POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Benchmarking Self Supervised Representation Learning methods on Biological Datasets

Supervisors

Prof. Elisa FICARRA

Prof. Paolo GARZA

Prof. Francesco PONZIO

Candidate

Nicola OCCELLI

April 2022

Summary

Deep Learning, and especially Deep Supervised Learning has been the driving force of computer vision in the last decade. Although this approach is the most promising when applied to big dataset of annotated images, as the labelled data decrease in quantity, also its performance decreases in quality. This is ever so true when applied to highly specialized domains, such as Biological Imaging. For this very reason, scientists have been developing the so called self-supervised paradigm. This paradigm allows to learn a model even with a small number of annotated data, without compromising the quality or generalization power of the learned model. In this work we explore different ways of performing self-supervised learning, exploiting the new Pytorch-based framework: PytorchLightning. We run multiple experiments on different biomedical datasets on NVidia-powered GPU computing nodes. Our main goal is to benchmark different self-supervised learning approaches, and find out which one works best in the biomedical domain. As a side-product of this work, we produce a reusable and scalable code base, which will speed up research by removing completely the boilerplate, that allows for fast prototyping and experimenting.

Acknowledgements

I would like to say thanks to my family on top of everyone else, for always believing in me, even when I am not believing in myself. I would like to thanks equally my girlfirend, for always bearing with me and never stop caring about me. Special thanks to my supervisors for helping me find the inspiration for my thesis. Lastly, I would like to extend a thank you to every person that have been a positive influence in my journey to this day.

Table of Contents

List of Tables

List of Figures

Acronyms

AE
Auto-Encoder

AI
artificial intelligence

ANN
Artificial Neural Network

CE
Cross-Entropy

CNN
Convolutional Neural Network

CRC
Colorectal Cancer

CT
Computerized Tomography

DAE
Denoising Auto-Encoder

DL
Deep Learning

FCN
Fully Connected Network

GAN

Generative Adversarial Network

GB

GigaByte

Gb

GigaBit

GLCM

Gray-Level Co-occurrence Matrix

GPU

Graphics Processing Unit

HE

Hematoxylin and Eosin

LR

Learning Rate

MESO

Mesothelioma

MI

Mutual Information

MSE

Mean Squared Error

NN

Neural Network

PRP

Predict Relative Position

RAM

Random Access Memory

ROI

Region of Interest

SL

Supervised Learning

SSL

Self Supervised Learning

SSRL

Self Supervised Representation Learning

SVM

Support Vector Machine

TPU

Tensors Processing Unit

UL

Unsupervised Learning

VAE

Variational Auto-Encoder

VRAM

Video Random Access Memory

WSI

Whole Slide Image

Chapter 1

Introduction

1.1 Technological Motivation

Ever since the advent of LeNet5[1] in 1989, neural networks have been the main approach to solve some of the most important computer vision problems. The neural network paradigm then developed into what we call today Deep Learning (to which we will be referring to as DL hereon). DL is a technique consisting in stacking multiple neural networks layer on top of one-another, the more layers we stack, the deeper the architecture. DL models are able to extract a hierarchy of representations of an image, starting from intuitive high-level representation obtained in the first superficial levels, and ending with complex low-level representation at the deepest levels. One particular kind of DL models are Convolutional Neural Networks (to which we will be referring to as CNN hereon). CNNs are inspired by the processing done inside the visual cortex of animal's brain (including human's). Today we employ CNNs to solve tasks like (but not limited to) Semantic Segmentation, Image Classification[2] and Object Detection. Together with the development of these models, the reason why we are able to run these learning algorithms successfully is the concurrent high-speed development of the hardware components needed to run them in the first place [3]. Some of these hardware components are Graphical Processing Units (GPUs for short), and Tensor Processing Units (TPUs for short). These are computing units, typically equipped with 10 to 20 Gb of high-speed dedicated RAM memory, and specialised processors, able to perform trillions of floating point operations per second. The most successful paradigm studied by researchers in the last decade is Supervised Learning, and it consists in learning DL models from large annotated datasets, (i.e. ImageNet [4]). Although, not every imaging domain can afford a large annotated dataset. This means that the Supervised Learning paradigm can fail to learn an effective model for that task, because of the lack of data. For this reason, another paradigm emerged,

called Self Supervised Learning (SSL hereon). SSL consists in learning meaningful representation by training the model on unlabelled data. After this first training, the model can be fine-tuned to perform a downstream task (i.e. image classification), or can be used as is, as a feature extractor, to train a separate downstream model on said features (i.e. a Support Vector Machine classifier).

