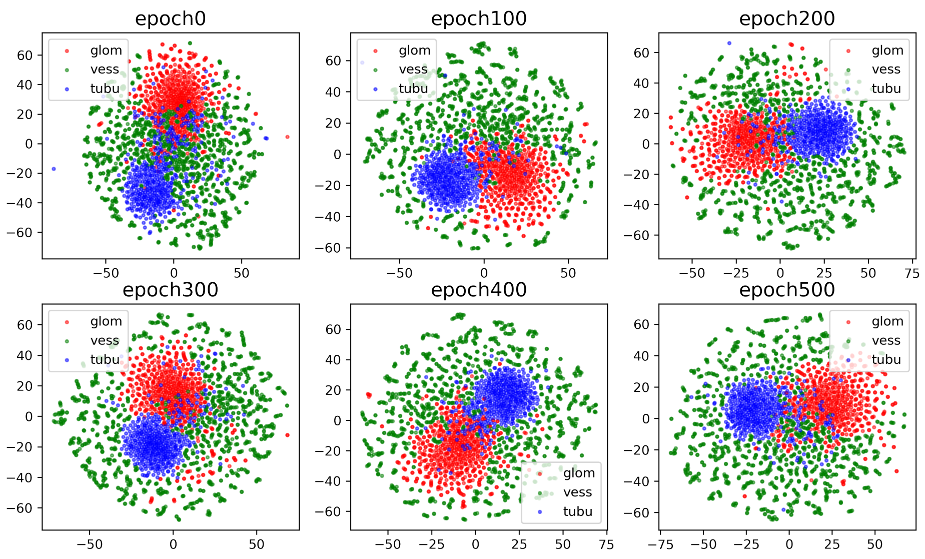


Figure 4.1: *2-dimensional t-SNE plot of the features trained by SimCLR at every 100 epochs*

4.1.1 Classification

The mean of the accuracy scores of the multinomial logistic regression model trained with the features extracted by SimCLR and different percentage of labels are calculated, see Table 4.1. A plot of the mean of the accuracy scores trained with different percentage of labels based on the SimCLR features at different epochs is provided in Figure 4.2. The bands in the plot are the 95% confidence interval of the mean accuracy scores. It

Table 4.1: *The mean accuracy scores of the linear classification model trained with features extracted by SimCLR and different percentage of labels at every 100 epochs*

Percent \ Epochs	**0**	**100**	**200**	**300**	**400**	**500**
1%	76.90%	81.44%	80.39%	80.24%	79.88%	79.80%
10%	85.31%	**88.44%**	88.06%	88.13%	87.97%	87.87%
30%	88.61%	91.22%	90.96%	90.99%	91.20%	90.99%
50%	90.13%	92.55%	92.38%	92.29%	92.34%	92.39%
70%	91.01%	93.21%	92.98%	92.91%	92.97%	93.11%
90%	91.52%	93.63%	93.60%	93.22%	93.80%	93.63%

shows that more labels used for training would lead to less uncertainty, depicted as narrower confidential interval bands. Comparing different epochs, it indicates that at epoch=100, the SimCLR features perform best to be classified. Comparing different percentage of labels, it shows that with only 1% labels the classifier could reach 81.44% mean accuracy score at epoch=100 on the test dataset with 99% labels, which is fairly good. Increasing the percentage of labels to 10%, the mean accuracy score at epoch=100 improves to 88.44%, where receives the maximum marginal benefit. The growth of mean accuracy score slows down from increasing 10% labels to 90% labels. With 90% labels, the accuracy score is around 93.63%, but the 5% gain of accuracy score is costly considering the expense of 80% extra labels. Afterward, we shift our perspective of evaluation from

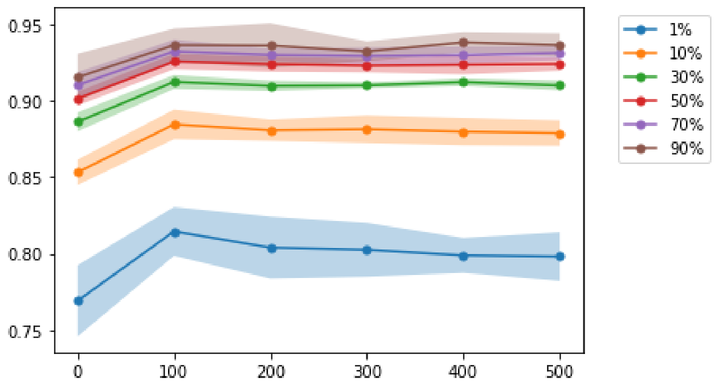
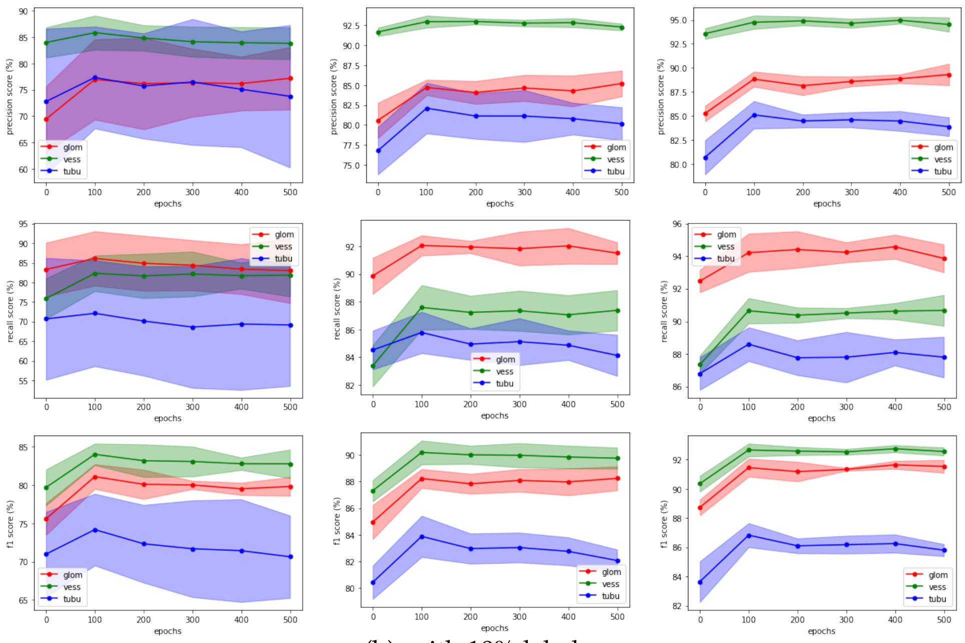


