UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Deep Learning with scarce resources for histopathology image analysis

Rodrigo da Silva e Santos

Mestrado em Engenharia Informática

Dissertação orientada por:
Prof. Doutor Nuno Cruz Garcia

2024

To my family, for their unconditional support and for always believing in my dreams. This achievement is as much yours as it is mine.

Acknowledgments

I would like to express my sincere gratitude to all the people who have made it possible for me to complete this master's thesis.

First of all, I would like to thank my family for the unconditional support they have given me throughout my academic career.

I would like to express my deepest appreciation to my supervisor, Professor Nuno Cruz Garcia, whose guidance and support were fundamental to the development of this thesis.

I would also like to thank the professors and collaborators at the university/institution for the opportunity to carry out this work, as well as for the resources made available to carry out this research.

Finally, I would like to express my gratitude to all my friends and colleagues who have contributed in some way to this work.

This work would not have been carried out without the support and contributions of all these people, and for that I am deeply grateful.

Resumo

Nos últimos anos, a aprendizagem profunda (*deep learning*) emergiu como uma ferramenta de grande relevância para a análise de imagens de histopatologia, sobretudo no que diz respeito à análise de imagens de lâmina completa (*Whole Slide Imaging*, ou WSI). Estas imagens apresentam uma resolução extremamente elevada, frequentemente atingindo dimensões que superam os 100,000 x 100,000 píxeis. Além disso, a complexidade dos tecidos analisados aumenta exponencialmente o desafio, pois exige que médicos especialistas realizem anotações manuais extremamente detalhadas, frequentemente ao nível do píxel. No entanto, este processo é altamente dispendioso em termos de tempo e recursos humanos, sendo particularmente impertinente para especialistas que já enfrentam cargas de trabalho elevadas. Este cenário torna difícil a criação de grandes conjuntos de dados anotados, que são fundamentais para treinar redes neurais com eficiência. Além disso, a variabilidade intrínseca das imagens e a presença de ruído em algumas amostras pode introduzir mais barreiras, fazendo com que o treino de modelos de deep learning dependa de abordagens alternativas para garantir a precisão e generalização eficazes, especialmente em ambientes com recursos escassos e dados limitados.

O objetivo principal deste trabalho foi explorar abordagens alternativas, que reduzam a dependência de dados anotados por especialistas, ao mesmo tempo que mantêm ou melhoram a performance de modelos de deep learning na análise de imagens de histopatologia. Especificamente, este trabalho propõe o uso de técnicas de Aprendizagem de Instâncias Múltiplas (*Multiple Instance Learning*, ou MIL) combinadas com métodos de Aprendizagem Auto-supervisionada (*Self-Supervised Learning*, ou SSL) para superar as limitações tradicionais que se colocam na análise de WSIs. A aprendizagem auto-supervisionada surge como uma solução particularmente promissora neste contexto, uma vez que permite ao modelo aprender representações visuais ricas e robustas a partir de dados não anotados. Isso é conseguido ao transformar as características brutas das imagens em representações que podem ser usadas de maneira mais eficaz pelos modelos de forma que consiga melhorar a performance em várias tarefas, tais como classificação de tumores ou segmentação de tecidos.

Neste trabalho, a abordagem proposta foi implementada através da modificação de uma arquitetura MIL já existente, onde substituímos o extrator de características (*feature extractor*) original por métodos avançados de SSL. O extrator de características tem um papel fundamental, pois é responsável por transformar os dados brutos da imagem em representações que capturam tanto os padrões globais quanto os detalhes locais das WSIs. Essas representações são utilizadas pelo

modelo de classificação para prever com precisão o diagnóstico baseado nas imagens de histopatologia. A introdução de SSL nesta arquitetura permite reduzir significativamente a necessidade de anotações manuais, mitigando o problema do elevado custo de anotação e melhorando a capacidade do modelo de generalizar para novos conjuntos de dados.

Diversos métodos de SSL foram explorados como potenciais extratores de características, sendo SimCLR, MoCo, MoCo v2, DINOv2 e SwAV os modelos implementados para melhorar a classificação de imagens de lâminas inteiras.

Dentre os modelos treinados, o SimCLR destacou-se com um desempenho sólido, alcançando uma precisão máxima de 96.3% e uma precisão balanceada de 96.2%. Este modelo demonstrou ser eficaz como ponto de partida para a exploração de técnicas SSL, mostrando consistência nos resultados. Ao contrário do SimCLR, os resultados com o MoCo não foram tão promissores quanto o esperado. O modelo treinado a partir do zero alcançou uma precisão de 79.6%, sendo a menor registrada de 44.4%. Acredita-se que o tamanho reduzido do conjunto de dados utilizado, 5% do total, contribuiu para o underfitting do modelo, já que a quantidade de dados pode ter sido insuficiente para capturar a complexidade das imagens. Apesar disso, o MoCo ainda apresenta potencial com ajustes adicionais e maior disponibilidade de dados para treino. Utilizando uma versão pré-treinada do MoCo v2, os resultados foram variados, com uma precisão máxima de 87% e uma mínima de 40.7%. Embora o modelo tenha sido pré-treinado por 800 épocas no ImageNet, a diferença entre imagens naturais e de histopatologia parece ter limitado o desempenho do modelo, destacando a necessidade de um pré-treinamento específico em imagens médicas. O MoCo v2 teve melhores resultados com datasets menores, sugerindo que o aumento do tamanho dos dados não trouxe benefícios proporcionais ao desempenho. O modelo baseado em Vision Transformer, DINOv2, apresentou uma precisão máxima de 92.6% e uma precisão balanceada de 93.1%, indicando um desempenho satisfatório. No entanto, a expectativa era de resultados superiores, dado que a arquitetura *Vision Transformer* (ViT) tem mostrado grande potencial em tarefas de visão computacional. A principal limitação identificada foi a quantidade de dados utilizados (25% do total), visto que o modelo exige recursos computacionais significativos para treinamento. Dentre todos os métodos SSL avaliados, o SwAV foi o que apresentou os melhores resultados, com uma precisão de 98.2% e precisão balanceada de 97.9%. O SwAV não apenas superou os demais modelos em termos de desempenho, mas também demonstrou consistência ao alcançar uma precisão acima de 90% na maioria dos experimentos. Isso sugere que o SwAV é uma técnica promissora para a extração de características em imagens de histopatologia, mesmo sem o pré treino em conjuntos de dados de histopatologia.

Os resultados sugerem que, apesar das limitações computacionais e da falta de dados específicos de histopatologia, as técnicas de SSL aplicadas mostraram-se eficazes na extração de características de imagens de WSI. A diferença de desempenho entre os modelos treinados do zero (SimCLR e MoCo) e os modelos pré-treinados (MoCo v2, DINOv2 e SwAV) indicam que há uma vantagem em utilizar modelos pré-treinados, especialmente quando os recursos de hardware e tempo são limitados. Por outro lado, a análise também ressalta que os modelos pré-treinados

em datasets genéricos como o ImageNet, que contém images naturais, de objectos e cenários, nem sempre conseguem capturar adequadamente as características de imagens médicas. Modelos como o MoCo v2 e DINOv2, embora promissores, necessitariam de um pré-treinamento mais ajustado ao domínio de histopatologia para alcançar resultados de estado da arte. O SwAV, por sua vez, destacou-se por apresentar resultados consistentes, sugerindo que, mesmo sem um pré-treinamento específico, a abordagem de SSL utilizada pode ser eficaz quando adaptada adequadamente.

Embora as técnicas de SSL tenham mostrado ser eficientes, elas ainda exigem recursos computacionais significativos, especialmente quando aplicadas a grandes imagens de lâmina completa.

Este trabalho também abre várias oportunidades para investigações futuras. Um caminho promissor seria explorar o uso de pré-treino em dados específicos da área médica, como grandes coleções de imagens de histopatologia já disponíveis em repositórios hospitalares. Ao pré-treinar os modelos em dados específicos do domínio, espera-se que as representações aprendidas sejam ainda mais adequadas para as tarefas subsequentes de classificação. Além disso, a investigação de abordagens híbridas, que combinem SSL com outras formas de aprendizagem, como a aprendizagem supervisionada ou por reforço, poderá resultar em modelos ainda mais eficientes para a análise de imagens médicas.

Em conclusão, este trabalho demonstrou que a aprendizagem auto-supervisionada (SSL), quando combinada com a aprendizagem de instâncias múltiplas (MIL), representa uma estratégia promissora para a análise de imagens de histopatologia. Ao mitigar a necessidade de dados anotados, estas técnicas abrem novos horizontes para o desenvolvimento de sistemas de diagnóstico auxiliado por computador (CAD) mais eficientes e acessíveis. Estes avanços têm o potencial de transformar a prática clínica, tornando os sistemas CAD uma ferramenta indispensável no diagnóstico e tratamento de doenças, ao mesmo tempo que reduzem a carga de trabalho dos especialistas médicos.

Palavras-chave: Aprendizagem Auto-Supervisionada, Aprendizagem de Instâncias Múltiplas, Aprendizagem Profunda, Images Histopatológicas, Imagens de Lâmina Completa

Abstract

Computer-Aided Diagnosis is crucial in pathology, notably with the introduction of whole slide imaging (WSI). Deep learning excels in WSI analysis, yet challenges persist due to high resolution (e.g., 100,000 x 100,000 pixels) and labor-intensive pixel-level annotations.

Recent advancements in Deep Learning have proven to be effective to address these issues. More specifically, Multiple Instance Learning (MIL) and Self-Supervised Learning (SSL) are the current state-of-the-art methods to overcome the limitations associated with WSI analysis. MIL is ideally suited for histopathology image analysis, as it enables to learn from just WSI labels without requiring the need for pixel-level annotation. Additionally, SSL has proven to excel at extracting rich and meaningful representations from large unlabeled datasets which is specifically well-suited for high resolution WSI since it doesn't demand pixel-level annotations.

This work is build on a prior architecture that uses both methods, in which we extend the research by incorporating other state-of-the-art SSL techniques that outperform the current SSL method employed, such as MoCo, DINO and SwAV. We trained from scratch some of the SSL models implemented, while the others, we used their pre-trained version due to computational bottlenecks. Even though there is downsides regarding using pre-trained models in which were pre-trained on different domains, some of the best results we got was from pre-trained models, proving once more that SSL is a promising technique when dealing with lack of annotations.

Keywords: Self-Supervised Learning, Multiple Instance Learning, Deep Learning, Histopathology Images, Whole-slide Imaging

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and motivation

Nowadays, Computer-Aided Diagnosis (CAD) is key for a successful diagnosis. CAD algorithms have begun to be developed for disease detection, diagnosis, treatment planning, and prognosis prediction to complement the opinion of the pathologist [20]. The introduction of whole slide imaging (WSI) technology has brought a major change in the field of pathology.

Traditionally, pathologists relied on microscopic examination of physical glass slides to diagnose diseases. These slides captured only a small portion of the tissue sample at a time which requiring them to meticulously scan the entire slide at various magnifications to build a comprehensive image.

The arrival of WSI offered a digital and comprehensive view of the entire tissue section making it more efficiently for pathologists since it is more practical, they can remotely review slides without having to handling them physically, and thus allows researchers and pathologists from different institutions around world to collaborate together.

However, it has some limitations. The whole-slide images can reach enormous sizes, which require more space and analyzing these high-resolution images require powerful computing resources. Although WSI can help and be beneficial for pathologists, it doesn't replace the need for their expertise. We still need highly trained pathologists to perform analysis and interpret these highly structured complex tissues.

Additionally, the majority of medical experts deal with demanding schedules, hard workloads and time constraints, which the human cognitive process can be easily disturbed leading to incomplete diagnosis and misdiagnosis [33]. Therefore, CAD is becoming more important than ever.

In recent years, Deep Learning (DL) has provided state-of-the-art results in a wide variety of image analysis tasks [29], consequently DL has emerged as a powerful tool for automating WSI analysis. Most of the DL techniques needs labels, and thus we need medical experts to annotate WSIs.

However, one particular challenge when dealing with WSIs, is the large resolution (it can extend to 100,000 x 100,000 pixels) which leads to two main problems (1) it makes labor-extensive

and time-consuming job for high expertise medicals to annotate WSIs, and (2) to train deep neural networks in an end-to-end manner. One example in the analysis of WSIs is a process called segmentation which is a crucial task that deals with extraction of regions of interest (ROI), consequently requiring accurate pixel-level annotations to identify and outline specific structures and regions within WSIs. Hence, working with DL techniques that require less annotations or even it doesn't require at all is essential to take the burden of medical experts and reach closer to the fully automation of WSI analysis. Therefore, addressing the limitations of deep learning for WSI analysis in resource-scarce environments is essential for its adoption in CAD.

1.2 Objectives and contributions

This thesis aims to address the limitations imposed in histopathology image analysis. The main challenge is the high resolution of WSI and the limitation of pixel-level annotation. To address these limitations, we will employ Deep Learning techniques that are well-suited for histopathology image analysis. More precisely, we will use Multiple Instance Learning (MIL) and Self-Supervised Learning (SSL).

We extend the work presented in the paper "Dual-stream Multiple Instance Learning Network for Whole Slide Image Classification with Self-supervised Contrastive Learning" [31] by implementing other SSL methods to further improve its WSI classification method. Through enhancing the feature extractor, it improves the representations underlying the whole-slides images, consequently improving the input images to the classification model. The model will understand better the input and potentially get better results on classifying whole-slide images.

The main contributions of this thesis are:

- Provide an overview of histopathology image analysis using Deep Learning techniques.
- Explain how Multiple Instance Learning is the go-to approach in Deep Learning to deal with whole-slide images.
- Explore Self-Supervised Learning techniques that outperform the original feature extractor to further replace it in the architecture we built upon [31].
- Pre-process the Patch Camelyon datatset.
- Implement state-of-the-art Self-Supervised Learning techniques by training some of them from scratch and others by using their pre-trained version.
- Analyze the results from the different methods implemented (i.e., MoCo [24], MoCo v2 [12], DINOv2 [37] and SwAV [6]).

1.3 Dissertation outline

This thesis is organized as follow:

- Chapter 2 - The Background and Related Work chapter is where its explained important concepts and state-of-the-art works on multiple instance learning and self-supervised learning.
- Chapter 3 - The Methodology chapter presents the framework we will built upon and the self-supervised learning methods we are going to implement.
- Chapter 4 - The Experiments chapter shows the dataset chosen, how we implemented several self-supervised learning methods, and the training protocol for our main model.
- Chapter 5 - The Results and Discussion chapter is where its showcased the results obtained.
- Chapter 6 - The Conclusion and Future Work chapter presents a summary of the work done and our thoughts on how to proceed further research.

Chapter 2

Background and Related Work

In this chapter, we will present important concepts to fully understand this work. Additionally, we will present related work on Multiple Instance Learning and Self-Supervised Learning.

2.1 Histopathology Image Analysis

First and foremost, histopathology is the study of diseases by examining tissues and/or cells under a microscope for interpretation of tissue specimens in order to identify and characterize diseases. In order to visualize the different components within the tissue, the sections are dyed with one or more stains. Typically, it is used hematoxylin and eosin (H&E). Hematoxylin is a basic dye that stains proteins a blue color, and eosin stains a pink color, resulting in images of tissue samples with a purple color [22]. The goal of staining is to highlight cellular components (e.g., intracellular organelles and proteins). Nowadays, histopathology image analysis is critical in the field of medical imaging, particularly for clinical cancer diagnosis, prognostic analysis and treatment response prediction.

Computer-Aided Diagnosis (CAD) is systems that assist and complement the medical experts opinion on various tasks including diagnoses. It leverages the power of image processing by analyzing and interpret insights from medical images that might go unnoticed by the human eye. In reality, CAD has become one of the major research subjects in medical imaging [21]. While CAD systems are becoming more relevant, it is important to mention that the technology that helped CAD systems to reach its importance is the digitization of glass slides.