1.2 Moral Motivation

In the field of biology and medicine it is common to produce images using microscopes, Computerized Tomography (CT) scans and many other instruments. These images are typically digitalized and stored in large clinical datasets. Although the amount of data is consistent, annotating a microscopy image is costly, and requires skills that are usually only possessed by a few employees at the hospitals. This means that the amount of annotated data is far smaller than the raw data itself, and most of the times is not suitable to train a supervised model. Moreover, fine-tuning and domain adaptation techniques are difficult to apply, since the biological and medical domain are far apart from that of natural images (which are the images on which most of the state-of-the-art models are trained). For this reason, SSL fits perfectly into this framework, and we think that it represents one of the main research avenues for the years to come.

1.3 Chapters content

The work we are going to present is of two different natures. One component is a research of the literature surrounding Self-Supervised learning in the current Deep Learning landscape. This will include a general overview of the approach, compared to the other state-of-the-art approaches currently in use in the research community. The second, and most substantial part of is the coding and experimenting of some of what we thought are the most promising techniques of SSL. For this last part, we wrote a Python experimenting library/framework based on the recently released PytorchLightning framework. One of our goals since the beginning was to focus on writing reusable code, removing as much boiler plate as possible. During the development, we consulted and included multiple existing solutions taken from the code repositories of researchers, and while they should be on the top of the priority list, readability and reproducibility are often disregarded. For this reasons, we decided to adopt a framework that makes these two issues its main goals. What we obtained is a Python library, available at our GitHub , easily scalable, aimed at removing the effort of coding from the experimenting phase.

1.4 Clinical Objectives

From the clinical point of view, this work's objective is the automated diagnosis of cancer (or cancer subtypes) to a patient, given a (or multiple) microscopy images of tissue samples.

The reason for automation has it's basis in the two major drawbacks that can be found in the literature.

- *Time*: Manual annotation is time consuming, especially for large datasets, and even more for large images.

- *Subjectivity*: Manual annotations are subjective. The annotation of different images is affected by variability. The effect of variability is twofold: it affects inter-observer (i.e. the annotations of the same image between two different observers are affected by variability) and intra-observer (i.e. different image's annotation by the same observer are still affected by variability)

For the reason just explained the research community is focusing its efforts towards the automated diagnosis of tumors. The automated diagnosis can be subdivided in two main branches:

- *Automated Segmentation*: This branch focuses on the automatic partition of heterogeneous WSI samples of a given tissue into homogeneous tiles of the same kind of tissue.

- *Automated Classification*: This is the branch that this work will focus on. Researchers working in this direction aim at classifying regions of tissue into either benign-malign class system, or any other class system that best defines the task at hand. This classification is based on quantitative features. In our work, the qualitative features will be extracted by means of a Neural Network model.

The progress towards automated classification has been notable in the last few years. Many different anatomical regions such as colon, brain, lungs, breast and prostate have benefited from these new techniques. The most way of tackling this problem however have been of the feature selection type: a limited local-descriptor set has been extracted from the input images, such as Gray-Level Co-occurrence Matrix or (GLCM), wavelet transforms and many others. These hand-crafted features have then been fed to a machine learning classification algorithm such as Support Vector Machine classifiers, Logistic Regression classifier or Random Forest classifiers. The shortcoming of these methods has not been the classificator itself, but instead the very features they have been trained on. The negative

effects of hand picking features is twofold. First it requires the researcher a non-trivial knowledge of which of the virtually infinite features are best suited for the classification task. Second, the generalization power of the algorithm is severely affected by such choice.

The natural reaction to the aforementioned limitations is the adoption and specialization of Deep Learning architectures to the biomedical domain. CNNs are the architectures that provide the greatest performances when applied to the discussed problems.

1.5 Technical Objectives

Our main focus in this work is to compare the performances of supervised learning with self-supervised learning. We are driven by the lack and high cost of annotations in the biomedical field. Our objective is then to have a benchmark of multiple self-supervised approaches, applied to multiple datasets, with different characteristics. On top of that, we want to validate the results simulating different amounts of labelled data for our algorithms, this way, we hope to find ad highlight the point where supervised learning stops being optimal, and self-supervised learning shows its strength in working with a fraction of labels with respect to the supervised approach. Our second goal is to create a coding base, that does not stops to be useful as soon as this works is terminated, but keeps being relevant, and allows for an easy implementation of other newer self-supervised methods, without requiring to write the whole project from the ground up.