Figure 4.2: *The accuracy scores of the linear classification model based on SimCLR features at every 100 epochs*

overview to each class. Respecting the trade-off between the cost of labels and the accuracy scores, we focus on the models trained with 1%, 10% and 30% labels. Their mean precision, recall and F1 scores at different training epochs are plotted in Figure 4.3. The 95% confidence interval bands of the models trained with 1% labels are remarkably wide. For instance, that of the precision score for tubules is about 28%. The uncertainty becomes much less when the labels for training are increased to 10%, where the 95% confidence interval of precision score for tubules is around 5%. Overall, the improvement of the scores is big from 1% labels to 10% labels, while little from 10% labels to 30% labels. Hence, the models trained with 10%



(a) with 1% labels (b) with 10% labels (c) with 30% labels

Figure 4.3: *precision, recall and F1 score of the classification model based on SimCLR features*

at epoch=100 are regarded as the ones manage the cost of labels but maintain good scores. The scores of the model trained with 10% labels at epoch 100 are listed in Table 4.2. The first two columns show a trade-off between the precision and recall scores within some classes: glomeruli have high recall scores above 90% while low precision scores below 90%, and vessels are the reversal. Tubules have worse performance on both precision and recall scores. According to F1 scores, the performance of the SimCLR features for each class are ranked by: vessels, glomeruli and tubules.

Table 4.2: *The precision, recall and F1 score at epoch=100 of the classification model trained with 10% labels based on SimCLR features*

labels \ scores	**precision**	**recall**	**F1**
glomerulus	84.69%	92.06%	88.22%
vessel	92.93%	87.58%	90.18%
tubule	82.01%	85.77%	83.88%

4.2 MoCo

First, the 2-dimensional t-SNE plot of the features trained by MoCo at every 100 epochs is presented, see Figure 4.4. At epoch=0, the weights pre-trained on ImageNet gather the points of each class in to clear but small clusters. At epoch=100, the small clusters of each class start to merge into bigger clusters, where the boundaries between the clusters become sharp. From then, there is no notable development in the clusters. It can be speculated that the features extracted by MoCo at are capable of recognizing the three classes.

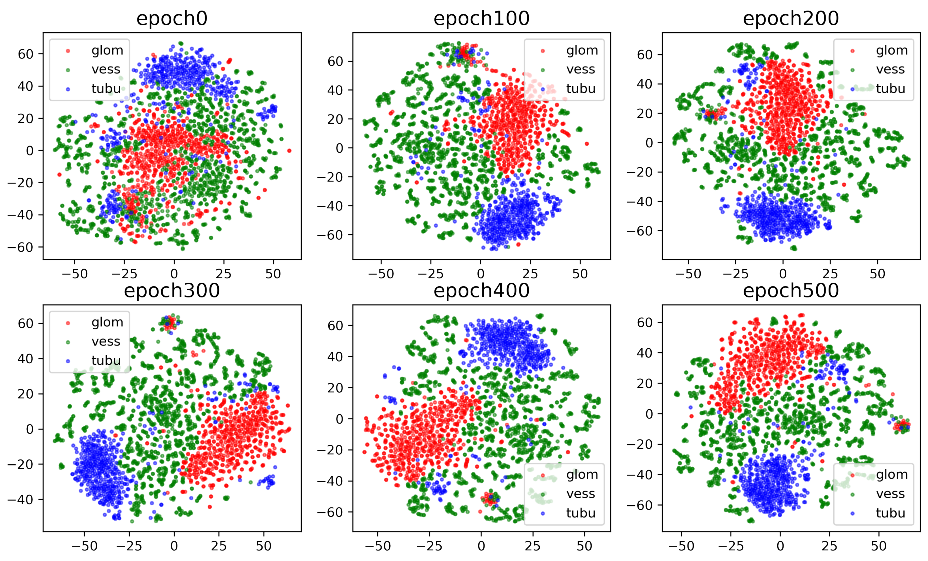


Figure 4.4: *2-dimensional t-SNE plot of the features trained by MoCo at every 100 epochs*