2.1.1 Whole Slide Imaging

The introduction of whole-slide imaging (WSI), also known as ”virtual microscopy”, aims to emulate the traditional light and view of a tissue under the microscope. First, it needs a specialized hardware (i.e., a scanner) to digitize glass slides, which yields a large digital image of the tissue sample. These so-called ”digital slides” illustrates the entire slide at high resolution and at various magnifications, eliminating the need for physical slides. Second, it needs specialized software (i.e., virtual slide viewer) to view and/or analyze these large digital slides. These software facilitate image management, sharing with other colleagues, and performing image analysis, allowing

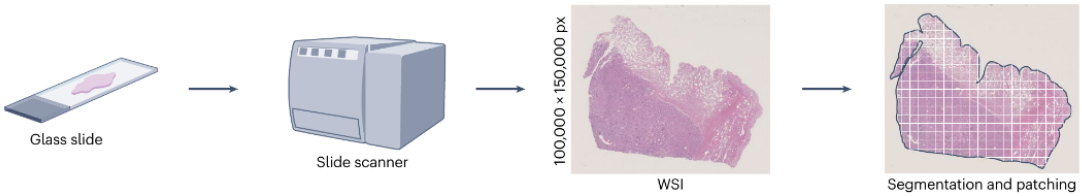

Figure 2.1: Tissue acquisition and patching. The first step where glass slides are digitized with a WSI scanner, which subsequently go through patching and segmentation. From [41].

clinicians to navigate digital slides (e.g., zoom and pan) on a digital screen by reproducing the traditional light microscopy experience [18]. However, WSI files are larger than common digital image files requiring comprepression-decompression tools to archive these massive digital files (e.g., a virtual slide with 1600 megapixels would require about 4.6 GB of memory). One of the most common image file formats used to store WSI is TIFF (i.e., Tagged Image File Format), yet there is a lot of image file formats to use. Consequently, software translation libraries have emerged to read the data stored across many file formats into a common open representation via an API (i.e., Application Programming Interface). Currently, one of the most established libraries used in WSI domain is OpenSlide, a C library which provides a simple interface to read WSI. By employing this technology, it enables the application of image techniques and image analysis tools to aid medical experts in the examination of WSI.

In the past years, deep learning-based techniques have proven state-of-the-art results in a variety of image analysis tasks, including whole-slide imaging analysis, and thus the widespread usage of deep learning-based methods to automate WSI analysis has hugely increased. However, deep learning-based solutions have shown many technical challenges [28, 42]: (1) large dimensionality of whole-slide images (e.g., resolutions can be up to 100,000 x 100,000 pixels), while deep neural networks operate on much smaller dimensions (e.g., 224 x 224 pixels), making it hard to train deep neural networks in an end-to-end manner; (2) scarcity of labeled data, whereas most deep learning-based algorithms require vast amounts of curated data whereby it needs medical expertise to annotate the data, specifically to delineate and localize with precision, hence we get pixel-level annotation on whole-slide images. Although, it is a costly, time-consuming and tedious process. Additionally, most of the times the data is based on small samples sizes with limited geographic variety, leading to algorithms with poor generalization [28]; (3) stain variability across laboratories since there is numerous stain procedures such as hematoxylin and eosin (H&E), gram stain, giemsa tain, periodic acid schiff reaction, prussian blue, and others. Hematoxylin and eosin is one of the most common stains and can be found on most WSI datasets. Hematoxylin stains cell nuclei blue, while eosin stains cytoplasm and connective tissue pink [21], resulting in whole-slide images with purple color. Staining is a form tissue preparation before it goes under examination. The goal is to highlight important structures within the tissue and different stains are used to highlight particular proteins or important features within a tissue type [22].

2.2 Deep Learning

Deep Learning (DL) is a subfield of machine learning that employs artificial neural networks (ANN), which are inspired by the human brain. ANNs works by having an input layer receiving raw data, followed by one or more hidden layers connected one after the other that process and transform the input data through a series of nonlinear transformations, finally leading to the output layer which delivers the model's prediction. The term "deep" refers to the multiple hidden layers a model can have.

In the past decade, deep learning gained a lot of followers and hope into it. The notorious fame of DL set off when AlexNet [30] achieved a staggering victory in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [14]. AlexNet uses a common architecture within DL, called Convolutional Neural Network (CNN). CNN is a neural network model architrecture that uses convolutional layers and its known for their high performance, specially in computer vision and feature extraction. After enormous victory, the use of CNNs exploded, and several researchers started to affirm that "the deeper the better" by stacking many layers making the neural networks "deeper". As a result, it started the famous problem, vanishing gradients. This happens when the network is too deep, and the gradients become so small to the point that the update of the parameters become ineffective, therefore affecting the learning process of the neural network. This problem was further tackled by a new architecture called Residual Network or simply known as ResNet [23], with the introduction of Residual Blocks. ResNet introduces a new technique called skip connections whereby connects activation of a layer to further layers by skipping some layers in between, which forms the so called residual block. ResNets are composed of residual blocks that are stacked together. This architecture allows to train deeper networks without dealing with the issue mentioned earlier.

Another remarkable architecture that revolutionized this field is Transformers. More particularly, it revolutionized the Natural Language Processing field and further more the Computer Vision field within the DL field. When the Transformer model was first released, Recurrent Neural Networks (RNN), long short-term memory (LSTM) [25], and gated recurrent neural networks were the dominant approaches for dealing with sequential data.

As the Transformer model architecture were introduced [43], it began a shift from RNNs to Transformers in NLP, driven by the fact that RNNs process data sequentially meaning it doesn't capture global properties and cannot use parallel computation, and thus making the training slower. Also, it has issues that we talked earlier such as vanishing gradients and exploding gradients problems. The Transformer model solves this problems and instead of using recurrency, it relies on an attention mechanism that can draw global dependencies between input and output [43]. Also it allows for significantly more parallelization making it more efficient in use of GPUs (i.e., graphic processing units). Nowadays, the Transformers model is the new go-to approach to solve many tasks in NLP, such as text summarization, text generation, question-answering, translation, and many more.

Even in Computer Vision, it was developed the **Vi**sion **T**ransformer (ViT) [15], a Transformer-

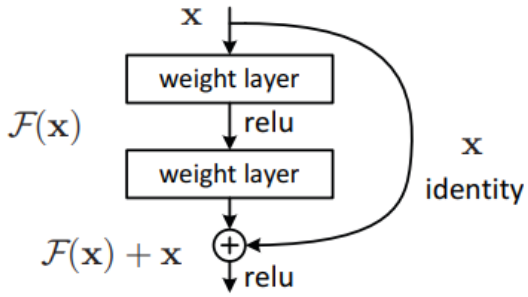

Figure 2.2: Residual Block [23].

based model specialized on image tasks. ViT is inspired on the Transformer architecture, relying on the same attention mechanism. Still, there's some differences between the two models. Instead of using an encoder attention branch and a decoder attention branch, ViT uses only the encoder attention branch, which then yields the output to a single linear layer as the head to perform the prediction. Also, ViT requires pretraining on large datasets to achieve an optimal performance.

One main challenge remains in DL, which is the need of annotations. Creating high quality annotations requires high expertise and is extremely time-consuming. As mentioned earlier, there's models that need huge datasets to be trained on to achieve great performance, and especially when dealing with medical images (e.g., whole-slide images). To address this issue, there is other DL paradigms that doesn't require a huge amount of annotations and can work in scarce label environments. On this work, we will take advantage of those DL paradigms, which are Weakly Supervised Learning (WSL) and Self-Supervised Learning (SSL).

2.2.1 Weakly Supervised Learning

Weakly Supervised Learning (WSL) is a paradigm of machine learning that works with limited or noisy labels. Unlike traditional supervised learning that demands high quality annotated data, weakly supervised learning allows models to learn from data that might be noisy, insufficiently annotated, or even obtained by an unreliable source. Hence, this approach being a popular choice for alleviating the annotation bottleneck in machine learning. WSL encompasses three types of weak supervision: incomplete supervision, inexact supervision and inaccurate supervision. This work aims to use inexact supervision as a weak supervision form. Inexact supervision relates to learning from imprecisely labeled data such as coarsely-grained label, which is when we only have image-level labels instead of object-level labels [50]. For this work, we will use Multiple Instance Learning as the weakly supervised method.

Multiple Instance Learning

Multiple Instance Learning (MIL) is a form of WSL and is currently the state-of-the-art for solving histopathology classification tasks. In MIL, training instances are arranged in sets, which are called bags, and instead of providing a label for each instance, a single class label is provided for the entire bag. There are two main MIL approaches: instance-level approach and embedding-level

approach. Instance-level approach focus on using an instance classifier f that scores each instance, then a pooling operator g aggregates the instances scores resulting in a bag score.

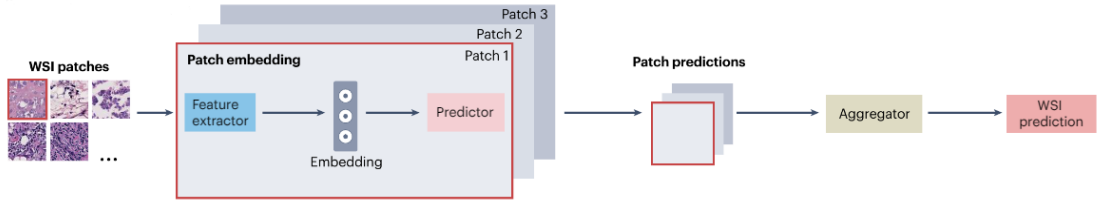

Figure 2.3: Instance-level approach of MIL. From [41].

On the other hand, embedding-level approach starts by using f as a feature extractor, which maps each instance to a low-dimensional embedding, followed by a MIL pooling operator g that generates the bag embedding, and thus further processed by a bag-level classifier to output a bag score.

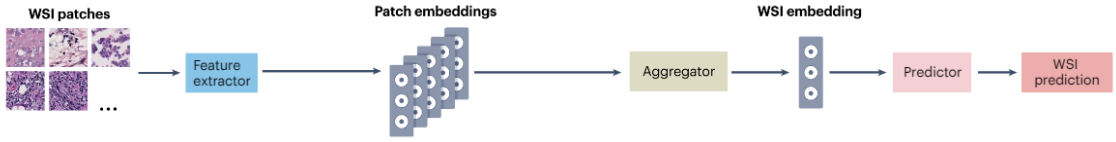

Figure 2.4: Embedding-level approach of MIL. From [41].

The standard MIL assumption states that positive bags contains at least one positive instance, whilst negative bags contains only negative instances. However, in some cases, to be considered a positive bag, it needs multiple positive instances. For instance, in a scenario with images of a road where the model needs to detect if it has traffic jam or not, a car would be a positive instance. Yet, it takes many cars to create a traffic jam [3].

In the context of this work, each patch is an instance and each whole-slide image is a bag meaning we only have the label for the whole-slide image. In our case, whenever a patch is positive (i.e., it contains malignant tumor), the whole-slide image associated with the patch is considered positive, therefore it contains malignant tumor.

Let Y be the label of a bag X, the label of the bag is given by:

$$Y = \begin{cases} +1 = if \exists y_i = +1 \\ -1 = if \forall y_i = -1 \end{cases} \tag{2.1}$$

2.2.2 Self-Supervised Learning

Self-Supervised Learning (SSL) is a subcategory of unsupervised learning in machine learning. The key idea is to let the model learn underlying structures and representations of the data without labels - this is called the pretext task (also called auxiliary task or representation learning). The pretext task is the initial process whereas a model is pre-trained with no labels to learn meaningful features and patterns of the data in a supervision fashion by automatically generating supervision

signals (also called pseudo-labels) from the unlabeled dataset. There is several pretext tasks that a model can be pre-trained on, such as Innate relationship, Generative, Self-prediction, Contrastive learning, and others. Although, this is not the main goal. The true interest relies on the next task of the SSL pipeline, called downstream task. This is where the concept of Transfer Learning comes to play, more specifically fine-tuning. The downstream task revolves around the concept of transfer learning, which the learned representations from the pretext task are transferred to a specific downstream task via fine-tuning. Unlike the pretext task that does not need labels, downstream task needs labeled data, yet does not need large quantities as if it were needed for training a model from scratch. The latter task is the one we want to perform, which can range from image classification, image segmentation, object detection, and among others within the field of computer vision to all the way to Natural Language Processing (NLP) with tasks such as documentation processing, sentence completion, text suggestions, and many more. In our work, we want to perform image classification. Compared to supervised learning, SSL can provide generalist models that can be further fine-tuned for many downstream tasks without the need for large curated datasets.

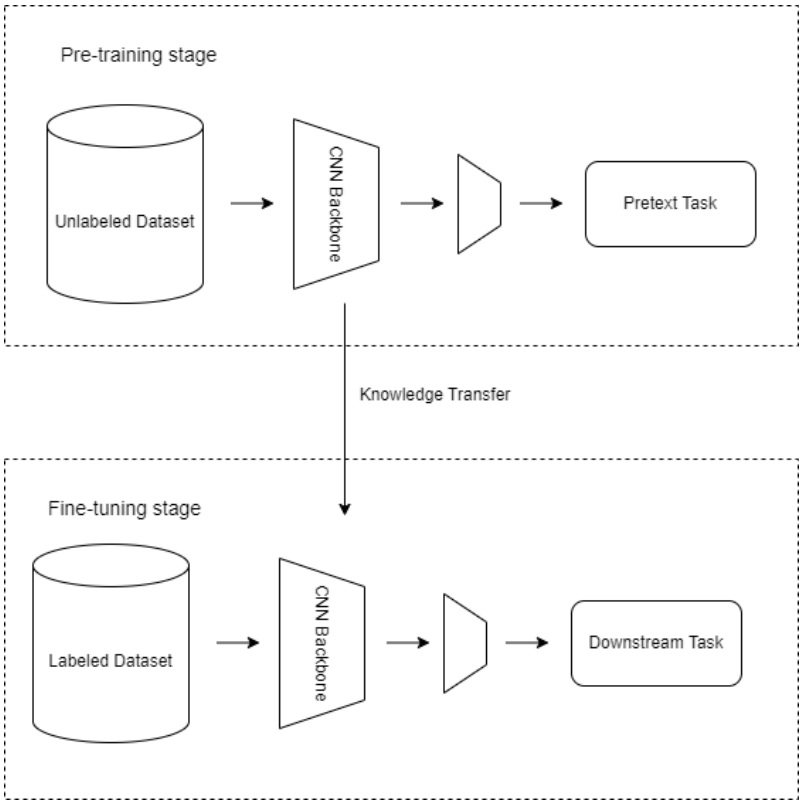

Figure 2.5: Self-Supervised Learning pipeline. The pre-traning stage is where the model learns representations from the data by self-generating supervising signals through a pretext task. The fine-tuning stage is where the model is evaluated through a downstream task.

Constrastive Learning

Contrastive Learning (CL) is one of many strategies that SSL can use as a pretext task to learn useful representations on unlabeled data. The core idea is to create positive pairs and negative pairs while teaching the model how to distinguish them. Positive pairs (or similar pairs) are two transformed views of the same image whilst negative pairs (or dissimilar pairs) are any other images different from the transformed views. The positive examples are generated from a set of random augmentations of an input image (e.g., color jittering, flip, rotation, and many others). This makes the positive pairs slightly different because it alters the image appearance, but preserves the global features and its semantic content (e.g., a flipped dog is still a dog). Afterwards, the model is trained to maximize the similarity between the positive pair and minimize it with the negative pairs. CL is a recent approach which gained popularity for its performance on developing robust representations from the input data by maximizing agreement between similar views of the same image. There are several contrastive learning approaches, yet on our work we will use SimCLR [9], MoCo [24], MoCo v2 [12], SwAV [6], and lastly DINOv2 [37].

2.2.3 Transfer Learning

Transfer Learning is a machine learning technique that uses a model previously trained on a task as a starting point to train a model on a second task. Instead of training a model from scratch, this technique leverages the knowledge gained (e.g., rich features and representations) from solving one problem to improve generalization in another task. There are two stages: pre-training and fine-tuning. Pre-training is the first stage where the model is trained from scratch using a large dataset. Upon the train is complete, we freeze the initial layers so we can use on the next stage (i.e., fine-tuning) as a starting point. Usually, the pre-trained model learns general features that can be transferred to other domains. Next, it comes the fine-tuning stage whereby we initialize with the pre-trained models' weights and proceed to further training on the target task.