Chapter 2

Background

2.1 Neural Networks

As previously discussed in 1.1, Artificial Neural Networks (ANNs) take relative inspiration by the architecture of the optical nervous system. The general structure of a ANN is made up of multiple nodes (or neurons), connected by set of weights (or axons), which enables them to work with each other [5]. The neurons in the model works together to generate an output, which is, in the case of classification models, a probability distribution over the set of possible labels. All networks are typically divided into three sections, as shown in 2.1. The input layer has one neuron for each input value. If the input was an image, each pixel's color would have it's own input neuron. The hidden layer then connects the input with the output layers, and can have as many neurons as it is required. The number of neurons has many effects on the behaviour of the network, some of them being changing the complexity of the model, affecting the generalisation power and the learning process and so on. By increasing the number of stacked layers in the hidden part of the network, we can go from shallow to Deep Learning Models.

The architecture shown above represents a Fully Connected Network (FCN), meaning that every neuron is connected with each other. This design choice means that the number of total connections in a network is so high, that deep networks are too big to be trained effectively.

Convolutional Neural Networks (CNNs) represent the solution to this problem. CNNs now present convolutional windows, typically called kernels, that essentially group together a number of input values equal to the size of the filter, squared. The typical parameters that characterize a convolutional filter are the kernel size, effectively the size of the square window, and the stride, the amount of pixels that the filter moves each time. A more intuitive representation of the concept of convolutional filter can be seen in 2.2

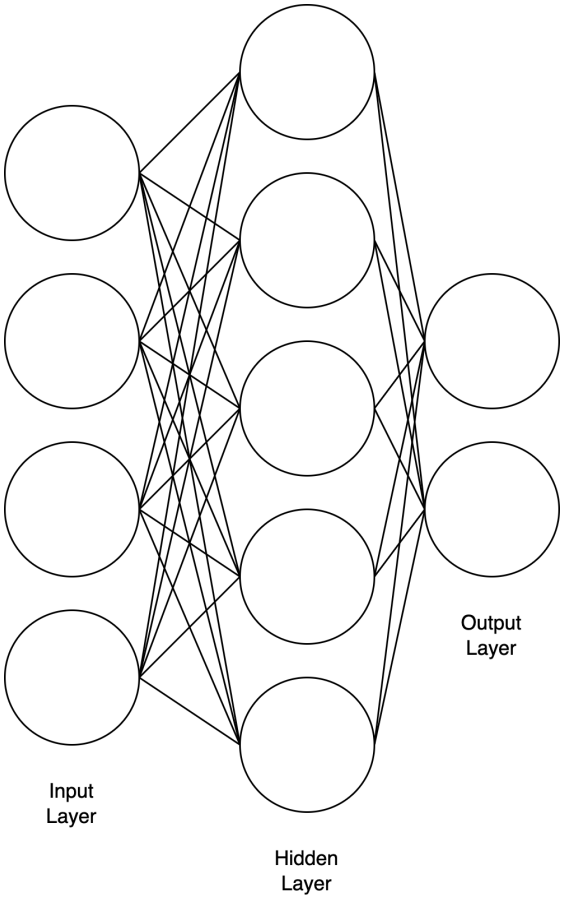

Figure 2.1: The typical Artificial Neural Network Architecture for a classification model. A first input layer on the left presents the same number of neurons as the number of inputs coming from the outside world. Then the hidden layer(s) can be multiple, and determine the depth of the model. Lastly, the output layers presents one neuron for each output class

CNNs can be applied to two main paradigms, Supervised Learning (SL) and Unsupervised Learning (UL).

Supervised learning means learning a mapping function between the inputs and a set of labels. To carry out this type of learning, inputs needs to be pre-labelled. For each input sample, a CNN trained to perform Supervised Classification, will output a probability distribution over the set of labels (often also called targets). The objective of this learning paradigms is to minimize the error function between the actual true value (often referred as ground truth), and the network's predictions. We do not dive further into Unsupervised learning since it will not be used in our work.