4.2.1 Classification

The mean of the accuracy scores of the multinomial logistic model trained with the features extracted by MoCo at different epochs and different percentage of labels are calculated, see Table 4.3. A line chart of the mean accuracy scores in the table is provided in Figure 4.5. The bands in the plot are the 95% confidence interval of the mean accuracy scores. The mean accuracy scores grow little as the training epochs continue, showing that Moco did not learn further specific knowledges with the specialized kidney biopsy WSI datasets. Generally, the mean accuracy scores of the classification model are slightly higher at epoch=300. Regarding the efficiency of labels, the classification model trained with 10% labels receives the maximum marginal benefit, where the mean accuracy score is raised to 89.14% from 89.10% at epoch=300. Next we shit our perspective of evaluation from overview to each class. Considering the trade-off between the cost of labels and the accuracy scores, we focus on the models trained with 1%, 10% and 30% labels. The precision, recall and F1 scores of the three models every 100 epochs are plotted in Figure 4.6. At epoch=300, the MoCo features maintain good robustness even for the models trained with only 1% labels, the biggest 95% confidence interval is about 10%. Increasing the percentage of labels from 1% to 10% makes scores grow effectively, for instance, at epoch=200, the precision scores for tubules and vessels increase about 12%. Compared to this, the benefits of increasing the labels from 10% to 30% is small, about 3% for the scores. Thus, we regard the model trained with 10% labels at epoch=300 are the optimal choice. The precision, recall and F1 scores of the model trained with 10% at epoch=300 are listed in Table 4.4. Glomeruli and tubules have high recall scores about 94.84% and 90.76% respectively, and low precision scores

Table 4.3: *The mean accuracy scores of the linear classification model trained with features extracted by MoCo and different percentage of labels at every 100 epochs*

Epochs / Percent	**0**	**100**	**200**	**300**	**400**	**500**
1%	79.24%	79.97%	80.83%	80.91%	80.97%	80.93%
10%	88.53%	88.96%	89.10%	**89.14%**	89.13%	89.12%
30%	91.49%	91.29%	91.45%	91.47%	91.44%	91.45%
50%	92.80%	92.59%	92.63%	92.63%	92.62%	92.64%
70%	93.53%	93.36%	93.24%	93.29%	93.29%	93.28%
90%	94.33%	94.29%	93.95%	93.99%	94.02%	94.08%

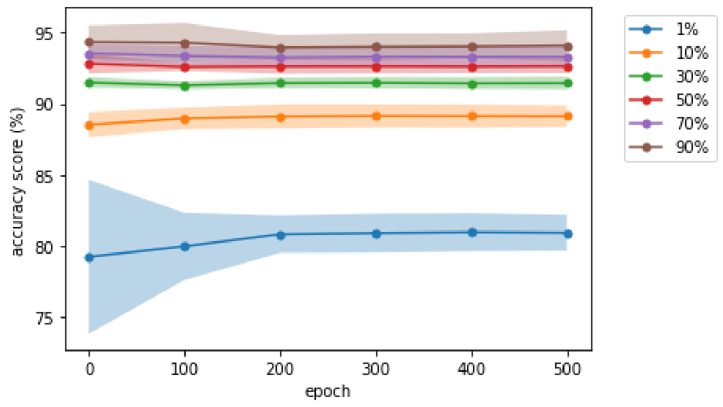


Figure 4.5: *The accuracy scores of the linear classification model based on Moco features at every 100 epochs*

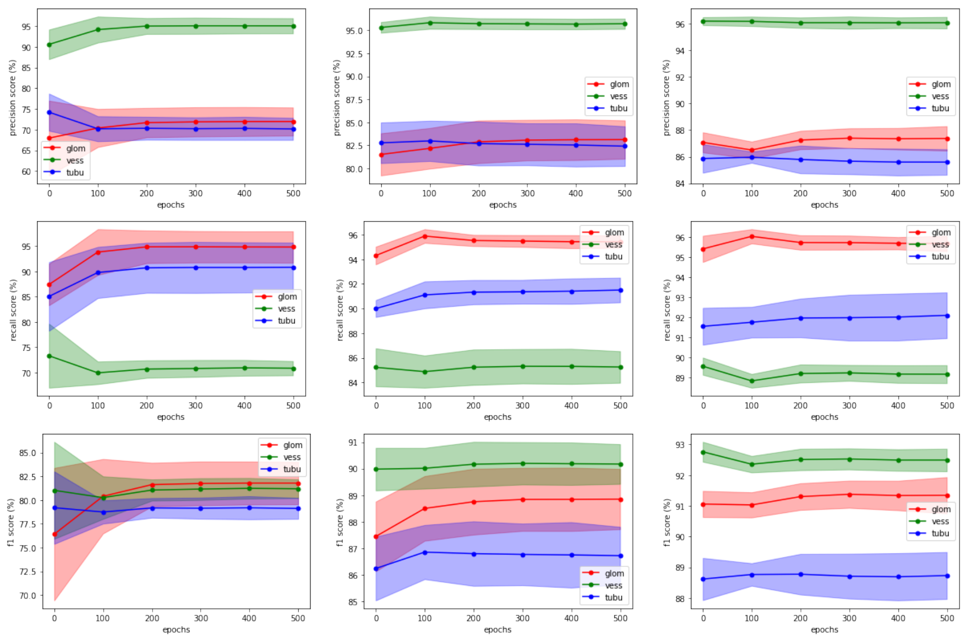
about 71.86% and 70.22% respectively. On the contrast, vessels have high precision scores about 95.07% and low recall scores about 70.80%. According to F1 score, the abilities of MoCo features to classify glomeruli, vessels and tubules are similar, about 81.74%, 81.15% and 79.14% respectively.