This approach allows to train more efficiently leading to reduced training time by avoiding to train from scratch which enables to fine-tune on smaller datasets, consequently reducing operational costs. Also, it reduces overfitting and improves reusability for a numerous tasks by allowing the model to adapt to different scenarios. This technique helps alleviate the scarce-annotation environments. For instance, Large Language Models (LLMs) requires a large set of data to achieve optimal performance, whilst ViTs (Vision Transformers) have the same issue. In the context of this work, this strategy allows to work with ViTs without the need to train from scratch, which we have neither the computational resources nor the large amount of data required. This method is also used on self-supervised learning and on feature extracting making this concept important for our work.

2.2.4 Feature Extractor

Feature extractor is a component of a machine learning model that is capable of learning relevant patterns and useful features from raw input data. Feature extraction is the backbone of computer

vision. It involves the identification and extraction of distinctive structures within an image. These structures are characteristics of an image that helps algorithms to distinguish one image from another, and can range from simple edges and corners to more complex patterns and textures. The goal of this process is to transform raw pixel data into a set of representations that are more informative, non-redundant and compact which can be effectively further utilized by machine learning algorithms. The importance of feature extraction lies in its ability to enhance the performance of models by reducing its dimensionality, consequently reducing noise by discarding redundant and repeated information, while reducing the risk of overfitting risk and reducing multicollinearity which can negatively impact the model's performance on unseen data, and many more benefits. Traditionally, machine learning relied on handcrafted feature engineering. This can range from feature selection whereby statistical methods such as correlation analysis is used to obtain a subset of the most relevant features from the original data, to feature transformation through which is applied mathematical functions to represent existing features to be more suitable for the algorithm. For example, principal component analysis (PCA) is a common technique to reduce the dimensionality of a dataset while retaining most of its important features. Lastly, handcrafted feature engineering which requires deep domain expertise to identify and extract important features that may boost the model's performance, yet this can be time-consuming, labor-intensive, and prone to errors.

Feature extraction became more interesting when deep learning came to play. DL revolutionized feature extraction by introducing powerful neural networks that can automatically learn features directly from data. In the computer vision field, the most common methods to perform image feature extraction are convolutional neural networks (CNNs), recently vision transformers (ViTs), autoenconders, and self-supervised learning methods.

CNNs excel at analyzing visual data through the use of convolutional layers with filters that automatically learn to detect edges, patterns and shapes in images. Nowadays, CNNs are the chosen backbone for computer vision tasks for its ability to extract local features. However, it diminishes the image resolution, consequently it may loose some spatial information caused by information skipping (e.g., strided convolution, global average pooling, and max pooling), while studies have shown that ViTs show stronger preservation of spatial information [38]. This leads to ViTs challenging the dominance of CNNs in image tasks. The ViT is based on the same attention mechanism as the original transformer architecture, and since the transformer was design to take a sequential input, ViT transforms an input image onto a fixed-size patches, then these patches are flatten and treated as tokens for the transformer encoder. This allows ViTs to capture long-range dependencies more effectively than CNNs.

Another DL technique for feature extraction is autoencoders. Autoenconders are composed by an encoder and a decoder (encoder-decoder architecture). An encoder compresses the image into a smaller representation, capturing the key features, while the decoder uses the compressed representation to reconstruct the original image, which in essence by trying to reconstruct the image, it learns a compressed representation which highlights the essential features.

Finally, self-supervised learning is a paradigm that has gained popularity in the feature extraction process. SSL has become a prominent technique, especially when dealing with annotation-scarce environments which is the case of medical imaging, thereby leads to reducing the number of manual annotations required to train a medical imaging model, resulting in reduced both the cost and time required for labeling the dataset and consequently model development [26]. For that reason, the use of SSL in the medical imaging field has grown exponentially over the last decade, as shown in the figure 2.6.

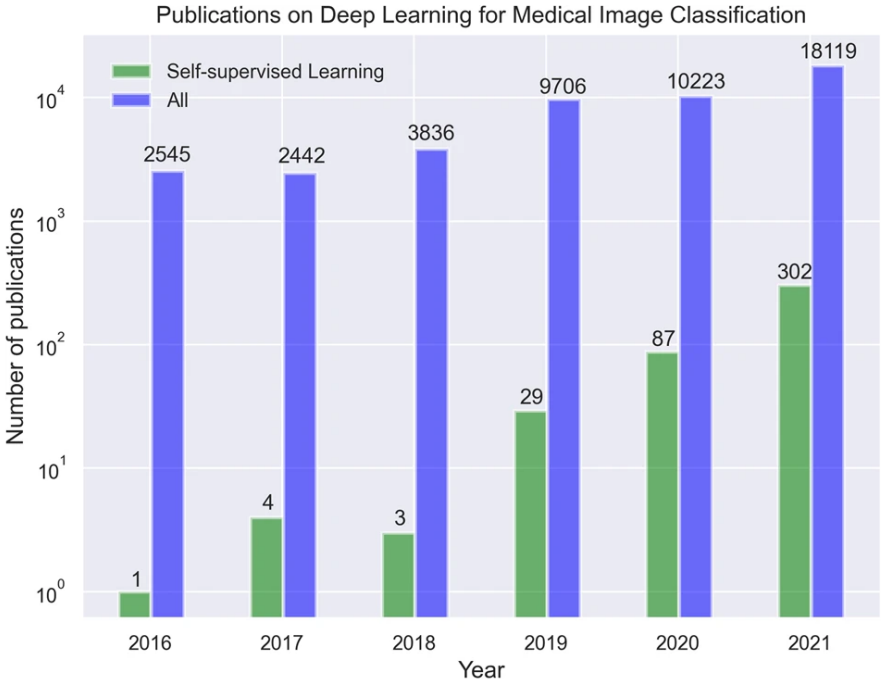

Figure 2.6: Timeline showing the number of publications on deep learning applied to medical image classification per year. The figure also shows the rapid grow of self-supervised learning methods applied to medical image classification. From [26].

2.3 Multiple Instance Learning

Nowadays, most of the state-of-the-art methods follow an attention based approach, whereby the fundamental work is AB-MIL [27]. AB-MIL is a classic approach that presented a MIL fully parameterized by neural networks. By using neural networks to parameterize functions, the model can be trained in an end-to-end manner by backpropagation and allowing the model to be more flexible. Also, the authors presented a trainable MIL pooling based on the attention mechanism. They use a trainable weighted average of instances where weights are determined by a two-layered neural network. The proposed attention-based MIL pooling enables the assignment of various weights to instances inside a bag, and thus the final representation of the bag may be quite informative for the bag-level classifier, which makes a very flexible operator. Aditionally, the attention mechanism allows to easily provide ROI, by creating a heatmap by multiplying the patches by its

corresponding attention weight, making the operator easily to interpret. Therefore, the attention mechanism could potentially be useful for pratical applications where its more beneficial to provide ROIs in addition to the final diagnosis. Moreover, in [48] a derivation of instance probability under the similiar attention mechanism is employed, which serves for feature destilation using the idea of pseudo bags and a double-tier process. Furthermore, TransMIL [40] presented a Transformer based MIL which they explore both the spatial and morphological information among the instances. The authors of CLAM [34] proposed a variation of AB-MIL, while also constraining and refining the feature space by performing an instance-level clustering over the sub-regions of high diagnostic value identified. Other approaches used recurrent neural networks (RNN) [2], which tile-level feature representations at different scales are fed into RNN to incorporate the information across the whole slide and deliver the final classification result.

2.4 Self-Supervised Learning

Nowadays is very common to use self-supervised learning approaches [9, 24, 10, 12, 1, 4, 7, 47, 19, 5, 37, 49] to capture higher-level semantic information on vast amounts of unlabeled data, especially in medical imaging since its an expensive and time-consuming task to acquire high-quality annotations. Despite the scarcity and high cost of labeled data, often exists large amounts of cheap unlabeled data remained unexploited [11]. In turn, SSL can leverage these unlabeled data and learn general good representations of data.

A large body of work on self-supervised learning focuses on instance classification [16, 46, 9, 24] and contrastive learning [9, 24, 10, 12]. However, usually it requires either a large batch [9] or memory banks [24, 46]. Another approach is forming group of instances such as clustering [1, 4, 7, 47].

2.4.1 SSL for learning the Feature Extractor

There are many strategies to perform feature extraction. Typically, the feature extractor is a CNN pre-trained on a large natural images dataset such as ImageNet [14, 34, 40]. The architecture of CNN used have been range from large convolutional kernels (for instance AlexNet[30]), to deeper CNNs with residual blocks and smaller kernels (such as ResNet) [23], and finally vision transformers [15]. [41] Yet, the gap between WSIs and natural images may lead to sub-optimal representations, and thus in recent years, SSL methods are becoming the norm for pre-training feature extractors, showing that it provides more robust and generalized representations than supervised learning methods [41, 45, 13, 24].

Chapter 3

Methodology

In this chapter, we will demonstrate the work developed that takes advantage of the scarce-annotation environment which is the case of histopathology image analysis field. We take a state-of-the-art multiple instance learning framework [31], which uses a self-supervised learning technique, SimCLR [9], to learn the feature extractor so it potentially gets better results on WSI classification. However, there are various newest state-of-the-art self-supervised learning methods that outperform SimCLR. Therefore, we will replace SimCLR with other state-of-the-art SSL methods that outperform the original feature extractor. The following figure 3.1 is an overview of the framework we will be built upon [31] whereby the region delineated by green represents the part we will modify.

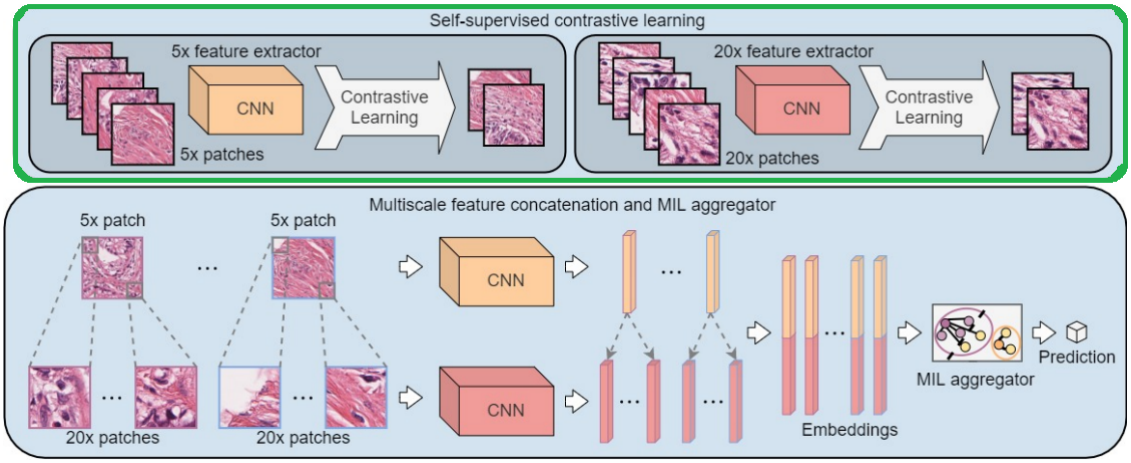

Figure 3.1: Overview of DSMIL from [31]. Separately, patches extracted from each magnification of the WSIs are used for self-supervised contrastive learning, to further compute embeddings of patches. Then, the embeddings of different scales are concatenated into feature pyramids to train the MIL aggregator. This work will focus on the self-supervised contrastive learning, delineated by green, which represents the learning of the feature extractor.

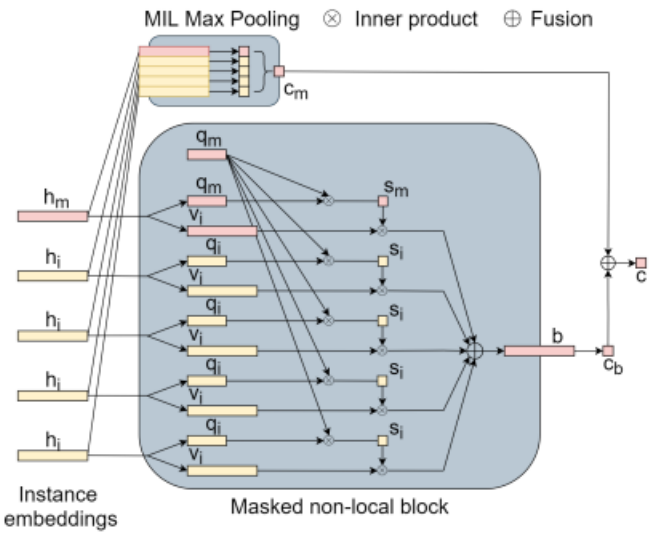

Figure 3.2: MIL aggregator of DSMIL. The max-pooling branch calculates the critical instance by max pooling the instance scores. The distances between each instance and the critical instance are measured by the aggregation branch, which then uses the distances as weights to sum the instance embeddings and create a bag embedding. The final score is produced by averaging the scores of both streams.

3.1 DSMIL: Dual-stream multiple instance learning networks for tumor detection in Whole Slide Image

DSMIL [31] is a novel deep MIL model called "Dual-stream Multiple Instance Learning Network for Whole Slide Image Classification with Self-supervised Contrastive Learning". The major innovations in DSMIL framework are the design of a new aggregation function g, which consists of a masked non-local block and a max-pooling block for feature aggregation. For the feature extractor f, the authors used a self-supervised contrastive learning framework, specifically SimCLR [9] to learn instance embbedings which are further fed into g. Additionally, DSMIL combines multiscale embeddings using a pyramidal strategy, that guarantees the local constraints of the attentions for patches in a WSI. Our work is largely inspired by DSMIL, which we will go over each component in the following.

MIL Aggregator with Masked Non-Local Operation

DSMIL [31] applies a two-stream architecture to learn jointly the instance (patch) and a bag (image) classifier. A bag of patches of a WSI is denoted as $B = \{x_1, \ldots, x_n\}$. Given a feature extractor f, each instance x_i can be transformed into an embedding $h_i = f(x_i) \in \mathbb{R}^{L\times 1}$.

The first stream uses an instance classifier on each instance embedding, which is followed by a max-pooling operator on the scores:

$$\begin{aligned} c_m(B) &= g_m\left(f\left(x_i\right), \ldots, f\left(x_n\right)\right) \\ &= \max\left\{W_0 h_0, \ldots, W_0 h_{N-1}\right\} \end{aligned} \tag{3.1}$$

where W_0 is a weight vector. This stream determines the instance with the highest score, called critical instance.

The second stream aggregates the instance embeddings into a bag embedding, which is then attributed a score by a bag classifier. Its obtained the embedding h_m of the critical instance, and transform each instance embedding h_i, including h_m, into two vectors, query $q_i \in \mathbb{R}^{L\times 1}$ and information $v_i \in \mathbb{R}^{L\times 1}$, which are as follows:

$$q_i = W_q h_i, \quad v_i = W_v h_i, \quad i = 0, \ldots, N-1 \tag{3.2}$$

where W_q and W_v each is a weight matrix. Next, we define a distance measurement U between an arbitrary instance to the critical instance, described as:

$$U\left(h_i, h_m\right) = \frac{\exp\left(\langle q_i, q_m\rangle\right)}{\sum_{k=0}^{N-1}\exp\left(\langle q_k, q_m\rangle\right)} \tag{3.3}$$

Additionally, using the distances to the critical instance as the weights, we can determine the bag embedding b which is the weighted element-wise sum of the information vectors v_i of all instances, that can be described as:

$$b = \sum_{i}^{N-1} U\left(h_i, h_m\right)_{v_i} \tag{3.4}$$

Then, the bag score c_b is given by:

$$\begin{aligned} c_b(B) &= g_b\left(f\left(x_i\right), \ldots, f\left(x_n\right)\right) \\ &= \mathbf{W}_b \sum_{i}^{N-1} U\left(\mathbf{h}_i, \mathbf{h}_m\right)\mathbf{v}_i = \mathbf{W}_b\mathbf{b} \end{aligned} \tag{3.5}$$

where W_b is a weight vector for binary classification.