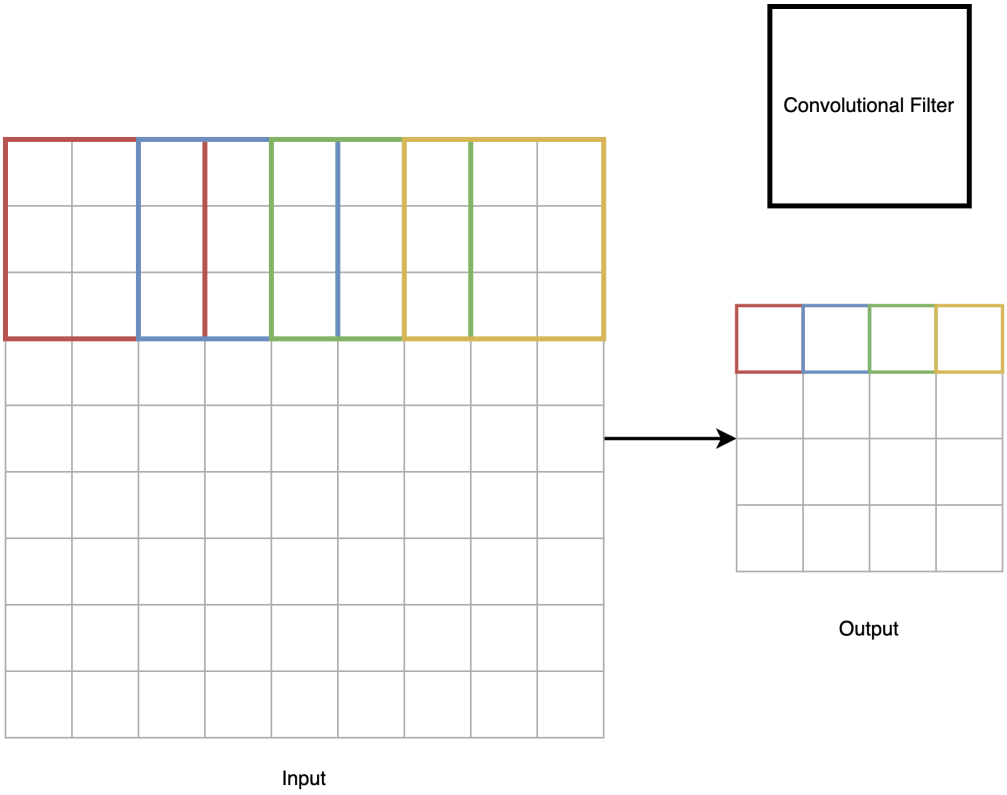

Figure 2.2: The picture shows the first four steps of a convolution. On the left we can see the input image which we set it to be 9x9 pixels. In the top-center part of the image we can see the filter, which we set it to be of size 3 (3x3 pixels), and with stride 2 (the filter travels 2 pixels towards the next position). Each big colored square on the input image represents the next position of the filter in the image. Each small colored square in the output image on the right, represent the output of the convolution. It is clear how the filter size and stride affect the size of the filter output.

The learning of neural network depends on multiple parameters, one of which is the loss function. Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimise through learning. Each neuron will still receive an input and perform a operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the network will still express a single perceptive score function (the weight).

2.2 Bottleneck of supervised learning

Supervised learning is the standard approach to neural network's training. The task of supervised learning consists of learning a mapping functions based on input-output sample pairs [6]. In order for the model to learn an approximately correct mapping function, the training samples need to be representative of the world that is generating the data. To increase the confidence of having a sample which is representative of the real world, we can increase the size of the sample, and this means adding more data and more labels to our dataset. When it comes to adding labels, most often the process requires a human to manually examine each sample and give it an annotation, according to the content. This very process can be more or less expensive in terms of time and money. When building labels for a segmentation dataset, for instance, we need to have a label for each pixel of each image, and this can require quite an amount of resources, when iterated over a sample size of thousands of images. While instead for classification tasks, we only need one target value per image. The cost of annotations varies also depending on the domain of the data. For instance, annotating an image coming from ImageNet [7], does not require any special skill from the annotator, since the dataset mainly contains everyday objects such as chairs and tables, or species of animals, such as Labrador retriever or Siamese cat. On the other hand, if we approach more specific domains such as biomedical images, the number of people bearing the skills necessary to write correct annotations drops drastically, and with it, the total amount of annotation we can possibly get access to also decreases. For this reason, often the high cost of producing an annotation translates in a lack thereof.

Given the cost of labels that we have just discussed, many methods were developed to address the lack of labels, the most notable of them being Transfer Learning, Domain Adaptation and Self Supervised learning.