Table 4.4: *The precision, recall and F1 score at epoch=200 of the classification model trained with 10% labels based on MoCo features*

scores / labels	**precision**	**recall**	**F1**
glomerulus	71.86%	94.84%	81.74%
vessel	95.07%	70.80%	81.15%
tubule	70.22%	90.76%	79.14%

4.3 SwAV

First, the 2-dimensional t-SNE plot of the features trained by SwAV is presented in Figure 4.7. At epoch=0, there are some clusters of the classes overlap each other. At epoch=100, it seems that the clusters of tubules merge at center and the clusters of vessels gather at periphery. The 2-dimensional t-SNE plot of SwAV features does not reveal obvious clusters, so the quantified evaluation with linear classifiers is very necessary.



(a) with 1% labels (b) with 10% labels (c) with 30% labels

Figure 4.6: *precision, recall and F1 score of the classification model based on MoCo features*

4.3.1 Classification

The mean of the accuracy scores of the multinomial logistic classification model trained with the features extracted by SwAV and different percentage of labels are calculated, see Table 4.5. A line chart of the mean accuracy scores in the table in is Figure 4.8. The bands in the chart are the 95% confidence interval of the mean accuracy scores. The bands become narrower apparently after increasing the percentage of labels for training from 1% to 10%. As the training epochs continue, the mean accuracy scores slightly increase from 91.35% at epoch=0 to 91.39% at epoch=100, then decrease. Therefore, the model trained with 10% labels at epoch=100 has the maximum marginal benefit considering the cost of labels, training time and the mean accuracy scores. Here, the optimal model is trained with 10% labels at epoch =100, with the mean accuracy score around 91.39%. Afterwards, we shift the perspective of evaluation from overview to each class. We focus on the classification models trained with 1%, 10%, 30% models. The precision, recall and F1 scores of the three models every 100 epochs are shown in Figure 4.9. Regarding precision scores, it is noteworthy that the scores of tubules vary little with the percentage of labels used for training,

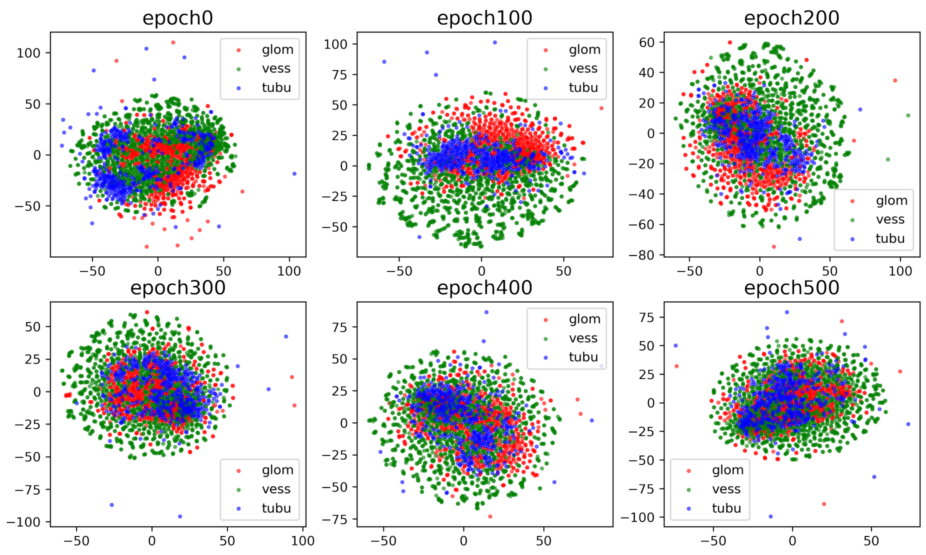


Figure 4.7: *2-dimensional t-SNE plot of the features trained by SwAV at every 100 epochs*

which are 87.33%, 90.40% and 90.65% at epoch=100 for the models trained with 1%, 10% and 30% respectively. Vessels have the largest growth in precision scores, from 77.02% to 93.65% at epoch=100, with the percentage of labels used for training increased from to 1% to 10%. With regard to recall, the increasing percentage of labels used for training improve the scores of tubules obviously, which are about 51.44%, 85.10%, 89.52% at epoch=100 for 1%, 10% and 30% labels. The scores of glomeruli and vessels are about 92.16% and 93.14% respectively, which are quite good in the models trained with 10% labels at epoch=100. For F1, the scores in the models trained with 30% labels at epoch=100 are fairly good, which are about 93.81%, 94.84% and 89.96% for glomeruli, vessels and tubules respectively. Considering the marginal benefit of the percentage of labels, we list the scores for the three classes at epoch=100, see Table 4.6. Out of our expectation, SwAV features not only have high scores on the measurement metrics, they also balance the trade-off between precision and recall scores. The precision and recall scores for glomeruli are about 90.38% and 92.16%; for vessels are about 92.79% and 93.14%; for tubules are about 88.65% and 85.10%. It tells that SwAV features are roughly equally good at assigning the labels of each class properly as it is at classifying instances of each class properly.