Although this process is similar to self-attention [43], it differs in the query-key matching, since it is only performed between the critical instance and the other instances (including the critical instance itself). Furthermore, the dot product measures the similarity between two queries, consequently instances that are more similar have higher values, and thus instances more similar to the critical instance will be given higher attention weights. The final bag score is the average of the scores of the two streams, described as:

$$\begin{aligned} c(B) &= \frac{1}{2}\left(g_m\left(f\left(x_i\right), \ldots, f\left(x_n\right)\right) + g_b\left(f\left(x_i\right), \ldots, f\left(x_n\right)\right)\right) \\ &= \frac{1}{2}\left(\mathbf{W}_0\mathbf{h}_m + \mathbf{W}_b\sum_{i} U\left(\mathbf{h}_i, \mathbf{h}_m\right)\mathbf{v}_i\right) \end{aligned} \tag{3.6}$$

While the distance measurement applies an inter-instance selection based on the similarity to the critical instance, the information vector v_i employs an intra-instance feature selection. Figure 3.2 shows the architecture of the aggregator.

SSCL framework for learning the feature extractor.

Moreover, the authors proposed using a self-supervised contrastive learning (SSCL) framework for learning the feature extractor f. They used SimCLR [9], which is a state-of-the-art self-supervised contrastive learning framework. SimCLR employs a contrastive learning strategy

whereby a CNN is trained to identify sub-images from the same image. This is done by maximizing the agreement between the augmented images using a contrastive loss.

Locally-Constrained Multiscale Attention.

The final component of DSMIL is the Locally-Constrained Multiscale Attention. They used a pyramidal concatenation strategy to combine features of WSIs from various magnifications. For that, its extracted the feature vector of the patch at low magnification and the feature vectors of the sub-images at higher magnification within each low magnification patch. For example, considering a patch with size of 224x224 at 5x magnification, it will contain 4 sub-images of size 224x224 at 10x magnification. Therefore, its concatenated the feature vector at 5x magnification with each feature vector at 10x magnification, resulting in 4 feature vectors. One of the benefits that come from this strategy is that it enables the network to choose the relevant information v_i across different scales since the information from different magnifications are preserved in the feature vectors.

3.2 SimCLR: Simple Framework for Contrastive Learning

Contrastive learning has proven to be a powerful approach to leverage and improve performance when there's limited labeled data, which is the case in WSIs analysis. SimCLR [9] employs a strategy that employs a maximization of an agreement between different augment versions of the same data example. This framework has four major components: a stochastic data augmentation module, a base encoder, a projection head, and a contrastive loss function. The stochastic data augmentation module works by randomly transforming any given image into two augmented versions of the same image, denoted $\tilde{x}_i$ and $\tilde{x}_j$, which its considered a positive pair. The process of the transformation is a combination of a random crop, random color distortion, and a random Gaussian blur. The base encoder $f(.)$ is a neural network that extracts representation vectors from the augmented instances processed earlier. Specifically, they selected the ResNet-50 [23] as the base encoder network, therefore $h_i = f(\tilde{x}_i) = ResNet(\tilde{x}_i)$ where $h_i \in \mathbb{R}^d$ is the output after the average pooling layer.

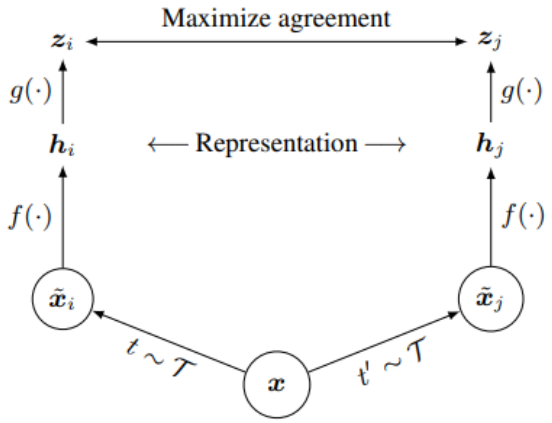

Figure 3.3: Overview of SimCLR. From [9].

Next, the projection head $g(.)$ is a multi-layer perceptron with one hidden layer using ReLU (rectified linear unit) as an activation function, which maps the representation vectors into the space where the contrastive loss is applied: $z_i = g(h_i) = W^{(2)}\sigma\left(W^{(1)}h_i\right)$ where σ is ReLU. They found valuable to define the contrastive loss on the output of the projection head, instead of the representation vectors from the base encoder. Lastly, given a set $\{\tilde{x}_k\}$ including a positive pair of samples $\tilde{x}_i$ and $\tilde{x}_j$. The contrastive loss function is defined for a contrastive prediction task, which its goal is to identify $\tilde{x}_j$ in $\{\tilde{x}_k\}_k \neq i$ for a given $\tilde{x}_i$. Therefore, the loss function for a positive pair of images (i, j) can be described as:

$$\ell_{i,j} = -\log \frac{\exp\left(\operatorname{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp\left(\operatorname{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_k\right)/\tau\right)} \tag{3.7}$$

where $\mathbb{1}_{[k\neq i]} \in \{0, 1\}$ is an indicator function evaluating to 1 if $k \neq i$, τ denotes a temperature parameter and $sim(u, v) = u\top v/\left\|u\right\|\left\|v\right\|$ denoting the cosine similarity.

3.3 MoCo: *Mo*mentum *Co*ntrast for Constrative Learning

Moreover, another state-of-the-art framework is Momentum Contrast (MoCo) [24]. MoCo is a self-supervised learning framework which can be used to build large and consistent dictionaries. It uses a contrastive loss to train a visual representation encoder to perform a dictionary look-up by matching an encoded query to a dictionary of encoded keys, which an encoded query should correspond to its matching key and distinct to others. While in [9], they used only one encoder network, this framework employed two encoders: an encoder network and an encoder momentum. For the dictionary look-up task, MoCo considers an encoded query q and a set of encoded samples $\{k_0, k_1, k_2, ...\}$ which are the keys of a dictionary. Also, considering that there is a single key k_+ in the dictionary that matches q. The contrastive loss is low when q is similar to its positive key k_+ and distinct to all other keys, which are considered negative keys k_- for q. The similarity is measured by the contrastive loss function, which in this case they used InfoNCE, that is described as:

$$\mathcal{L}_q = -\log \frac{\exp\left(q.k_+/\tau\right)}{\sum_{i=0}^{K} \exp\left(q.k_i/\tau\right)} \tag{3.8}$$

where t is a temperature hyper-parameter. Since there is one positive key k_+ and K negative samples, the loss is the log loss of a (K + 1)-way softmax-based classifier that tries to classify q as k_+ [24]. Also, to train the encoder networks that represent the queries and keys, the contrastive loss acts as an unsupervised objective function. The query representation is described as $q = f_q(x_q)$ where f_q is an encoder network and x_q is a query sample. In a similar way, the key representation is described as $k = f_k(x_k)$ where f_k is an encoder network and x_k is a key sample.

The core of this framework is based on keeping the dictionary as a queue of data samples, which helps to decouple the dictionary size from the mini-batch size, allowing to build larger dictionaries, and since its more flexible it can be set as a hyper-parameter. The samples in the dictionary are gradually replaced. The current mini-batch is enqueued to the dictionary while the

oldest mini-batch in the queue is removed, which can be beneficial since the oldest mini-batch encoded keys are the most outdated being the least consistent with the newest encoded keys.

Although, using a queue is important to make the dictionary larger, it also carries the issue of making it unmanageable to update the key encoder by back-propagation. To address this issue, the authors proposed momentum update, which can be described as:

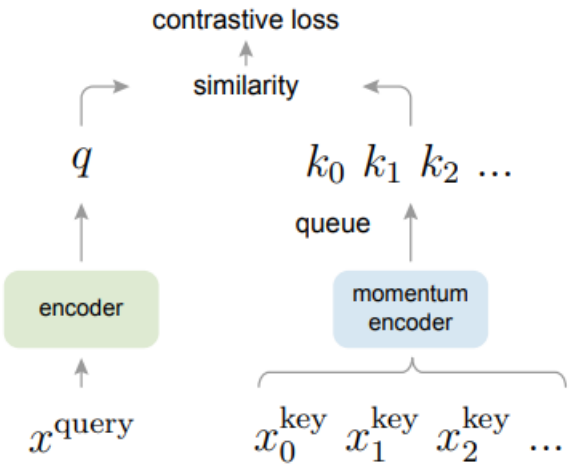

Figure 3.4: Overview of MoCo. Momentum Contrast (MoCo) [24] employs a contrastive loss to match an encoded query q to a dictionary of encoded keys to train a visual representation encoder. Using a set of data samples, the dictionary keys $\{k_0, k_1, k_2, ...\}$ are immediately defined. In order to decouple the dictionary from the mini-batch size, it is constructed as a queue, with the newest mini-batch being dequeued and the current mini-batch being enqueued. An encoder slowly encodes the keys while is powered by a momentum update with the query encoder.

$$f_k \leftarrow m f_k + (1 - m) f_q \tag{3.9}$$

where $m \in [0, 1]$ is a momentum coefficient, and only the f_q are updated by back-propagation. The momentum update makes f_k progress more smoothly than f_q.

Meanwhile, this framework follows a pretext task, called instance discrimination task. They consider a query and a key as a positive pair if both originate from the same image, and otherwise is considered as a negative pair. An overview of MoCo [24] is shown in figure 3.4.

3.4 MoCo v2

MoCo v2 [12] is a follow-up paper from MoCo, where they make some improvements by implementing two design improvements used in the SimCLR framework [9]. First, they replace the fully connected head in MoCo with a 2-layer MLP head (hidden layer 2048-d with ReLU). This only affects the unsupervised training stage (e.g., pre-traning stage). Additionally, they extend the original augmentation by adding the blur augmentation. Interestingly, by adding an extra augmentation alone without the MLP head, it improves the MoCo baseline on ImageNet [14] by 2.8%.

3.5 DINO: self *di*stillation with *no* labels

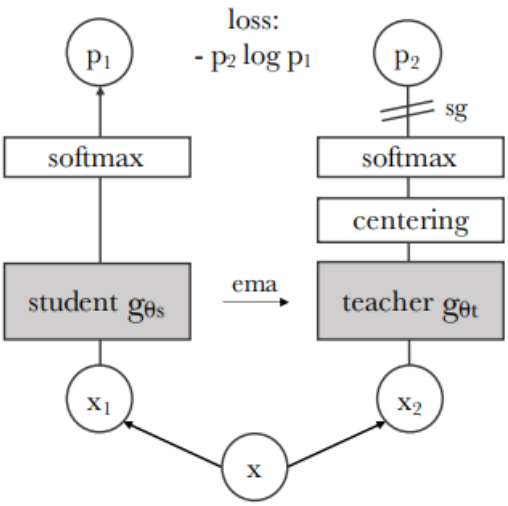

Figure 3.5: Overview of DINO[5]. The student and teacher networks receive two distinct random transformations of an input image from the model. The two networks share the same architecture, but their parameters differ. A mean calculated across the batch centers the teacher network's output. Both networks generate a K-dimensional feature over the feature dimension, which are normalized using a temperature softmax. A cross-entropy loss is then used to determine how similar they are. The gradients are only propagated through the student, therefore we apply a stop-gradient (sg) operator on the teacher. An exponential moving average (ema) of the student parameters is used to update the teacher parameters.

Furthermore, there's a new self-supervised system designed by Facebook AI called DINO (self **di**stillation with **no** labels) [31].

DINO employs self-distillation which is a form of knowledge distillation whereas a student network $g_{\theta s}$ is trained to match the output of a given teacher network $g_{\theta t}$, which is parameterized by θ_s and θ_t respectively. Both networks produce probability distributions over K dimensions, indicated by P_s and P_t, given an input image x. The probability P is determined by applying a softmax function to normalize the network g output.

$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^{K} \exp(g_{\theta_s}(x)^{(k)}/\tau_s)} \tag{3.10}$$

with $\tau_s > 0$, a temperature parameter is employed to control the sharpness of the output distribution. A similar formula is applied to P_t with the temperature parameter τ_t. We learn to match these distributions, given a fixed teacher network $g_{\theta t}$, by minimizing the cross-entropy loss regarding the student network's parameters θ_s:

$$\min_{\theta_s} H(P_t(x), P_s(x)), \tag{3.11}$$

where $H(a, b) = -alogb$.

Following the latter equation, to adapt to self-supervised learning, they used a multi-crop strategy [6], by generating a set of different views of a given image. The set contains several local

views of smaller resolution and two global views x_1^g and x_2^g. Next, to encourage "local-to-global" correspondences, all the crops are passed through the student while only the global views are passed through the teacher. Therefore, they minimize the loss as:

$$\min_{\theta_s} \sum_{x \in \{x_1^g, x_2^g\}} \sum_{\substack{x^{'} \in V \\ x^{'} \neq x}} H(P_t(x), P_s(x^{'})). \tag{3.12}$$

This loss is general and can be applied to any number of views. The authors employed multiple local views of resolution 96^2 that cover only small areas (such as less than 50%) of the original image and two global views of resolution 224^2 that cover a large (like more than 50%) portion of the original image, in accordance with the standard multi-crop configuration. While the sets of parameters θ_s and θ_t differ, both networks have the same architecture g, a Vision Transformer (ViT) [15]. The parameters θ_s are learned by minimizing the latter equation with the help of stochastic gradient descent.

Regarding the teacher network $g_{\theta t}$, unlike the classic approach in knowledge distillation where we have a $g_{\theta t}$ $apriori$, they build it from past iterations of the student network. The authors used an exponential moving average (EMA) on the student weights, i.e., a momentum encoder [24]. The update rule is as follows:

$$\theta_t \leftarrow \lambda \theta_t + (1 - \lambda)\theta_s \tag{3.13}$$

where λ follows a cosine schedule from 0.996 to 1 during training. The teacher demonstrated superior performance throughout the training process, therefore guiding the student by giving higher-quality target features.

Moreover, to avoid model collapse DINO employed a centering and sharpening of the momentum teacher output. The centering has an effect of preventing one dimension to dominate while encouraging collapse to the uniform distribution. On the other hand, sharpening has the opposite effect. Both operations balance their effects, and so they are enough to prevent collapse when a momentum teacher is present. Additionally, selecting this method exchanges stability for less dependence over the batch: the centering operation can be thought of as adding a term c to the teacher and only needs first-order batch statistics. Regarding the term c, its updated with an exponential moving average allowing this method to work effectively across varying batch sizes:

$$c \leftarrow mc + (1 - m)\frac{1}{B}\sum_{i=1}^{B} g\theta_t(x_i), \tag{3.14}$$

where $m > 0$ is a rate parameter and B is the batch size. Also, by using a low value for the temperature τ_t in the teacher softmax normalization yields output sharpening. The DINO overview is illustrated in figure 3.5.

3.6 DINOv2

Later on, they improved DINO with its successor DINOv2, where they combine key elements from DINO [5] and iBOT [49]. Firstly, they introduced an automatic pipeline to build a curated image

dataset, instead of using uncurated data, as usually done in the self-supervised learning paradigm. First, they removed near-duplicated images to decrease redundancy and enhance diversity among images. Next, they retrieved images from their uncurated data source that are similar to images from their curated data source, using a self-supervised ViT-H/16 network pretained on ImageNet-22k to compute the image embeddings, and a cosine-similarity as a distance measure between images. Finally, they perform a k-means clustering of the uncurated data. An overview of these data processing pipeline is shown in figure 3.6.