2.2.1 Transfer Learning

Transfer Learning addresses the lack of annotated data by training a model in a two step approach. First, it trains a model in a Supervised fashion on a large annotated dataset. Second, it transfers the learned knowledge to a second task, which has to be somehow related to the first one. In order to successfully apply Transfer Learning, the model must have a certain generalisation power, in order to be able to transfer the knowledge obtained on the first task, to the second, related task. Moreover, the two tasks need to be somehow related (i.e. it is sensible to think that Transfer Learning applied to a model trained to recognize trucks, could yield a model that is able to recognize motorbikes, and would fail if the end task is to recognise planes)

2.2.2 Domain Adaptation

Domain Adaptation is a special case of Transfer Learning, in which the feature space of the source and domain target are the same. Domain adaptation techniques address the gap that exists between the source label-rich, and the target label-poor data distributions.

This works mainly focuses on the Self Supervised approach to reduce the annotation requirement for label-hungry models, by learning meaningful representation via a pretext task, that does not requires effort to generate labels.

2.3 Embeddings

Before jumping into the definition of Self-supervised learning, we must first introduce the concept of embedding, and how it relates to the concept of neural network. We define embeddings the relatively low-dimensional representation of the high-dimensional object served as input to the neural network. After each convolutional layer of a common DL architecture, we obtain an embedding, which is often referred to as hidden representation. Hence, we can state that at each layer, we obtain a different embedding of the starting image. What we are commonly interested in, is the last level of embeddings, which will also be the most compressed in the case of the architecture in the picture above. We can think of this embedding as the projection of our high-dimensional input image (i.e. 224x224x3) onto a lower-dimensional space. Ideally, if a model is able to convert an image into a meaningful embedding, the embeddings themselves will have some important properties. As an example, similar images in terms of content, will ideally yield embeddings that are close in the feature space (by means of distance metrics i.e. cosine distance) Moreover, powerful embeddings can be reused across different domains.

2.4 Self-Supervised Learning

As previously mentioned, Self-supervised learning is a family of methods that aim at reducing the amount of supervision needed by classical Supervised Learning methods. The concept of SSL first emerged in the field of robotics, where it referred to the automatic labelling achieved by crossing the inputs obtained by different sensors [8]. The same concept has been further explored by computer scientist in the field of deep learning. Now we can summarize the most important features of SSL as follows:

- Use a semi automated process to obtain labels from the data itself
- Predict part of the data from other parts of the data itself

More in detail, concerning the second point, we can think of situations where we corrupt or erase part of the data, and learn how to recreate them from what is left untouched of the original data.

What we have just described, is what we will call from now on the pretext task. The pretext task (or objective) is, as we just described, a task where we either:

- Perform a classification over the labels obtained by the semi automated process described above
- Simply predict part of the data after having corrupted the original data as described above.

The two objectives described above, clearly do not have the need of "hard" labels (i.e. manual annotation of the content of an image, after the active process of understanding each image by a human annotator). Instead, we are completing a task that needs semi automated labels (first point), or no labels at all (second point). The power of Self-Supervised learning, stands in the art of defining a proper pretext task. By defining a good objective on unlabelled data, ideally, we will then obtain a model that is able to extract embedding that are meaningful, for almost any downstream task (i.e. image classification-based objectives or object detection).

In the following sections, we are going to shed some lights over the two main branches of Self-Supervised learning: Constrastive and Generative Learning [9].

2.4.1 Contrastive

Contrastive learning is a learning paradigm that uses similar objective functions to Supervised Learning. The core of the method stands in contrasting the latent representation of different input pairs, while pushing together latent representation of similar input pairs. The Contrastive Learning paradigm divides again in two major branches: context-instance contrast and instance-instance contrast.

Context-Instance contrast

We use Context-Instance contrast in a SSRL framework when we exploit the relations that a (local) portion of data has with respect of itself (global), for this reason we can think of it as a local-global contrast. We will focus particularly on that portion of methods called Predict Relative Position (or PRP for short). This approach main goal is predicting the mutual positions among local components of the input. In this case the global context serves as ground truth, required to predict such relative position. In the following sections we will dig deeper in two methods belonging to this category: Rotation Prediction and Jigsaw Puzzle Solving.