Table 4.5: *The mean accuracy scores of the linear classification model trained with features extracted by SwAV and different percentage of labels at every 100 epochs*

Percent \ Epochs	**0**	**100**	**200**	**300**	**400**	**500**
1%	79.97%	77.51%	75.27%	72.65%	71.12%	69.72%
10%	91.35%	**91.39%**	90.01%	87.85%	86.93%	86.12%
30%	93.58%	93.66%	92.63%	90.99%	90.13%	89.69%
50%	94.16%	94.36%	93.45%	92.02%	91.25%	90.94%
70%	94.65%	94.81%	94.01%	92.81%	91.78%	91.56%
90%	94.89%	94.89%	94.19%	93.08%	91.97%	91.97%

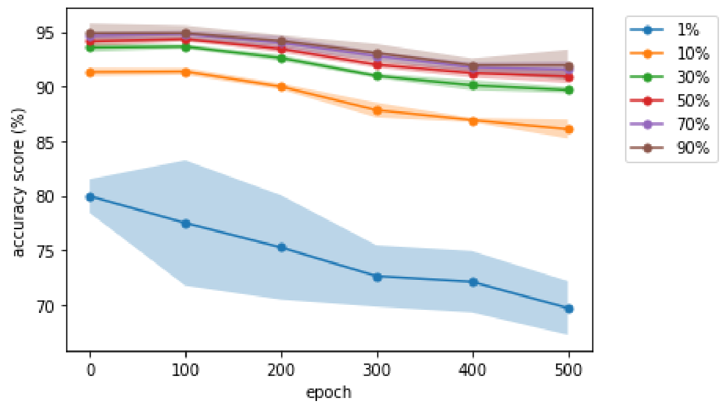
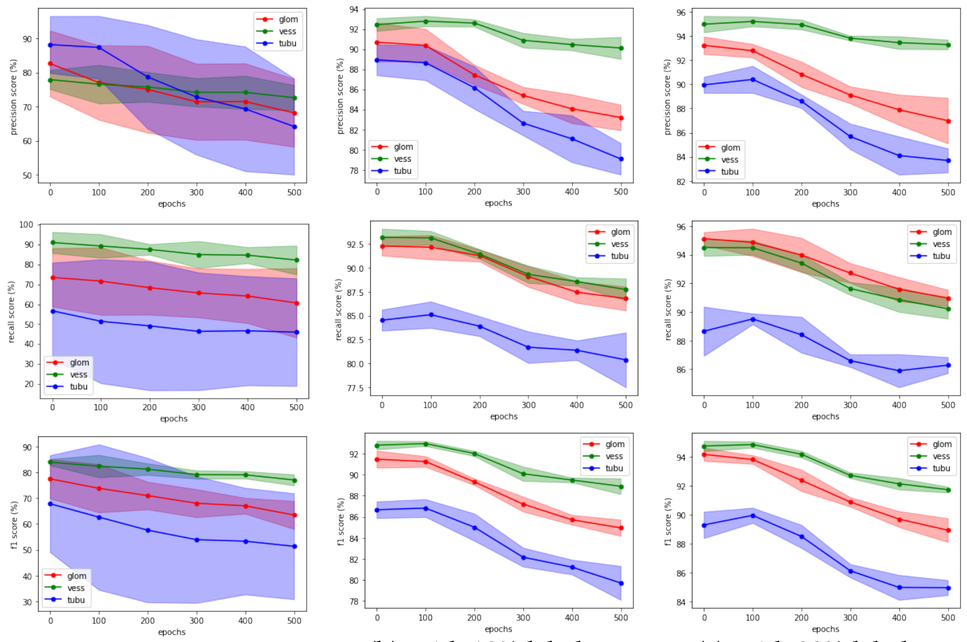


Figure 4.8: *The accuracy scores of the linear classification model based on SwAV features at every 100 epochs*

4.4 Barlow Twins

First, the 2-dimensional t-SNE plot of the features trained by Barlow Twins is plotted, see Figure 4.10. At epoch=0 and epoch=100, the points of the three classes are totally blended. After epoch=200, it implies that the points of glomeruli and tubules are gathering slightly. From then, the points of tubules and glomeruli seem to gather more closely, yet there are not clear clusters. Observing the t-SNE plot of BarlowTwins features, we speculate that the ability of BarlowTwins features for classification is limited.



(a) with 1% labels (b) with 10% labels (c) with 30% labels

Figure 4.9: *precision, recall and F1 score of the classification model based on SwAV features*

4.4.1 Classification

The mean of the accuracy scores of the multinomial logistic classification model trained with the features extracted by BarlowTwins and different percentage of labels are calculated, see Table 4.7. A line chart of the mean accuracy scores in the table see Figure 4.11. The bands in the chart are the 95% confidence interval of the mean accuracy scores.The 95% confidence interval bands are not wide, even for the models trained with 1% labels. From epoch=0 to epoch=100, the mean accuracy scores are relatively low and steady, which are about 53%, 61% and 65% for the models

Table 4.6: *The precision, recall and F1 score at epoch=100 of the classification model trained with 10% labels based on SwAV features*

scores labels	**precision**	**recall**	**F1**
glomerulus	90.38%	92.16%	91.26%
vessel	92.79%	93.14%	92.96%
tubule	88.65%	85.10%	86.83%