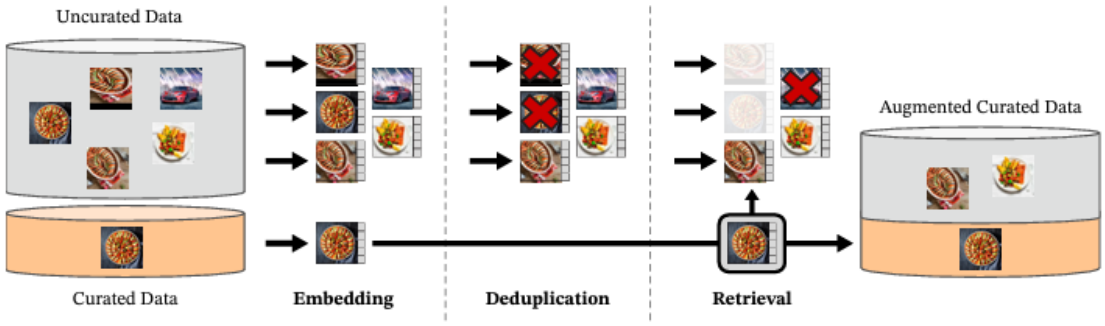

Figure 3.6: Overview of the data processing pipeline. From [37].

Next, unlike the knowledge distillation approach used in DINO, they adjusted the approach by not using a pre-existing teacher network, but a self-supervised learning method in which the teacher network is built on from previous iterations of the student network by using an exponential moving average (i.e., EMA). Also, some patches of the input are masked for the student network, then the cross-entropy is calculated between the features of the two patches. Additionally, they replace the teacher softmax-centering of DINO [5] and iBOT [49] by the Sinkhorn-Knopp batch normalization of SwAV [6], whilst they apply the softmax normalization for the student network. In addition, they use a KoLeo regularizer, which derives from the Kozachenko-Leonenko differential entropy estimator to promote a uniform span of the features within a batch, yet before computing this regulizer, they use a l_2 to normalize the features. Finally, they adapt the resolution since small objects can disappear at low resolution. However, by training at high resolution, it demands more memory and time, therefore they increase the resolution of images to 518 x 518 during a short period at the end of pre-training.

3.7 SwAV: *Sw*apping *A*ssignments between multiple *V*iews of the same image

SwAV (**Sw**apping **A**ssignments between multiple **V**iews of the same image) is another state-of-the-art self-supervised learning approach that surpassed state-of-the-art methods such as SimCLR [9] and MoCo [24]. SwAV innovated in two ways: (1) they proposed a scalable clustering loss which works in both large or small batches without the need of a large memory bank nor a momentum encoder [24]; (2) they introduced a new data augmentation strategy called multi-crop that

uses a mix of views, instead of two standard resolution crops with no additional computational or memory cost during training, resulting in improvements when using this strategy on other self-supervised methods [9, 24].

Unlike other clustering-based methods [1, 4] which are offline, the authors proposed an online clustering-based self-supervised learning method. Usually, clustering-based methods alternate between two steps: cluster assignment and training. Cluster assignment is where image features of the entire dataset are clustered, while the training step revolves around predicting cluster assignments (i.e., "codes") for different image views. However, these methods are not fit for online learning since they need multiple passes over the dataset to compute the image features required for further clustering.

SwAV is inspired by contrastive instance learning [46] in the sense that instead of considering the codes as a target, it imposes a consistent mapping between views of the same image. Yet, instead of comparing image features, it compares cluster assignments (or "codes"). More accurately, a code is computed from an augmented version of an image, which then we predict that code with other augmented versions of the same image. In Fig 3.7, its illustrated the relation between contrastive learning and the SwAV framework.

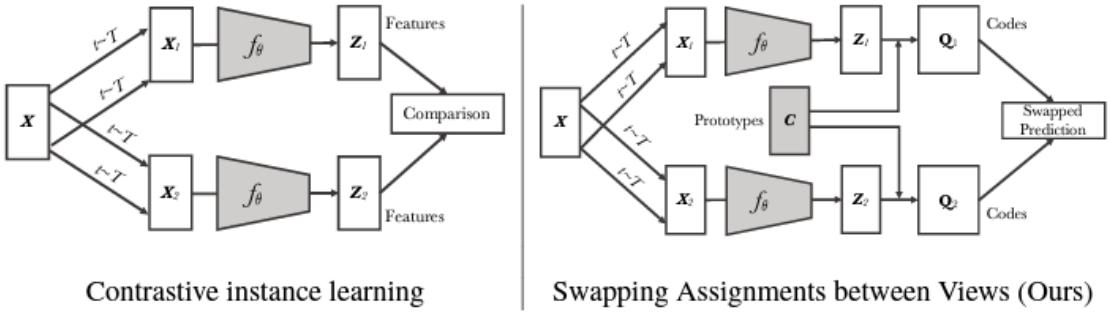

Figure 3.7: **Contrastive instance learning (left) *vs.* SwAV [6] (right).** In contrastive learning methods that are applied to instance classification, the features from different views of the same input are compared directly to each other. In SwAV, the "codes" are obtain by assigning features to a set of prototype vectors, then it comes the "swapped" prediction problem where the codes obtained from one augmented view are used to predict another view, which leads to not directly compare image features.

Given two image features z_t and z_s from two different augmentations of the same image, we match those to a set of K prototypes $\{c_1, c_2, ..., c_K\}$ yielding their codes q_t and q_s, followed by a "swapped" prediction with the following loss function:

$$L(z_t, z_s) = l(z_t, q_s) + l(z_s, q_t), \tag{3.15}$$

where the function $l(z, q)$ computes the fit between features z and a code q. If both features share the same semantics, it should be possible to predict the code from the other feature.

We start by applying a transformation t from the set T of image transformations to each image x_n resulting into an augmented view x_{nt}, followed by a mapping to a vector representation by

applying a non-linear mapping f_θ to x_{nt}, which then the image feature can be projected to the unit sphere (i.e., $z_{nt} = f_\theta(x_{nt})/||f_\theta(x_{nt})||_2$). Now, we can compute a code q_{nt} from z_{nt} by mapping it to a set a of K trainable prototype vectors $\{c_1, c_2, ..., c_K\}$. We denote by C the matrix whose columns are the prototype vectors.

The loss function in 3.15 has two terms that setup the "swapped" prediction problem. Each one represents the cross entropy loss between the code and the probability computed by the softmax of the dot products of z_i and all trainable prototypes in C:

$$\ell(\mathbf{z}_t, \mathbf{q}_s) = -\sum_k q_s^{(k)} \log p_t^{(k)}, \quad \text{where} \quad p_t^{(k)} = \frac{\exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_k\right)}{\sum_{k'} \exp\left(\frac{1}{\tau}\mathbf{z}_t^\top \mathbf{c}_{k'}\right)} \tag{3.16}$$

where τ is a temperature parameter [46].

In order to make this approach online, we need to compute the codes using only the images features within a batch making all the examples equally partitioned by the prototypes. By applying this equipartition, it ensures that the code for different images in a batch are distinct, therefore preventing the trivial solution that every image has the same code.

Given B feature vectors $Z = [z_1, z_2, ..., z_B]$ and the prototypes $C = \{c_1, c_2, ..., c_K\}$, we map those to yield codes denoted by $Q = [q_1, q_2, ..., q_B]$, while we optimize Q so it maximizes the similarity between the features and the prototypes vectors:

$$\max_{\mathbf{Q}\in\mathcal{Q}} \text{Tr}\left(\mathbf{Q}^\top \mathbf{C}^\top \mathbf{Z}\right) + \varepsilon H(\mathbf{Q}), \tag{3.17}$$

where H is the entropy function $H(\mathbf{Q}) = -\sum_{ij} \mathbf{Q}_{ij} \log \mathbf{Q}_{ij}$ and ε is a parameter that controls the smoothness of the mapping.

The authors observed that instead of using an high ε (i.e., strong entropy regularization) which causes all samples to collapse into an unique representation and being all assigned uniformly to all prototypes, they keep ε low.

Multi-crop strategy

The first step is applying transformations to the input which yields augmentation views of the input. This step is viewed as an important role for capturing information in terms of relations between objects or parts of a scene [9, 35]. However, increasing the number of crops quadratically increases the computational resources and memory requirements [6]. For that reason, the authors proposed a strategy called multi-crop, where they use two standard resolution crops and more V additional low resolution crops that cover only small parts of the image, which ensures only a small increase in computational cost [6]. Yet, only the full resolution crops are used to compute the codes. More specifically, the authors generalize the loss of Eq 3.15:

$$L(z_{t_1}, z_{t_2}, \ldots, z_{t_{V+2}}) = \sum_{i\in\{1,2\}} \sum_{v=1}^{V+2} \mathbb{1}_{v\neq i} \ell(z_{t_v}, q_{t_i}). \tag{3.18}$$

Additionally, the authors tested this new augmentation strategy on other self-supervised methods and it showed a better performance on several SSL methods including SimCLR [9].

Chapter 4

Experiments

In this chapter, we will present the experimental setup. It will cover the datasets used, which methods were trained from scratch and which relied on pre-trained models for feature extraction. Additionally, it will cover the DSMIL training procedure.

4.1 Datasets

Initially, we decided to use two datasets: Camelyon16 and TCGA Lung Cancer.

The Camelyon16 dataset [17] is a publicly available dataset that was collected in Radboud University Medical Center (Nijmegen, the Netherlands), and the University Medical Center Utrecht (Utrecht, the Netherlands) for the Camelyon16 challenge, hosted by International Symposium on Biomedical Imaging (ISBI), containing a total of 400 WSIs. The dataset are hematoxylin and eosin (H&E) stained whole-slide images of lymph node sections for detection and classification of metastases in breast cancer. The training dataset consists of 270 WSIs of lymph node (160 normal and 110 containing metastases). The test dataset consists of 130 WSIs.

The TCGA (The Cancer Genome Atlas) Lung Cancer dataset is a publicly available dataset launched by The National Cancer Institute (NCI) and the National Human Genome Research Institute (NHGRI) in 2006. The WSIs provide two sub-types of lung cancer, namely Lung Adenocarcinoma (LUAD) and Lung Squamous Cell Carcinoma (LUSC), with a total of 1054 diagnostic digital slides. The data was split into 840 training slides and 210 testing slides, with 4 low-quality corrupted slides being discarded [31].

However, due to the lack of computational resources, we couldn't train on Camelyon16 and TCGA Lung Cancer datasets due to the fact that after cropping into patches, the size of the dataset increased to millions of patches, and with these size it would take too much time and resources that we couldn't afford. Therefore, we decided to use a smaller dataset but still providing a challenge enough to benchmark our model. The dataset we choose is Patch Camelyon (PCam) which its size positions is between CIFAR-10 and ImageNet, providing a substantial challenge while remaining manageable with the availability of our resources.

The Patch Camelyon (PCam) [44] dataset is an image classification dataset which consists of 327.680 color images with a resolution of 96 x 96 pixels. PCam is derived from the Camelyon16

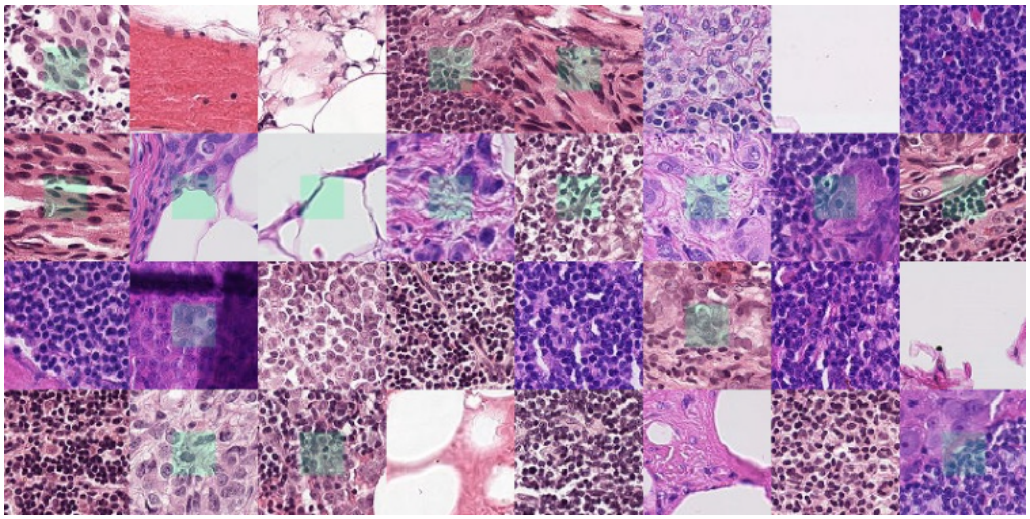

Figure 4.1: Samples from the PCam dataset. Green boxes indicate the presence of tumor tissue in the center region, which establishes a positive label.

Challenge [17], therefore these images are extracted from histopathologic scans of lymph node sections. Each image is annotated with a binary label indicating presence or absence of metastatic tissue.

The dataset is divided into a training set of 262,144 images, a validation set of 32,768 images, and a test set of 32,768 images. The training, validation, and test sets are balanced with a 50/50 split between positive and negative samples, and there is no overlap in whole-slide images (WSIs) across these splits. Although, we did not use the test set. The training script used had already a scheme that reserves 20% of all data, while the remaining 80% samples are used to construct a 5-fold cross-validation, meaning that we don't need to divide the dataset since the training script already does that for us. The 5 best models along with the corresponding optimal thresholds are used to perform inference on the reserved test set, whilst the final prediction for a test sample is the majority vote of the 5 models. Yet, we need a specific folder structure of our dataset to be used to train the model. For that, we need to create a folder for each bag (containing all patches belonging to the same bag), while each folder remains on a specific directory depending on the class it belongs to (i.e., normal or tumor). Except that, we don't know where the samples of the test set are extracted from. For that reason, we discarded the test set. In short, the dataset used consists of 294,912 images. Nevertheless, this meticulous curation enhances the reliability of model evaluations and mitigates the risk of overfitting.

4.2 Learning the feature extractor

As mentioned earlier, most self-supervised learning methods are pre-trained on ImageNet [14], a dataset that consists of over 14 million images across 21,000 categories. However, these categories ranges from natural images, objects and scenes, which the domain shift between WSI and ImageNet data are completely different, and thus may lead to sub-optimal representations. The

goal of training the feature extractor from scratch either with our own data or with a dataset that has similar characteristics to our target task is so it can produce better embeddings (i.e., representations) when we compute the embeddings on the target dataset (i.e., mapping each instance to an embedding). For that reason, instead of using a fixed ImageNet pre-trained feature extractor, we train from scratch our own feature extractor using SSL methods. However, from the SSL methods we described earlier, only the SimCLR [9] and MoCo [24] methods were trained from scratch using the PCam dataset. The rest of the SSL methods used on this work were already pre-trained.

Since the DSMIL architecture [31] used SimCLR [9] as the feature extractor, we decided to start our search to find the best feature extractor with SimCLR. This method is known for using large batch sizes but we didn't have the computational resources (i.e., graphic card memory) enough to deal with such batch sizes, so we changed the batch size from 4096 to 512 and the input shape to match the images size from the PCam dataset from (224, 224, 3) to (96, 96, 3) since each image has a resolution of 96 x 96 pixels. Also, the CNN backbone used for SimCLR was a ResNet-18 [23]. We began to train our feature extractor for 100 epochs using an Adam optimizer with a learning rate of 1 x 10^{-5} and a weight decay of 10^{-6}. In terms of dataset used, we tried with 100% of the PCam dataset but we soon realized that it would take too much time, therefore we reduced the data to 50% (i.e., 147,456 instances). Even with halve the data, it took approximately 15 minutes per epoch, thus in total 25 hours to train SimCLR [9] from scratch on PCam.

The next SSL method used was MoCo [24]. We used the first version of MoCo [24]. After implementing and adjusting the MoCo algorithm, we realized that it would take too much time to train on 100% of data. Therefore, we reduced the PCam dataset to 5% which is equivalent to 14,745 samples. Even so, it took approximately 12 minutes per epoch, totaling roughly 20 hours to fully train. We trained for 100 epochs with a learning rate of 1.5×10^{-1}, a weight decay of 10^{-4}, a batch size of 128, and the CNN backbone used was a ResNet-50 [23].