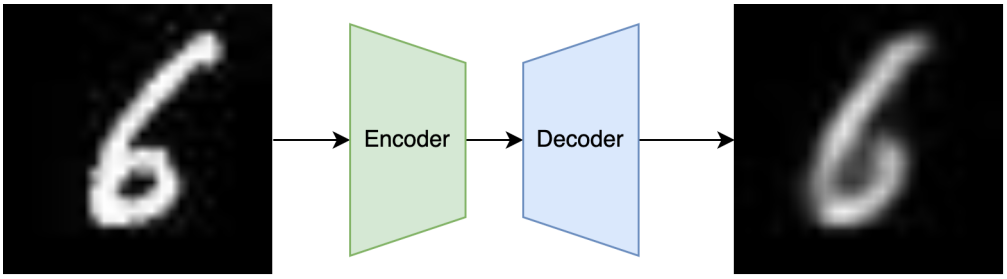

Figure 2.3: The figure shows the sample output of an encoder trained to reproduce a hand-written digit, from the MNIST dataset.

Instance-Instance contrast

Instance-Instance contrast is an alternative to the Context-Instance Maximizing-Mutual Information (MI) approach, that has been proved not to bring any actual improvement. Instance-Instance contrast instead models the the relationship among instance-level representation of different samples.

2.4.2 Generative

Auto-encoder Models

The auto-encoder (AE) models are the most diffused generative models, because of their flexibility. Blotzmann Machines can be seen as a first special version of auto-encoder. The objective of an auto-encoder model is to output a reconstruction of the input, which can be purposefully corrupted in some cases 2.3. It is generally composed by a feed-forward neural network, whose training is aimed at reproducing it's input at the output layer. The structure is composed of an encoder $h = f_{\text{enc}}(x)$, and a decoder $g = f_{\text{dec}}(x)$. The encoder and decoder function are composed to produce as output, a reconstruction of the input, in other words $\hat{x} = g\left(f\left(x\right)\right)$, where the objective is minimizing a certain reconstruction loss between the original input x and the reconstructed input $\hat{x}$, in other words, making x and $\hat{x}$ as similar as possible

Chapter 3

Datasets

The data we use use in our experiments can be classified generally as digital microscopy data. Digital processing of microscope images has been around since before the advent of Deep Learning, and has been developed as a tool to extract quantitative information about the specimen in exam. With the development of computer vision, the ability of digitalizing a microscopy picture has enabled the possibility of running deep learning models on this kind of data.

3.1 Mesothelioma dataset

Malignant pleural mesothelioma, is a tumor induced by the contact with asbestos and it's diagnosis represents a critical challenge for pathologists [10]. The diagnosis of this kind of tumor comes from histologic evaluation. The very diagnosis is based on morphological assessment, and has to be supported by radiology assessments, morphological findings and molecular tests (the latter only more recently) [11] Most commonly, mesothelioma is found in the pelura or the peritoneum. The dataset used in this work, referred to as MESO for short, deals with Epithelioid mesothelioma versus all other kind of mesothelioma. The WSIs in the dataset contains pictures of tissues affected by either one of the aforementioned cancer types. Epithelioid mesothelioma it is the most common type of mesothelioma, as it accounts for 50% to 60% of cases. Sarcomatoid mesothelioma instead, is a way rarer form of mesothelioma, as it accounts for 10% to 20% of cases. The remaining percentages of cases present either Desmoplastic mesothelioma, or a mix of the others. The insurgence of all the subtypes of mesothelioma cancer is caused by exposure to asbestos, a group of minerals used in construction for their resistance to corrosion and heat.

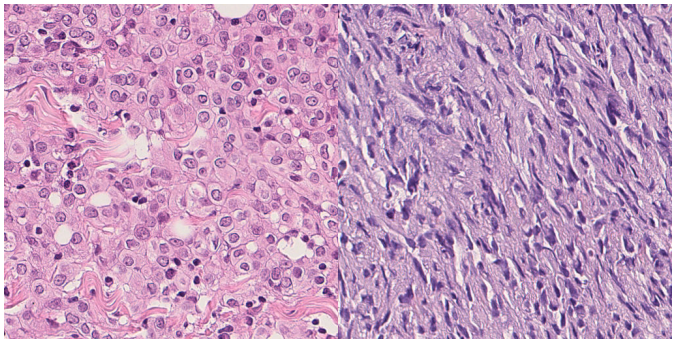

Figure 3.1: The images show a sample for each one of the classes in our MESO dataset. From top-left to bottom right the image shows: Epithelioid, Non-Epithelioid

3.1.1 Epithelioid Mesothelioma

Epithelioid mesothelioma present itself in a wide variety of histological patterns. Multiple different patterns can be observed in a single neoplasm, and typically there is always one single predominant pattern. Cells cointained in the epithelioid mesothelioma appear oval or cuboidal, and mimic the mesothelial cells that occur after incurring in injuries of various types. Mitoses are infrequent except for the more poorly differentiated epithelioid neoplasms, which are uncommon (6). Trabeculara, solid, tubulopapillary and psammoma can be considered the second most common histological pattern of epithelioid mesothelioma.