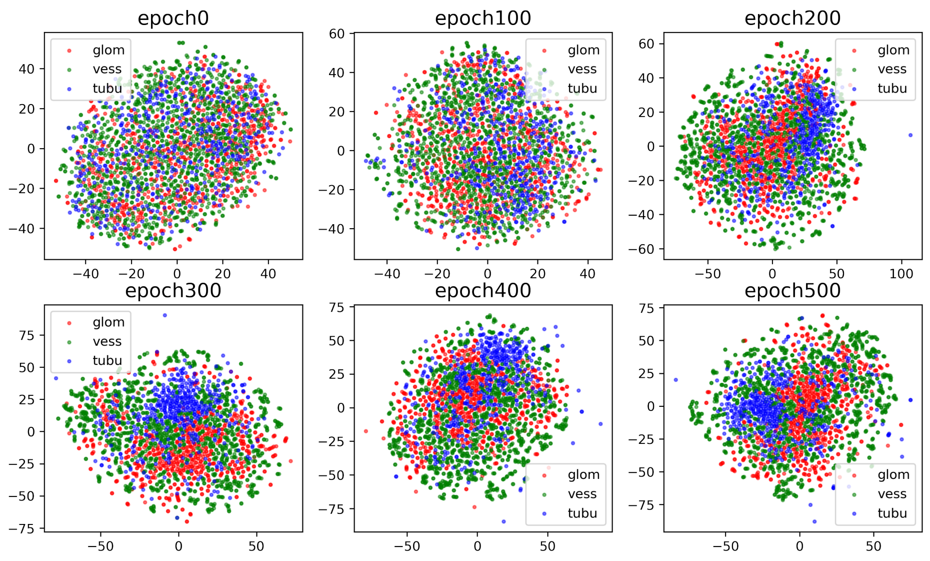


Figure 4.10: *2-dimensional t-SNE plot of the features trained by Barlowtwins at every 100 epochs*

trained with 1%. 10% and 30% labels, respectively. From epoch=100 and epoch=200, the mean accuracy scores rise sharply about 15%. Then, as training epochs continue, the mean accuracy scores go up gradually, so the highest scores are achieved at epoch=500. At epoch=500, the mean accuracy scores go up to 82.46% from 71.94% after increasing the percentage of labels for training from 1% to 10%, which is the largest improvement among them. The mean of the accuracy scores of the multinomial logistic

Table 4.7: *The mean accuracy scores of the linear classification model trained with features extracted by Barlow Twins and different percentage of labels at every 100 epochs*

Epochs / Percent	**0**	**100**	**200**	**300**	**400**	**500**
1%	52.86%	53.04%	66.88%	68.86%	70.94%	71.94%
10%	60.92%	61.75%	76.59%	79.55%	81.48%	**82.46%**
30%	65.19%	64.95%	80.37%	83.31%	85.07%	85.84%
50%	66.70%	66.89%	82.05%	85.15%	86.40%	86.97%
70%	67.78%	68.17%	83.19%	86.09%	87.65%	87.87%
90%	68.79%	68.91%	83.70%	86.90%	88.29%	88.76%

classification model trained with the features extracted by BarlowTwins and different percentage of labels are calculated, see Table 4.7. A line chart

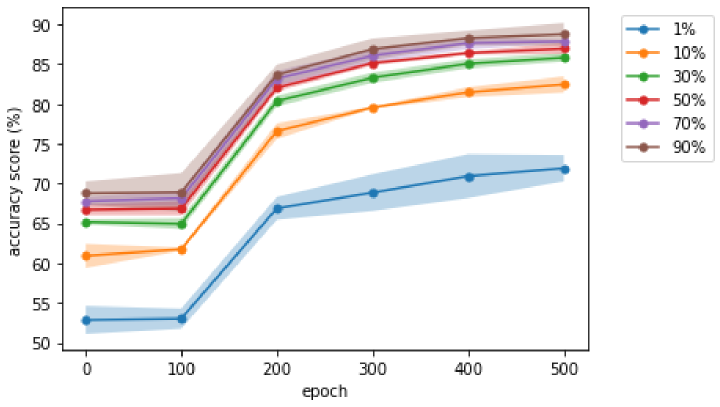
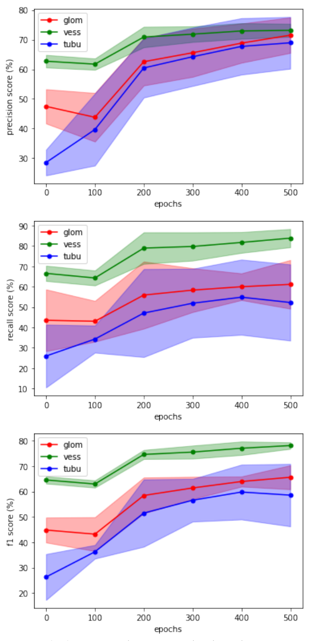


Figure 4.11: *The accuracy scores of the linear classification model based on BarlowTwins features at every 100 epochs*

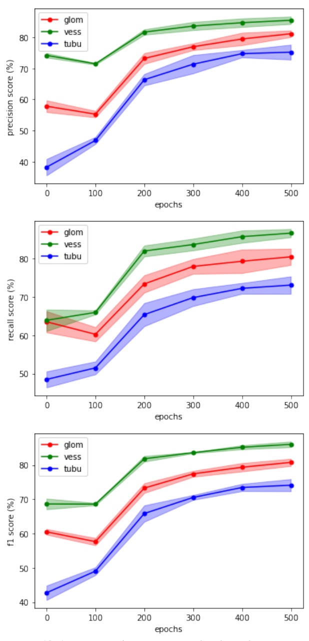
of the mean accuracy scores in the table see Figure 4.11. The bands in the chart are the 95% confidence interval of the mean accuracy scores.The 95% confidence interval bands are not wide, even for the models trained with 1% labels. From epoch=0 to epoch=100, the mean accuracy scores are relatively low and stable, which are about 53%, 61% and 65% for the models trained with 1%. 10% and 30% labels, respectively. From epoch=100 and epoch=200, the mean accuracy scores rise sharply about 15%. Then, as training epochs continue, the mean accuracy scores rise gradually, so the highest scores are achieved at epoch=500. At epoch=500, the mean accuracy scores achieve 82.46% from 71.94% after increasing the percentage of labels for training from 1% to 10%, which is the largest improvement among them.