Table 4.1: Summary of SSL Methods and Training Configurations

Method	Dataset Size	Batch Size	Learning Rate	Weight Decay	Training Time	CNN Backbone
SimCLR	147,456	512	1×10^{-5}	1×10^{-6}	25 hours	ResNet-18
MoCo	14,745	128	1.5×10^{-1}	1×10^{-4}	20 hours	ResNet-50

4.3 Pre-trained SSL methods

The rest of the SSL methods used on this work were pre-trained, instead of training them from scratch.

There are a lot of advantages by using pre-train models, yet the main advantage we were interested on was the reduced training time and the computational resources needed. For example, the next SSL method used was MoCo v2 [12], the second version of MoCo [24]. We used the pre-train version trained for 800 epochs, which if we were to train MoCo v2 from scratch for

800 epochs, it would take weeks with the current computational resources. The downside is that the model is not pre-trained on histopathology domain. Instead, the authors used ImageNet-1M with approximately 1.28 million images in 1000 classes and Instagram-1B with nearly 940 million images. They didn't specify which one they used to pre-train the model but it doesn't matter since neither one is close to the histopathology domain.

Next SSL model used was DINOv2 [37]. We used a pre-trained DINOv2 model from hugging face. Since the model was already pre-trained, and given that DINOv2 is a Vision Transformer [15], when we computed the features on the dataset, we experimented with different configurations such as the feature size and the number of attention heads.

Finally, the last SSL model that we tried is SwAV [6]. For the SwAV model, we simply downloaded the best pre-trained model with a ResNet-50 for a CNN backbone.

4.4 DSMIL Training Protocol

As it was explained earlier in section 3.1, before we train our main model, DSMIL, we need to compute the features on the dataset. For that, we need to use our pre-trained feature extractor to perform inference on the whole dataset. Next, we choose the size of the dataset since that depending on the SSL method we used to learn the feature extractor, it takes more or less time to train.

The hyperparameters that we adjusted were the learning rate, weight decay, and when necessary dropout, specifically patch dropout rate, and the dimension of the feature size referring to the output dimension when we performed the feature extractor to compute the embeddings.

The evaluation scheme we used to evaluate the model was a 5-fold cross-validation with a standalone test. This means that we performed a 5-fold cross validation with 80% of data, whilst the remaining 20% was reserved for the test set.

In terms of metrics, we computed the accuracy, balanced accuracy and the AUC. We used the TensorBoard with PyTorch to track and visualize these metrics along our training. We tracked each fold, thus in later figures showing the models' performance, it will show the performance of each fold within the same figure.

Regarding the training, we trained our model for 200 epochs with earlier stoppage if training has not improved after 25 epochs.

For the size of the dataset, we used different ratios depending on the SSL method used. The reason for that is the time it took for each SSL method. SimCLR [9] and MoCo [24] were trained from scratch so we couldn't use all data, otherwise it would take too much time. Regarding the other SSL models [12, 37, 6], we used a pre-trained version of them, therefore we could use bigger portions (e.g., 100% of the dataset). However, DINOv2 took too much time, hence we only used 25% of the dataset. For example, when we computed the embeddings on the dataset with DINOv2, it took 70 minutes, followed by an additional 35 minutes for training. In total, if we had use 100% of data, it would take 420 minutes (i.e., 7 hours). The reason it takes more time than the others is its architecture, which is based on the Vision Transformer [15] that while it offers state-of-the-

Table 4.2: Dataset size for each SSL method used

SSL Method	**Dataset Size (%)**	**Number of instances**
SimCLR	50%	147,456
MoCo	25%	73,728
MoCov2	25%	73,728
	100%	294,912
DINOv2	25%	73,728
SwAV	25%	73,728
	50%	147,456
	100%	294,912

art performance, it requires a larger computation due to their complex attention mechanisms and larger parameters, hence it is more time-consuming. In table 4.2, its illustrated the portion of the dataset used to train DSMIL with each SSL method.

Chapter 5

Results and Discussion

In this chapter, we will present and discuss the results obtained from the different SSL methods used to learn the feature extractor. This includes the performance of SimCLR, MoCo, MoCo v2, DINOv2, and SwAV frameworks on the PCam dataset.

5.1 SimCLR Results

As we can notice in table 5.1, overall the results were pretty good with most of them achieving an accuracy and a balanced accuracy over 90%. The best performance model achieved an accuracy of 96.3% with a balanced accuracy of 96.2% showing that SimCLR [9] is a promising SSL method for learning the feature extractor. However, this SSL method already showed its potential with the architecture we are building upon [31], achieving an accuracy of 92.4% with the same split and evaluation scheme, yet a different dataset (i.e., Camelyon16) even though the PCam dataset was extracted from the Camelyon16. It seems that SimCLR is a good starting point to explore different SSL methods to learn the feature extractor.

Table 5.1: Classification performance from the main model (i.e., DSMIL [31]) using SimCLR [9] to learn the feature extractor. SimCLR was already used on the original architecture but we tried to replicate the results on a different dataset (i.e., PCam dataset). It achieved great results with the best hyperparameter configuration being a learning rate of 1×10^{-4}, a weight decay of 1×10^{-3} and no dropout patch, which achieved an accuracy of 96.3% with a balanced accuracy of 96.2%.

Learning Rate	Weight Decay	Dropout Patch	Accuracy	Balanced Accuracy
1×10^{-3}	1×10^{-3}	0.0	0.9444	0.9243
1×10^{-4}	1×10^{-3}	0.0	**0.9630**	**0.9616**
1×10^{-2}	1×10^{-3}	0.0	0.9259	0.9429
1×10^{-3}	1×10^{-2}	0.0	0.9074	0.9156
1×10^{-4}	1×10^{-2}	0.0	0.9259	0.9245
1×10^{-3}	1×10^{-2}	0.2	0.9444	0.9559
1×10^{-3}	1×10^{-2}	0.5	0.8889	0.8889
2.5×10^{-4}	1×10^{-2}	0.0	0.8148	0.8248
1×10^{-2}	1×10^{-4}	0.0	**0.9630**	0.9583
1×10^{-3}	1×10^{-4}	0.0	**0.9630**	0.9524
1×10^{-4}	1×10^{-4}	0.0	0.8889	0.8706
1×10^{-3}	1×10^{-4}	0.3	0.8889	0.9004

Regarding the best performance model, we show in figure 5.1 and 5.2 the training loss and the validation loss progress over the course of the training. In figure 5.1, we can notice that the training loss smoothly decreases suggesting that the model are successfully minimizing the loss overtime without any problems. In figure 5.2, the model exhibits similar performance, decreasing steadily overtime indicating that the model is learning, specially on unseen data without any indications that is overfitting or underfitting. The blue line which indicates one of the folds is showing a fluctuation, yet minimal and it still follows the downward trend.

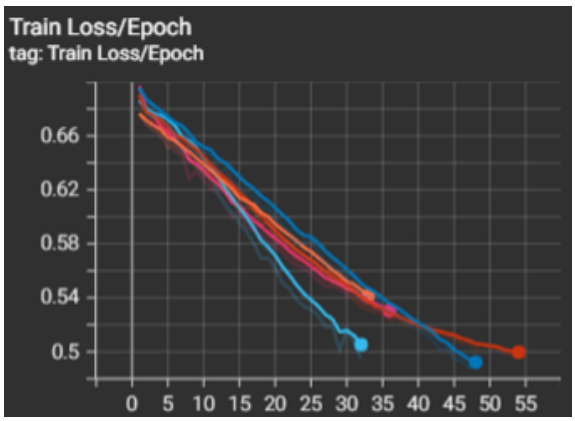

Figure 5.1: Training Loss per Epoch using SimCLR as the feature extractor. SimCLR was trained from scratch using the same dataset as of DSMIL (i.e., PCam dataset). All folds steadily decreases over a span of 55 epochs without any perturbations.

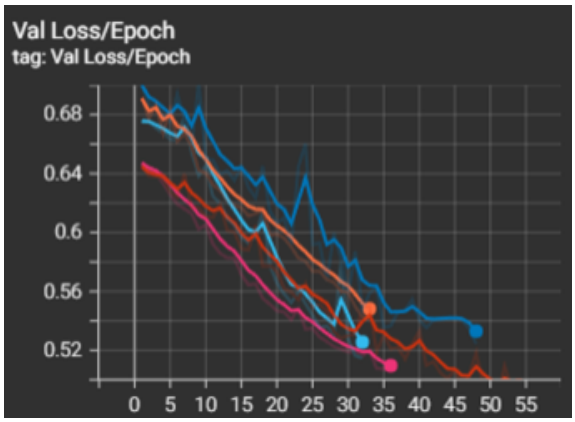

Figure 5.2: Validation Loss per Epoch using SimCLR as the feature extractor. All folds steadily decreases over a span of 55 epochs with the blue fold having minor fluctuation.

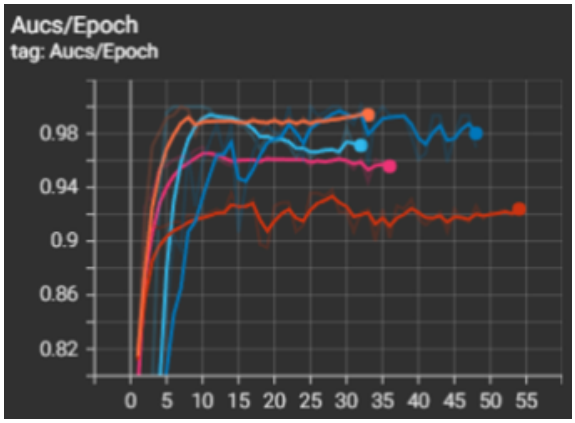

Figure 5.3: Area Under the Curve for each epoch using SimCLR as the feature extractor. All folds show a strong performance stabilizing around 15 epoch.

Moving on to the figure 5.3, which shows the AUC (i.e., area under the curve) for each epoch. It shows a strong performance within the first 10 epochs suggesting that the model learns quickly and around by epoch 15 to 20 the model starts to stabilize with minor fluctuations.

5.2 MoCo Results

As the table 5.2 demonstrates, the results were mixed and not as good as expected from the MoCo framework [24]. We trained for 200 epochs with earlier stoppage if the model hasn't improved after 25 epochs. The results goes as low as an accuracy of 44.4%, and the best performance model achieved an accuracy of 79.6% with a balanced accuracy of 79.5%. Since we only used 5% of the dataset (14,745 instances) to train it from scratch, it could be insufficient data to train the model suggesting that it may have led to underfitting. We could have used a bigger portion of the dataset, yet with limited time and computational resources, whilst given that there is pre-trained MoCo models available, we decided to try MoCo v2 pre-trained for 800 epochs, which we will demonstrate later on.

Table 5.2: Classification performance from DSMIL [31] using MoCo [24] to learn the feature extractor. It achieved poor results with the best hyperparameter configuration being a learning rate of 1×10^{-2}, a weight decay of 1×10^{-4} and a dropout patch of 0.5, which achieved an accuracy of 79.6% with a balanced accuracy of 79.5%.

Learning Rate	Weight Decay	Dropout Patch	Accuracy	Balanced Accuracy
2.5×10^{-4}	1×10^{-2}	0.0	0.6296	0.6104
1×10^{-3}	1×10^{-2}	0.0	0.6296	0.6591
1×10^{-4}	1×10^{-5}	0.0	0.4444	0.4625
1×10^{-2}	1×10^{-4}	0.0	0.7963	0.7783
1×10^{-2}	1×10^{-3}	0.0	0.7407	0.7283
1×10^{-2}	1×10^{-4}	0.5	0.7037	0.7195
1×10^{-2}	1×10^{-2}	0.2	0.6852	0.6529
1×10^{-1}	1×10^{-2}	0.2	0.7222	0.7381
1×10^{-1}	1×10^{-2}	0.0	0.7348	0.7275
1×10^{-2}	1×10^{-4}	0.5	**0.7963**	**0.7953**

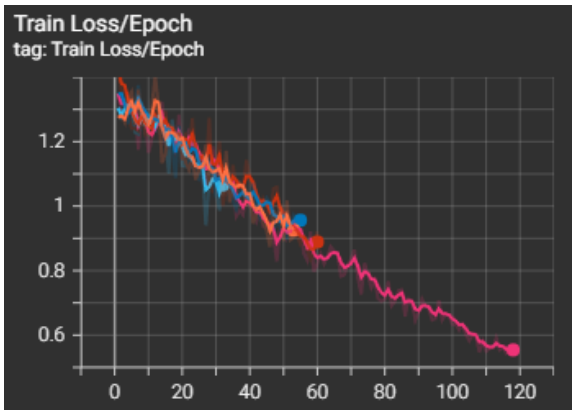

Figure 5.4: Training Loss per Epoch using MoCo as the feature extractor. MoCo was trained from scratch using the same dataset as of DSMIL (i.e., PCam dataset). Most folds steadily decreases over a span of 60 epochs without any fluctuations, while pink fold decreases over a span of twice the epochs (i.e., 120).

In figures 5.4 and 5.5, we show the best performing model training and validation losses respectively. The training loss illustrates a steady decrease overtime exhibiting minor fluctuations indicating that the model is learning. However, regarding the validation loss, it follows a downward trend but with major fluctuations suggesting that the model is unstable when performing on unseen data. It is a sign of overfitting, yet the results weren't the best and one certain cause of this behavior is the lack of data.

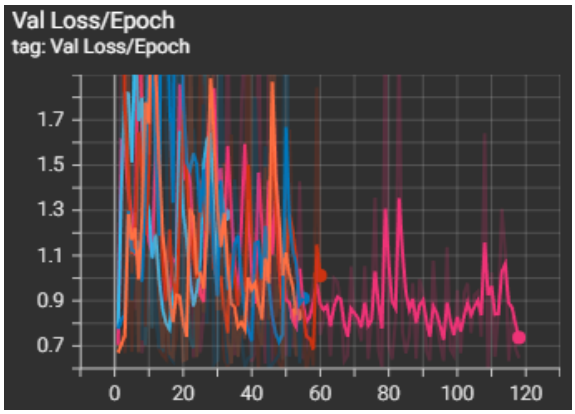

Figure 5.5: Validation Loss per Epoch using MoCo as the feature extractor. In this plot, all folds exhibit a more violent performance with major fluctuations, while still following a downward trend, which suggests that the model became unstable performing on unseen data.

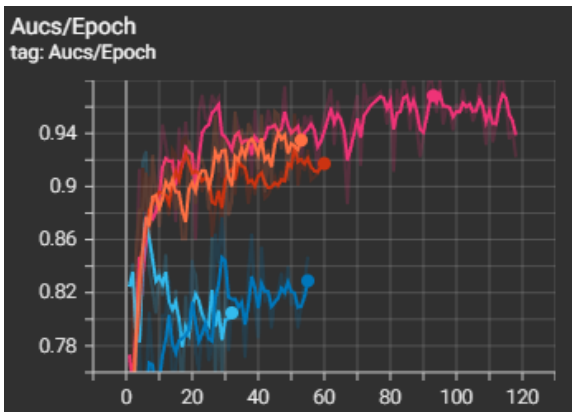

Figure 5.6: Area Under the Curve for each epoch using MoCo as the feature extractor. Most folds rapidly increase within the first 20 epochs, which then start to stabilize with minor fluctuations. Some with a better performance than others. Overall a good performance, even though the validation loss showed instability by the model.

Following to the figure 5.6, which demonstrates the AUC per epoch. Overall, it rapidly increases in the first 20 epochs, which by epoch 40 it starts to stabilize with small fluctuations. Both blue folds showed a lower performance compared to the other. Although, overall the model showed a good performance regardless of the instability shown on the validation loss plot.

5.3 MoCo v2 Results

In table 5.3, its illustrated the results using the pre-trained MoCo v2 [12] as the feature extractor. We experimented with 25% of data and 100% of data. Interestingly the results came out better with a smaller dataset. Still, we got results low as an accuracy of 40.7% and the highest with 87%. The best performing model got an accuracy of 87% with a balanced accuracy of 87.4%. We hoped for a better performance since it was pre-trained model for 800 epochs, yet it was pre-trained on

ImageNet [14] and as I mentioned before, the gap from those images and histopathology images is significant which can lead to poor performance.