3.1.2 Sarcomatoid Mesothelioma

Sarcomatoid mesothelioma is the most agressive, albeit rarest of the three mesothelioma histological types. The pattern of the sarcomatoid histological type can be individuated by a proliferation of fascicles-arranged spindle cells, characterized by different mitoic activity and nuclear apatia grades.

3.1.3 Desmoplastic Mesothelioma

Desmoplastic Mesothelioma appears as a patternless proliferation of the bland spindle cells inside a band of collagenous stroma.

The annotations of this dataset were obtained by a pathologist. For our work, this dataset presents the challenge of a binary classification between Epithelioid and Non-Epithelioid mesotheliomas.

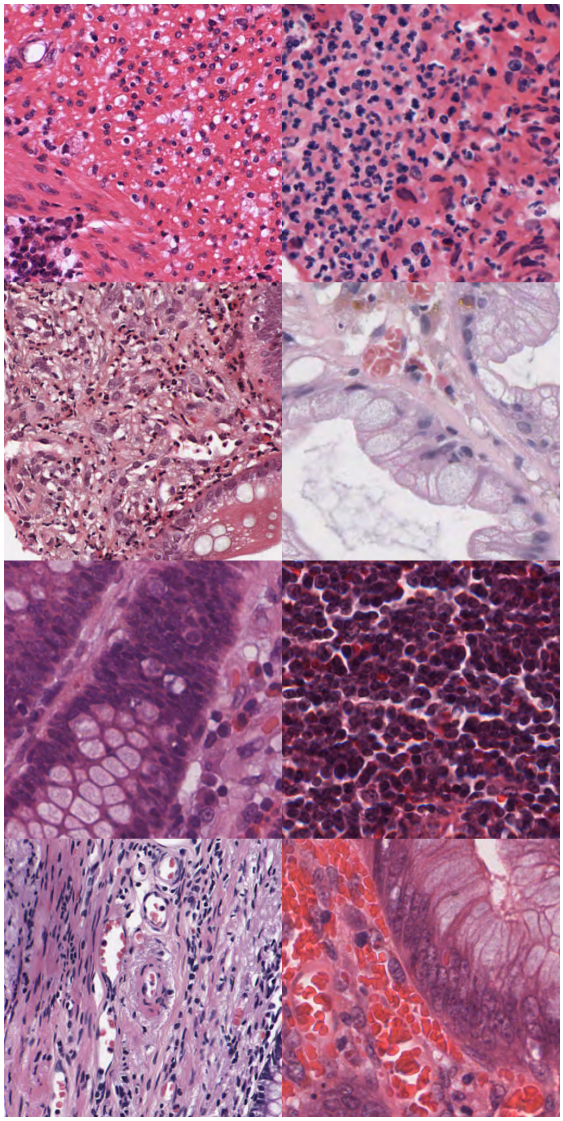

Figure 3.2: The images show a sample for each one of the classes in our CRC dataset. From top-left to bottom right the image shows: Ulcerative Colitis (UC), Adenocarcinoma (AC), Juvenile polip (J), Serreted Adenoma (Serr), Healthy (H), Primary colorectal lymphomas (PCL), Ulcerative Colitis (UC), Tubular Adenoma (T)

3.2 Colorectal Cancer Dataset

The second study case tackled in this work is a dataset concering Colorectal Cancer disease. This study case is particularly interesting because of the class distribution. It is a multi-class classification problem, and the different categories

are characterized by an important intra-class variability level [12]. This makes it so that the case study is relevant to test all those classification techniques that do not rely on pre-designed features, but instead discover those feature via gradient-based learning.

Data coming from the World Health Organisation show that the colorectal cancer stands in the second place as the most common tumor. The CRC is characterised by a high survival rate: around 90% rate of 5 year survival if diagnosed at an early stage. On the other hand, only around 40% of the polyps are removed before developing in tumors. It is precisely this low early-diagnosis rate that makes it among the main causes of cancer-caused in the occidental words.