Table 4.8: *The precision, recall and F1 score at epoch=500 of the classification model trained with 10% labels based on BarlowTwins features*

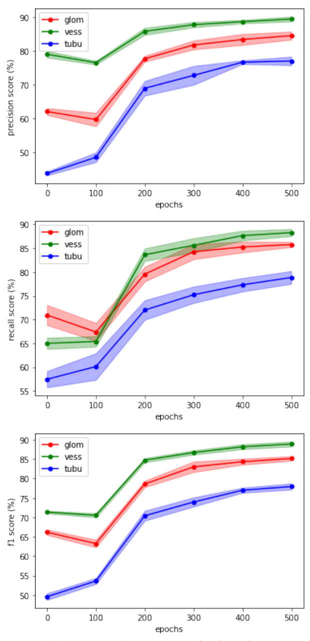
scores / labels	**precision**	**recall**	**F1**
glomerulus	81.11%	80.43%	80.77%
vessel	84.45%	86.61%	86.02%
tubule	75.21%	73.06%	74.11%



(a) with 1% labels



(b) with 10% labels



(c) with 30% labels

Figure 4.12: *precision, recall and F1 score of the classification model based on BarlowTwins features*

Chapter 5

Discussion and Conclusion

This chapter compares the performance of the four self-supervised learning algorithms on classifying the common anatomic structures on kidney biopsy WSI in section 5.1. Moreover, the limitations met in the experiments and corresponding future improvements are also discussed in this section. Finally, a brief summary of contribution of this study is drawn in section 5.2.

5.1 Discussion

In this study, the applications of four common self-supervised learning methods on kidney biopsy WSI patches classification are investigated, including SimCLR, MoCo, SwAV and BarlowTwins. We employ a pipeline which extracts features with self-supervised learning algorithms and evaluate the features with a linear classifier based on the measurement metrics. The self-supervised learning architectures share the same backbone network, ResNet-50 and are initialized by the weights pre-trained on ImageNet correspondingly. The linear classifiers are multinomial logistic regression models trained with different percentage of labels, including 1%, 10%, 30%, 50%, 70% and 90%.

With respect to the comprehensive performance, the features extracted by the four self-supervised learning algorithms gain sufficient accuracy scores supervised by only 10% labels: 88.44% at epoch=100 for SimCLR, 89.14% at epoch=300 for MoCo, 91.39% at epoch=100 for SwAV and 82.46% at epoch=500 for BarlowTwins see Table 5.1. So surprisingly, though there is no distinct clusters in the t-SNE plot of SwAV, SwAV obtains the highest accuracy scores. Moreover, the difference between the highest accuracy scores and the the accuracy scores at epoch=0 is about 3.13% for SimCLR, 0.61% for MoCo, 0.04% for SwAV, 21.54% for BarlowTwins. Except for BarlowTwins, the small difference is not entirely unexpected, as it shows that the self-supervised learning algorithms have learned many common patterns in the world through training on ImageNet, and update a little expertise patterns through training on the kidney biopsy WSI dataset.

Table 5.1: *The state and accuracy of the optimal linear classifier of the four SSL algorithms*

	%labels	**epoch**	**accuracy**
SimCLR	10%	100	88.44%
MoCo	10%	300	89.14%
SwAV	10%	100	91.39%
BarlowTwins	10%	500	82.46%

Another characteristics is about the trade-off between the precision and recall scores. The prediction of glomerulus and vessels for SimCLR and MoCo are particular cases: the precision scores of predicting glomerulus are low, while that of predicting vessels are high. The recall scores of predicting glomerulus and vessels demonstrate otherwise. Nevertheless, the inverse relationship may not seem like much when predicting glomerulus and vessels for SwAV and BarlowTwins and predicting tubules for all the four algorithms. It tells that SimCLR and MoCo are prone to classify some objects as glomerulus, while failed to recognize some vessels. The four algorithms are appropriate at classifying and recognizing tubules reckoning the small number of tubule samples. Albeit having the lowest scores, tubule prediction can be boosted by adding more tubule samples.

The four self-supervised learning algorithms could be ranked by the measurement scores. According to F1 scores, the descending order would be: SwAv, SimCLR, BarlowTwins, MoCo. The SwAV features outperform the other algorithms in the highest F1 scores of classifying the three anatomic structures, about 91.26% for glomerulus, 92.96% for vessels and 86.83% for tubules. Additionally, as aforementioned, the precision and recall scores for SwAV are high and similar, which can infer that the SwAV features recognize and classify the three anatomic structures. The second is SimCLR, whose F1 scores are lower due to the higher difference between precision and recall scores. The BarlowTwins features and MoCo features behave similarly on F1 scores, but the MoCo features have higher difference between precision and recall scores. Noting that as the specialized kidney biopsy WSI dataset is applied, the scores of the BarlowTwins features are moderately improved while that of the MoCo features are slightly deteriorated. As such, we rank BarlowTwins higher for this kidney anatomic structures classification task.

We acknowledge that several limitations in this study and anticipate to provide useful experience for future improvements. First of all, the ResNet-50 backbone can be replaced by the state-of-the art architecture: vision transformer, which is introduced for computer vision tasks in 2020 and demonstrated better performance and greater efficiency on image classification. Second, the number and the distributions of the samples of each classes definitely effect self-supervised learning. We did experiments with smaller kidney biopsy WSI dataset before, which contains 204 glomerulus, 359 vessels and 359 tubules. Aside from the lower scores, the performance of the features is best on tubules while worst on vessels, which is opposite to the results in this thesis. However, in practice the distributions of the unlabelled samples are unknown, so increasing the amount of data might help assure reliability. Furthermore, customized augmentation for kidney WSI might be useful for self-supervised learning, other than the default augmentation for object-centric natural images. Clinicians always pay special attention to kidney WSI, like magnification, density, shape of a tissue.