Table 5.3: Classification performance from DSMIL [31] using a pre-trained MoCo v2 model [12] as the feature extractor on two different dataset sizes (e.g., 25% and 100%). It achieved mixed results with the best hyperparameter configuration being a learning rate of 1×10^{-2}, a weight decay of 1×10^{-4} and no dropout patch, which achieved an accuracy of 87% with a balanced accuracy of 87.4% on 25% of data (i.e., 73,728 images).

Dataset Size	Learning Rate	Weight Decay	Dropout Patch	Accuracy	Balanced Accuracy
25% (73,728 instances)	1×10^{-2}	1×10^{-2}	0.0	0.8333	0.8113
	1×10^{-3}	1×10^{-2}	0.0	0.7778	0.7835
	1×10^{-2}	1×10^{-3}	0.0	0.6852	0.6678
	1×10^{-3}	1×10^{-3}	0.2	0.7222	0.7167
	1×10^{-1}	1×10^{-4}	0.0	0.8148	0.8250
	1×10^{-4}	1×10^{-4}	0.0	0.4074	0.3985
	1×10^{-1}	1×10^{-3}	0.0	0.7963	0.7966
	1×10^{-1}	1×10^{-5}	0.3	0.7593	0.8143
	1×10^{-1}	1×10^{-2}	0.0	0.8333	0.8231
	1×10^{-2}	1×10^{-4}	0.0	**0.8704**	**0.8738**
100% (294,912 instances)	1×10^{-2}	1×10^{-2}	0.0	0.6852	0.6852
	1×10^{-2}	1×10^{-3}	0.0	0.7222	0.7973
	1×10^{-1}	1×10^{-3}	0.0	0.6481	0.6676
	1×10^{-3}	1×10^{-3}	0.0	0.6852	0.7090
	1×10^{-2}	1×10^{-4}	0.3	0.7037	0.7353
	1×10^{-2}	1×10^{-5}	0.0	0.7778	0.7618
	1×10^{-4}	1×10^{-1}	0.0	0.7222	0.7468

We demonstrate the training loss in figure 5.7 of our best performance model, which shows a steady and consistently decrease along all the training process. All the folds (represent by the different colors) are all well align suggesting that the model is learning across the various subsets of data.

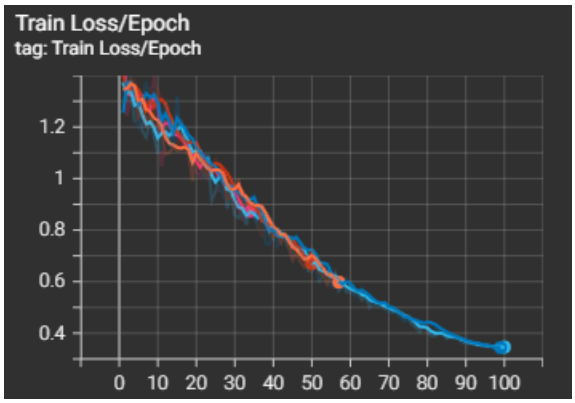

Figure 5.7: Training Loss per Epoch using a pre-trained MoCo v2 model as the feature extractor. Most folds steadily decreases over a span of around 60 epochs without any fluctuations, while both blue folds decreases over a span of 100 epochs.

In figure 5.8, its illustrated the validation loss, where once more is unstable with significant spikes and fluctuations, yet all folds show a downward trend decreasing gradually.

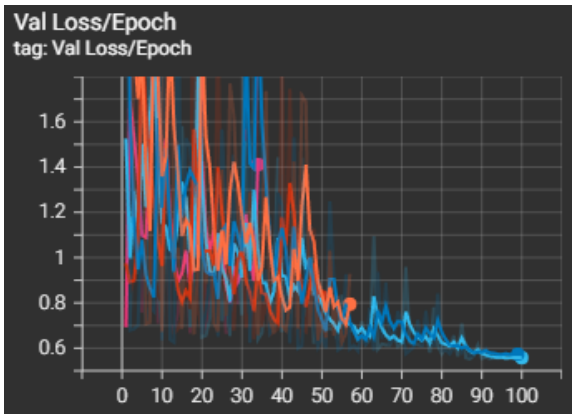

Figure 5.8: Validation Loss per Epoch using a pre-trained MoCo v2 model as the feature extractor. In this figure, all folds exhibit a more unstable performance with major fluctuations, while still following a downward trend, which suggests that the model became unstable performing on unseen data. A similar performance was shown using MoCo as the feature extractor.

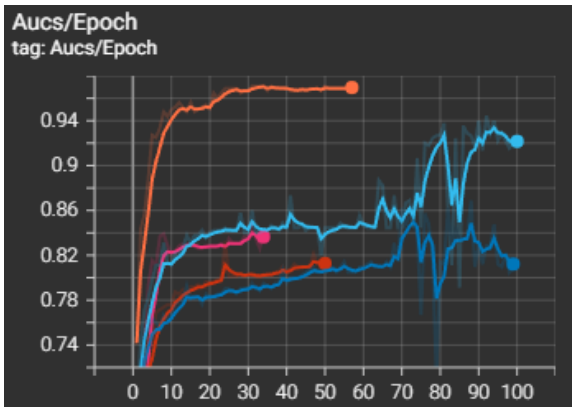

Figure 5.9: Area Under the Curve for each epoch using a pre-trained MoCo v2 model as the feature extractor. This plot showed a mixed performance with the orange fold exhibiting a strong performance with rapidly increase over 10 epochs, while the others showed a similar performance. Both blue folds got more unstable around epoch 75.

In figure 5.9, its shown the AUC along the training process. Once more most folds had an initial spike, and by epoch 20 it starts to stabilize. The orange fold showed the strongest performance with a rapidly increase over 10 epochs with then starts to stabilize. The folds illustrated with a lighter and a darker blue color starts to get more unstable around epoch 75 but with small fluctuations.

5.4 DINOv2 Results

In table 5.4, its represented the results for using DINOv2 [37] as the feature extractor. We vary two other hyperparameters, feature size and the number of attention heads. We got a mix of

results ranging from 59.2% to 92.6%. Several models got the top accuracy of 92.6%, but the best performing model got also a balanced accuracy of 93.1% which is not too far from the other models with the same accuracy.

Table 5.4: Classification performance from DSMIL [31] using a pre-trained DINOv2 model [37] as the feature extractor. It achieved mixed results with the best hyperparameter configuration being a learning rate of 1×10^{-2}, a weight decay of 1×10^{-2}, a feature size of 768, and 16 attention heads, which achieved an accuracy of 92.6% with a balanced accuracy of 93.1%.

Learning Rate	Weight Decay	Feature Size	Attention Heads	Accuracy	Balanced Accuracy
1×10^{-2}	1×10^{-2}	768	12	**0.9259**	0.9259
1×10^{-3}	1×10^{-2}	768	12	0.8704	0.8409
1×10^{-2}	1×10^{-3}	768	12	0.8889	0.8915
1×10^{-2}	1×10^{-4}	768	12	**0.9259**	0.9208
1×10^{-3}	1×10^{-3}	768	12	**0.9259**	0.9221
1×10^{-3}	1×10^{-4}	768	12	0.8148	0.7912
1×10^{-1}	1×10^{-3}	768	12	0.7963	0.7713
1×10^{-2}	1×10^{-2}	576	12	0.8519	0.8528
1×10^{-3}	1×10^{-2}	576	12	0.7963	0.7662
1×10^{-1}	1×10^{-3}	576	12	**0.9259**	0.8947
1×10^{-1}	1×10^{-4}	576	12	0.7778	0.7742
1×10^{-2}	1×10^{-2}	384	12	0.9074	0.8935
1×10^{-3}	1×10^{-2}	384	12	0.8704	0.8456
1×10^{-1}	1×10^{-2}	384	12	0.7407	0.7222
1×10^{-2}	1×10^{-3}	384	12	0.8148	0.8055
1×10^{-2}	1×10^{-2}	1024	16	0.6481	0.6169
1×10^{-3}	1×10^{-2}	1024	16	0.8333	0.8297
1×10^{-1}	1×10^{-2}	1024	16	0.7037	0.6910
1×10^{-2}	1×10^{-2}	768	16	**0.9259**	**0.9307**
1×10^{-3}	1×10^{-2}	768	16	0.7963	0.7222
1×10^{-1}	1×10^{-2}	768	16	0.8519	0.8182
1×10^{-2}	1×10^{-2}	768	24	0.7593	0.7230
1×10^{-3}	1×10^{-2}	768	24	0.8519	0.7895
1×10^{-1}	1×10^{-2}	768	24	0.5926	0.5500

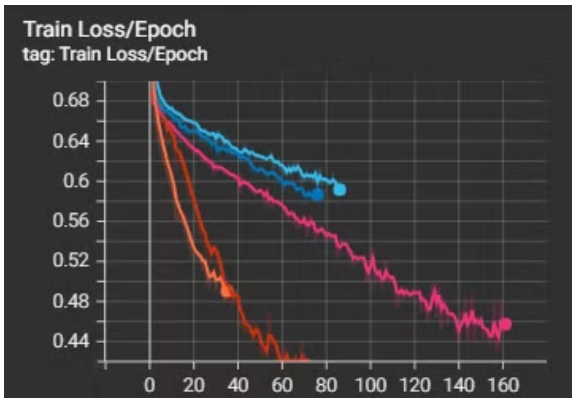

Figure 5.10: Training Loss per Epoch using a pre-trained DINOv2 model as the feature extractor. All folds exhibit a steady decrease overtime. Most folds stop at around epoch 80, whilst pink fold stops at twice the epochs compared to the others folds (i.e., 160).

In figures 5.10 and 5.11, its illustrated the training loss and the validation loss respectively. In 5.10, all folds exhibit a steady decrease suggesting a consistent performance across all different training subsets.

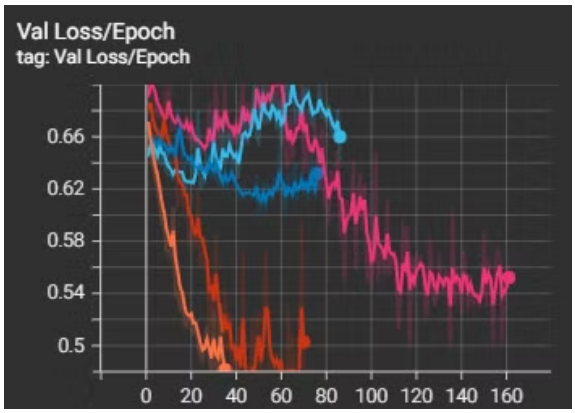

Figure 5.11: Validation Loss per Epoch using a pre-trained DINOv2 model as the feature extractor. This plot shows a more strange behavior compared to the training loss plot. Most folds exhibit a downward trend with minor fluctuations, except the light blue fold which ended up with a validation loss superior to its initial validation loss value. Although, overall the model shows a good performance.

On the other hand, figure 5.11 demonstrates a more erratic behavior compared to the training loss curves. There are folds with a lot of fluctuations, even the light blue fold got a validation loss superior to its initial value, yet the difference is minimal. Still, most folds follow a downward trend. These differences between the folds can be due to the variability in the data distribution across folds.

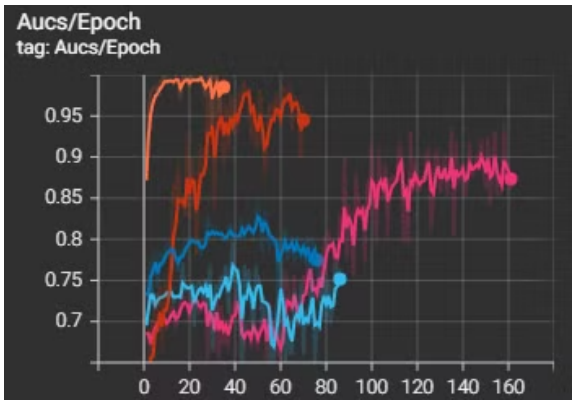

Figure 5.12: Area Under the Curve for each epoch using a pre-trained DINOv2 model as the feature extractor. Red and orange folds showed a strong performance with an initial spike, whilst pink fold needed more time to show its peak performance. Both blue folds didn't increase as much as the others folds. Overall a good performance by the model.

In figure 5.12, its illustrated the AUC per epoch. It shows an initial spike with some folds (e.g., light and darker blue colors) showing a more stabilize curve, while the other folds converge towards higher AUC values with minor fluctuations.

5.5 SwAV Results

In table 5.5, its illustrated the results of experimenting with SwAV [6] as the feature extractor. The results were by far the best, with the lowest accuracy of 87% and the highest of 98.2%. SwAV had a fast performance, hence we trained with all data. While the performance of the models were close to each other, the best performance model got an accuracy of 98.2% with a balanced accuracy of 97.9%.

Table 5.5: Classification performance from DSMIL [31] using a pre-trained SwAV model [6] as the feature extractor on three different dataset sizes (e.g., 25%, 50% and 100%). It achieved great results with the best hyperparameter configuration being a learning rate of 1×10^{-3}, a weight decay of 1×10^{-3} and a dropout patch of 0.2, which achieved an accuracy of 98.2% with a balanced accuracy of 97.9% on 100% of data (i.e., 294,912 images).

Dataset Size	Learning Rate	Weight Decay	Dropout Patch	Accuracy	Balanced Accuracy
25% (73,728 instances)	1×10^{-2}	1×10^{-3}	0.0	0.9444	0.94
	1×10^{-3}	1×10^{-3}	0.0	**0.9815**	0.975
	1×10^{-1}	1×10^{-3}	0.0	0.9444	0.9331
	1×10^{-3}	1×10^{-2}	0.0	**0.9815**	0.975
	1×10^{-4}	1×10^{-2}	0.0	0.9444	0.9375
	1×10^{-3}	1×10^{-4}	0.0	0.9444	0.9428
	1×10^{-4}	1×10^{-5}	0.0	0.9259	0.9028
	2×10^{-4}	1×10^{-1}	0.0	0.9630	0.9594
50% (147,456 instances)	1×10^{-2}	1×10^{-2}	0.0	0.9074	0.9038
	1×10^{-3}	1×10^{-2}	0.0	0.9074	0.9125
	1×10^{-3}	1×10^{-3}	0.0	0.9444	0.9455
	1×10^{-1}	1×10^{-3}	0.0	0.9630	0.9600
	1×10^{-4}	1×10^{-3}	0.0	0.9630	0.9610
	1×10^{-4}	1×10^{-1}	0.0	0.9074	0.8896
	1×10^{-4}	1×10^{-4}	0.0	0.9444	0.9460
	1×10^{-3}	1×10^{-4}	0.0	**0.9815**	0.9722
100% (294,912 instances)	1×10^{-3}	1×10^{-2}	0.3	0.9630	0.9565
	1×10^{-3}	1×10^{-2}	0.5	0.9259	0.9444
	2×10^{-4}	1×10^{-4}	0.5	0.8704	0.8056
	2×10^{-4}	1×10^{-4}	0.2	0.9444	0.9318
	1×10^{-3}	1×10^{-3}	0.2	**0.9815**	**0.9792**
	1×10^{-3}	1×10^{-3}	0.5	0.9630	0.9677
	1×10^{-2}	1×10^{-3}	0.2	0.9630	0.9603

In figures 5.13 and 5.14, its presented the training and validation loss respectively. All folds show a steady decrease in 5.13. All folds converge in early epochs with a consistent downward trend suggesting that the model is learning effectively across the different training subsets.