So long as self-supervised learning performs well on this classification task, evaluation of the experiments could be advanced by fine-tunning on a more specific task, for example, lesion object detection. It might be interesting to explore whether self-supervised learning can leverage the large amount of unlabelled data in actual medical imaging tasks, although it also might be challenging due to subtle difference between diverse patterns as well as the rigorous demands of precision and robustness in the tasks related to clinical diagnosis tasks.

5.2 Conclusion

This thesis examined if self-supervised learning based methods can improve the classification tasks on kidney biopsy WSI, in order to train with fewer labels while maintain good accuracy and other measurement metrics. The evaluation protocol is a linear classifier implemented through multinomial logistic regression with the features extracted by self-supervised learning. Its results of the experiments suggest that yes, it could generate features that learned from data itself without labels. In the linear classifier, the features extracted by the four algorithms all achieve good accuracy scores, higher than 85% with only 10% labels. Among the four algorithms, SwAv performs best, not only on the overall accuracy but also the scores of every class. Then the second is SimCLR. MoCo and BarlowTwins behave similarly, yet BarlowTwins gains more improvement from learning the kidney biopsy WSI dataset. Generally, the thesis confirms the feasibility of self-supervised learning for computer vision tasks about kidney WSI, with saving the high cost of manual annotation on a large scale medical image dataset.

Bibliography

[1] Zbontar, Jure, et al. "Barlow twins: Self-supervised learning via redundancy reduction." International Conference on Machine Learning. PMLR, 2021.

[2] Chuang, Ching-Yao, et al. "Debiased contrastive learning." Advances in neural information processing systems 33 (2020): 8765-8775.

[3] Oord, Aaron van den, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." arXiv preprint arXiv:1807.03748 (2018).

[4] Kleinbaum, David G., et al. Logistic regression. New York: Springer-Verlag, 2002.

[5] He, Kaiming, et al. "Momentum contrast for unsupervised visual representation learning." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.

[6] Chen, Xinlei, et al. "Improved baselines with momentum contrastive learning." arXiv preprint arXiv:2003.04297 (2020).

[7] Bohning, Dankmar. "Multinomial logistic regression algorithm." Annals of the institute of Statistical Mathematics 44.1 (1992): 197-200.

[8] Raghu, Maithra, et al. "Transfusion: Understanding transfer learning for medical imaging." Advances in neural information processing systems 32 (2019).

[9] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[10] Bendjillali, Ridha, et al. "Illumination-robust face recognition based on deep convolutional neural networks architectures." (2020).

[11] Bottou, Leon. "Stochastic gradient descent tricks." Neural networks: Tricks of the trade. Springer, Berlin, Heidelberg, 2012. 421-436.

[12] Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." International conference on machine learning. PMLR, 2020.

[13] Caron, Mathilde, et al. "Unsupervised learning of visual features by contrasting cluster assignments." Advances in Neural Information Processing Systems 33 (2020): 9912-9924.

[14] Van der Maaten, Laurens, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of machine learning research 9.11 (2008).

[15] Loshchilov, Ilya, and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts." arXiv preprint arXiv:1608.03983 (2016).

[16] Webster, J. D., and R. W. Dunstan. "Whole-slide imaging and automated image analysis: considerations and opportunities in the practice of pathology." Veterinary pathology 51.1 (2014): 211-223.

Appendix

The training loss of the linear classifier is not discussed in the body paragraphs because training with low percentage of labels could easily lead to overfitting. However, it has little effect on the conclusions drawn above since the comparisons of the four self-supervised learning algorithms focus on the results on test set but not the fine-tuning on training set. For integrity, the training accuracy scores of the multi-nomial logistic regression trained with 1%, 10%, and 30% labels is supplemented here respectively. The row is indexed by epochs and the column is indexed by the percentage of labels.

Table 5.2: *With SimCLR features, the training accuracy scores of multi-nomial logistic regression trained with 1%, 10%, and 30% labels*

	0	**100**	**200**	**300**	**400**	**500**
1%	92.84%	96.92%	96.94%	97.23%	98.07%	98.46%
10%	90.66%	93.93%	94.03%	94.34%	94.24%	94.65%
30%	91.82%	94.36%	94.65%	94.79%	94.82%	94.74%

Table 5.3: *With MoCo features, the training accuracy scores of multi-nomial logistic regression trained with 1%, 10%, and 30% labels*

	0	**100**	**200**	**300**	**400**	**500**
1%	89.23%	83.08%	84.10%	86.58%	85.12%	84.10%
10%	87.69%	84.08%	84.61%	85.10%	85.29%	85.15%
30%	89.41%	89.41%	89.67%	89.51%	88.95%	88.48%

Table 5.4: *With SwAV features, the training accuracy scores of multi-nomial logistic regression trained with 1%, 10%, and 30% labels*

	0	**100**	**200**	**300**	**400**	**500**
1%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
10%	99.79%	99.43%	98.92%	98.76%	98.45%	98.66%
30%	99.44%	99.31%	98.92%	98.01%	97.75%	97.15%

Table 5.5: *With BarlowTwins features, the training accuracy scores of multinomial logistic regression trained with 1%, 10%, and 30% labels*

	0	**100**	**200**	**300**	**400**	**500**
1%	99.96%	100.00%	100.00%	100.00%	100.00%	100.00%
10%	85.76%	98.51%	100.00%	100.00%	100.00%	100.00%
30%	77.02%	85.51%	98.28%	99.62%	99.76%	99.82%