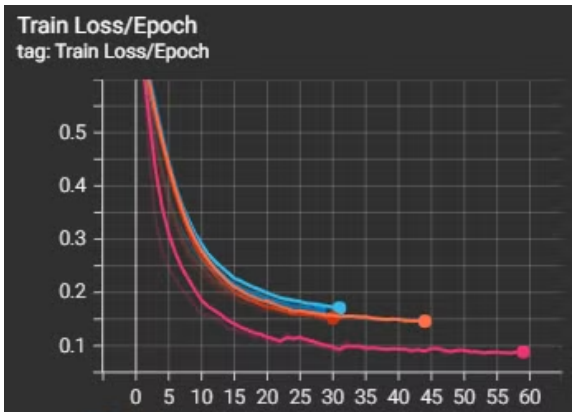

Figure 5.13: Training Loss per Epoch using a pre-trained SwAV model as the feature extractor. All folds exhibit a steady decrease over a span of 60 epochs.

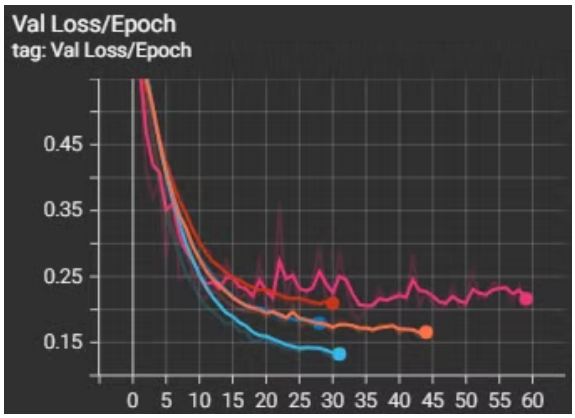

Figure 5.14: Validation Loss per Epoch using a pre-trained SwAV model as the feature extractor. All folds exhibit a similar behavior as the training loss plot. The pink folds shows minor fluctuations. Overall a great performance suggesting that the model is learning without any perturbations.

Regarding the validation loss in 5.14, all folds show a general downward trend with a smooth decrease, except one fold (i.e., pink color) that has minor fluctuations. Nevertheless, it could be some variability in that particular validation subset.

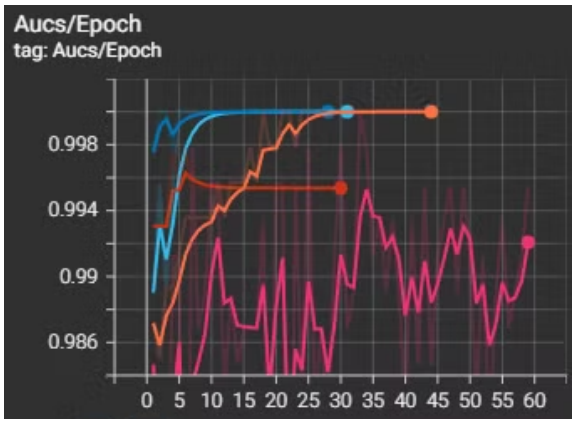

Figure 5.15: Area Under the Curve for each epoch using a pre-trained SwAV model as the feature extractor. All folds show a great performance with very high AUC values.

In figure 5.15, its presented the AUC per epoch while using SwAV. All folds show a strong performance with an initial spike, while some converge in the initial epochs (e.g., light blue, darker blue, and red lines). The orange fold starts to converge around epoch 25, and finally pink fold fluctuates along the training, yet is truly minimal since the y-axis ranges from 0.98 to 1.0.

Chapter 6

Conclusion and Future Work

In this thesis, we study the application of deep learning methodologies, particularly focusing on multiple instance learning (MIL) and self-supervised learning (SSL), to address challenges in histopathology image analysis, more specifically in whole slide imaging (WSI). The goal was to mitigate the reliance on large and annotated datasets, since it is a major bottleneck when developing computer-aided diagnosis (CAD) systems for medical imaging. Therefore, we choose a multiple instance learning framework that can handle whole-slide images, which can reach enormous dimensions (e.g., resolutions up to 100,000 x 100,000 pixels).

The core of this work involved enhancing the feature extractor in the chosen MIL architecture using SSL methods to enable the model to learn useful representations without the need of large curated datasets. Before training the MIL model, we need to compute the embeddings on the dataset using the feature extractor, which yields better embeddings, the better feature extractor used. The issue is that many SSL methods are pre-trained on huge datasets that consists of images of objects, scenery, and more in which the gap between WSI and those images can lead to sub-optimal embeddings. To improve the feature extractor, we need to close the gap by pre-training on datasets composed of histopathology images. Therefore, we trained from scratch SSL methods on the target dataset that we are going to use further on to train the MIL model. Not all of the SSL methods used were trained from scratch due to computational bottlenecks.

Even though we didn't use the original datasets from the architecture we built upon, the dataset used was extracted from the original datasets. From the two SSL models trained from scratch, MoCo got poor results with an accuracy of 79.6%, but one main reason was the lack of data given that we couldn't afford to train with more data, so it still is a good option. SimCLR, on the other hand, achieved great results with an astonish accuracy of 96.3%, showing one more time that it is a great SSL technique, even though there is more options that achieve state-of-the-art results.

Moreover, the rest of the SSL techniques were pre-trained models on datasets such as ImageNet-1M, ImageNet-22k, Instagram-1B, and others. This is a downside as we explained earlier. However, we got the best results with one of these pre-trained SSL models. MoCo v2 got a mixed results with a top accuracy of 87%, it could have been better if we payed more attention to it and adjusted more the model. Moving on to DINOv2, where it got greats results with a top accuracy of 92.6%, yet I hoped for a better performance since the Vision Transformer (ViT) based architecture

has achieved greats results in computer vision. Once again, the issue was lack of data considering that we only used 25% of all data, yet as a ViT-based model it requires an extensive computational resources that we couldn't afford to deal with. Finally, SwAV was the best performing model with the best results by far. SwAV got an astonish accuracy of 98.1% surpassing SimCLR by 1.8%. Despite there isn't a large gap betwen SwAV and SimCLR results, SwAV got consistent outcomes with $> 90\%$ accuracy in most cases, showing that is a promising SSL technique. Additionally, SwAV wasn't pre-trained on a histopathology dataset, which was the case of SimCLR.

In conclusion, this study has proven that both MIL and SSL are promising techniques for improving the pipeline of histopathology image analysis. By decoupling the performance of deep learning models from the dependence on large curated data, we can reduce costs, save time and prioritize other issues without the need of expert-driven data annotation.

6.1 Future Work

While this work has demonstrated the effectiveness of combining multiple instance learning (MIL) and self-supervised learning (SSL) methods, remains opportunities for future research on this topic. Future research should include pre-trained SSL methods fully on histopathology datasets, whilst explore other options within the SSL paradigm. Note that we focused on contrastive learning but there are other options that should be revised. On the other hand, MIL should continue to be explored within the medical imaging field since once more proved to be a promising technique in the way that handles whole-slide images.

Bibliography

[1] Yuki M. Asano, C. Rupprecht, and Andrea Vedaldi. "Self-labelling via simultaneous clustering and representation learning". In: *ArXiv* abs/1911.05371 (2019). URL: https://api.semanticscholar.org/CorpusID:207930156.

[2] Gabriele Campanella et al. "Clinical-grade computational pathology using weakly supervised deep learning on whole slide images". en. In: *Nat. Med.* 25.8 (Aug. 2019), pp. 1301–1309.

[3] Marc-André Carbonneau et al. "Multiple instance learning: A survey of problem characteristics and applications". In: *Pattern Recognition* 77 (2018), pp. 329–353. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2017.10.009. URL: https://www.sciencedirect.com/science/article/pii/S0031320317304065.

[4] Mathilde Caron et al. "Deep Clustering for Unsupervised Learning of Visual Features". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[5] Mathilde Caron et al. "Emerging Properties in Self-Supervised Vision Transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 9650–9660.

[6] Mathilde Caron et al. "Unsupervised learning of visual features by contrasting cluster assignments". In: (June 2020). arXiv: 2006.09882 [cs.CV].

[7] Mathilde Caron et al. "Unsupervised Pre-Training of Image Features on Non-Curated Data". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[8] Richard J Chen et al. "A general-purpose self-supervised model for computational pathology". In: (Aug. 2023). arXiv: 2308.15474 [cs.CV].

[9] Ting Chen et al. "A simple framework for contrastive learning of visual representations". In: (Feb. 2020). arXiv: 2002.05709 [cs.LG].

[10] Ting Chen et al. "Big self-supervised models are strong semi-supervised learners". In: (June 2020). arXiv: 2006.10029 [cs.LG].

[11] Xi Chen et al. "Recent advances and clinical applications of deep learning in medical image analysis". In: *Medical Image Analysis* 79 (2022), p. 102444. DOI: 10.1016/j.media.2022.102444. URL: https://doi.org/10.1016/j.media.2022.102444.

[12] Xinlei Chen et al. "Improved baselines with momentum contrastive learning". In: (Mar. 2020). arXiv: 2003.04297 [cs.CV].

[13] Ozan Ciga, Tony Xu, and Anne Louise Martel. "Self supervised contrastive learning for digital histopathology". In: *Machine Learning with Applications* 7 (2022), p. 100198. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2021.100198. URL: https://www.sciencedirect.com/science/article/pii/S2666827021000992.

[14] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[15] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: (Oct. 2020). arXiv: 2010.11929 [cs.CV].

[16] Alexey Dosovitskiy et al. *Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks*. 2015. arXiv: 1406.6909 [cs.LG]. URL: https://arxiv.org/abs/1406.6909.

[17] Babak Ehteshami Bejnordi et al. “Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer”. In: *JAMA* 318.22 (2017), pp. 2199–2210. DOI: 10.1001/jama.2017.14585. URL: https://doi.org/10.1001/jama.2017.14585.

[18] Navid Farahani, Anil Parwani, and Liron Pantanowitz. “Whole slide imaging in pathology: advantages, limitations, and emerging perspectives”. In: *Pathology and Laboratory Medicine International* 7 (2015), pp. 23–33. DOI: 10.2147/PLMI.S59826. URL: https://doi.org/10.2147/PLMI.S59826.

[19] Jean-Bastien Grill et al. “Bootstrap your own latent: A new approach to self-supervised Learning”. In: (June 2020). arXiv: 2006.07733 [cs.LG].

[20] Metin N Gurcan et al. “Histopathological image analysis: a review”. en. In: *IEEE Rev. Biomed. Eng.* 2 (Oct. 2009), pp. 147–171.

[21] Metin N. Gurcan et al. “Histopathological Image Analysis: A Review”. In: *IEEE Reviews in Biomedical Engineering* 2 (2009), pp. 147–171. DOI: 10.1109/RBME.2009.2034865.

[22] Simms L. Gurina TS. *Histology, Staining*. https://www.ncbi.nlm.nih.gov/books/NBK557663/. 2023.

[23] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[24] Kaiming He et al. “Momentum Contrast for Unsupervised Visual Representation Learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[25] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[26] Shih-Cheng Huang, Animesh Pareek, Matt Jensen, et al. “Self-supervised learning for medical image classification: a systematic review and implementation guidelines”. In: *npj Digital Medicine* 6 (2023), p. 74. DOI: 10.1038/s41746-023-00811-0. URL: https://doi.org/10.1038/s41746-023-00811-0.

[27] Maximilian Ilse, Jakub M Tomczak, and Max Welling. “Attention-based deep multiple instance learning”. In: (Feb. 2018). arXiv: 1802.04712 [cs.LG].

[28] Mahendra Khened, Avinash Kori, Hitesh Rajkumar, et al. “A generalized deep learning framework for whole-slide image segmentation and analysis”. In: *Scientific Reports* 11 (2021), p. 11579. DOI: 10.1038/s41598-021-90444-8. URL: https://doi.org/10.1038/s41598-021-90444-8.

[29] Mahendra Khened et al. “A generalized deep learning framework for whole-slide image segmentation and analysis”. en. In: *Sci. Rep.* 11.1 (June 2021), p. 11579.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Commun. ACM* 60.6 (2017), 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

[31] Bin Li, Yin Li, and Kevin W. Eliceiri. "Dual-Stream Multiple Instance Learning Network for Whole Slide Image Classification With Self-Supervised Contrastive Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 14318–14328.

[32] Xiang Li et al. "Selective Kernel Networks". In: (Mar. 2019). arXiv: 1903.06586 [cs.CV].

[33] Xintong Li et al. "A comprehensive review of computer-aided whole-slide image analysis: from datasets to feature extraction, segmentation, classification and detection approaches". In: *Artif. Intell. Rev.* 55.6 (Aug. 2022), 4809–4878. ISSN: 0269-2821. DOI: 10.1007/s10462-021-10121-0. URL: https://doi.org/10.1007/s10462-021-10121-0.

[34] Ming Y Lu et al. "Data efficient and weakly supervised computational pathology on whole slide images". In: (Apr. 2020). arXiv: 2004.09666 [eess.IV].

[35] Ishan Misra and Laurens van der Maaten. "Self-Supervised Learning of Pretext-Invariant Representations". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 6706–6716. URL: https://api.semanticscholar.org/CorpusID:208617491.

[36] *NEUROCLINOMICS2: LaSIGE project Page*. https://www.lasige.di.fc.ul.pt/Projects/NEUROCLINOMICS2. [Online - Accessed on 7-12-2019]. 2014.

[37] Maxime Oquab et al. "DINOv2: Learning robust visual features without supervision". In: (Apr. 2023). arXiv: 2304.07193 [cs.CV].

[38] Maithra Raghu et al. *Do Vision Transformers See Like Convolutional Neural Networks?* 2022. arXiv: 2108.08810 [cs.CV]. URL: https://arxiv.org/abs/2108.08810.

[39] Yangjun Ruan et al. "Weighted ensemble self-supervised learning". In: (Nov. 2022). arXiv: 2211.09981 [cs.LG].

[40] Zhuchen Shao et al. "TransMIL: Transformer based correlated multiple instance learning for whole slide image classification". In: (June 2021). arXiv: 2106.00908 [cs.CV].

[41] Andrew H Song et al. "Artificial intelligence for digital and computational pathology". en. In: *Nat Rev Bioeng* 1.12 (Oct. 2023), pp. 930–949.

[42] Hamid Reza Tizhoosh and Liron Pantanowitz. "Artificial Intelligence and Digital Pathology: Challenges and Opportunities". In: *Journal of Pathology Informatics* 9.1 (2018), p. 38. ISSN: 2153-3539. DOI: https://doi.org/10.4103/jpi.jpi_53_18. URL: https://www.sciencedirect.com/science/article/pii/S2153353922003510.

[43] Ashish Vaswani et al. "Attention is all you need". In: (June 2017). arXiv: 1706.03762 [cs.CL].

[44] Bastiaan S Veeling et al. "Rotation Equivariant CNNs for Digital Pathology". In: (June 2018). arXiv: 1806.03962 [cs.CV].

[45] Xiyue Wang et al. "Transformer-based unsupervised contrastive learning for histopathological image classification". In: *Medical Image Analysis* 81 (2022), p. 102559. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2022.102559. URL: https://www.sciencedirect.com/science/article/pii/S1361841522002043.

[46] Zhirong Wu et al. “Unsupervised Feature Learning via Non-parametric Instance Discrimination”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3733–3742. DOI: 10.1109/CVPR.2018.00393.

[47] Junyuan Xie, Ross Girshick, and Ali Farhadi. *Unsupervised Deep Embedding for Clustering Analysis*. 2016. arXiv: 1511.06335 [cs.LG]. URL: https://arxiv.org/abs/1511.06335.

[48] Hongrun Zhang et al. “DTFD-MIL: Double-Tier Feature Distillation Multiple Instance Learning for Histopathology Whole Slide Image Classification”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 18802–18812.

[49] Jinghao Zhou et al. “iBOT: Image BERT Pre-Training with Online Tokenizer”. In: (Nov. 2021). arXiv: 2111.07832 [cs.CV].

[50] Zhi-Hua Zhou. “A brief introduction to weakly supervised learning”. In: *National Science Review* 5.1 (Aug. 2017), pp. 44–53. ISSN: 2095-5138. DOI: 10.1093/nsr/nwx106. eprint: https://academic.oup.com/nsr/article-pdf/5/1/44/31567770/nwx106.pdf. URL: https://doi.org/10.1093/nsr/nwx106.