The colorectal cancer originates in colon or rectum lining, and more precisely from the epithelial cells. This typically happens in the innermost part of the intestinal wall. The typical diagnosis process of colorectal cancer requires visual examination of a sample of resected tissue, H&E stained and observed under the microscope. The diagnosis of malign tissue is done based on the observation of changes in the organization of the tissues. Healthy tissue is generally composed by glandular structures made of epithelial cells, and filled with non-epithelial cells. The adenoma is the main benign precursor of colorectal cancer, and it often manifests as elongated nuclei arranged in stratified fashion. Adenomas instead show irregularities with respect to healthy tissues, and can evolve into multiple types of precancerous growth. Adenocarcinomas, instead, manifest by producing abnormal glands and infiltrating the tissue surrounding the spot of interest.

The main challenge that this dataset poses for our work is the high inter dataset and intra-class variability. This very challenge is not only to be found in the CRC dataset, but is intrinsic of any histological imaging dataset.

- Adenocarcinoma (AC)
- Healthy (H)
- Juvenile polip (J)
- Moucinous Carcinoma (MC)
- Primary colorectal lymphomas (PCL)
- Serreted Adenoma (Serr)
- Tubular Adenoma (T)
- Ulcerative Colitis (UC)
- Villous Adenoma (V)

3.3 Details

3.3.1 Image Formats

Digitalized microscopy images comes in different formats, depending on the brand of the instrument used to take the picture in the first place.

- *SVS*: The SVS file format was originally developed by Aperio. This is a pyramid file, meaning that the file contains multiple instances of the same image, at different resolutions. The images contained in the file are typically tiled, with a fixed dimension, usually about 240x240 pixels.
- *ndpi*: The ndpi format was originally developed by Hamamatsu. This image format contains many JPEG tiles of the original images, since JPEG has a maximum dimension of 65535 pixels on one side.

The reason of these exotic file extension is to be found in the original size of the images. Many of the images we are going to use measure well over 1k pixels per side. This size would make it impossible to display them as a common digital image. For this reason, the images are tiled into smaller sub-images, small enough to be displayed.

3.3.2 Region of Interest

One of the most common problem when it comes to image classification of WSIs is the heterogeneity of the image content. By definition, for an image classification task, we have one class label per image. Trivially tiling an image and applying the same label to each tile can lead to a confused model. Since the image is a picture of a big portion of tissue, and the tumor is typically concentrated in a fairly limited portion of tissue, the network could be mislead by being fed patches of multiple tissue samples (not containing the tumor itself), together with the label saying that that very particular patch IS tumor. In order to avoid this, some images may come with annotation files in XML formats, that denote homogeneous regions of tissue called Regions of Interest (or ROIs for short). This kind of annotations consist in x,y coordinates of a bounding box that can be cropped away from the rest of the image, tiled, and used as training data for the model.

3.4 Processing

Both dataset's images are in the form of whole slide images. For this reason, given the constrains imposed by the models architectures and the available resources, the images need to be resized and/or cropped before being used in the training

pipeline. All the available images are used at the maximum zoom level available. Once the maximum zoom level image has been extracted, a tiling procedure takes place, where a given resolution is inputted (512x512 pixels for our work), and all available, non-overlapping tiles of that size are returned. Tiles are then filtered in order to discard portion of images where there is small to no tissue at all. The filter is done by computing the mean over each pixel's standard deviation over all channels 3.1, and compared against a hand-picked threshold

$$v_p = \frac{1}{w+h} \sum_{i=0}^{w} \sum_{j=0}^{h} \sqrt{\frac{\sum_c^3 \left(x_{i,j}^c - \mu_{i,j}\right)^2}{3}} \tag{3.1}$$

where $x_{i,j}^c$ is the value of the channel c of the pixel at position (i, j) and $\mu_{i,j}$ is the mean over the RGB channels of the pixel in position (i, j).

Once the tiles are extracted from each image, they are saved in PNG format in a separate directory. Since not all the images present the same original size (or number of valid tiles), we discard any number of tiles necessary to balance the number of tiles per each class. This choice of discarding is not only limited to the balancing problem, but it is also necessary to reduce the amount of data to accommodate the resources used to run the project.

To build the training, test and validation version of the dataset, the slides are first split by patient, and we make sure that all the three datasets contain different patients, to ensure generalization power. One thousand images are selected for each class, for each dataset.

If the pretext tasks allows it, we perform stochastic data augmentation at training time, where transformations are applied randomly.

3.5 Tiling

The digital representation of tissue samples (WSIs) have typically very large resolutions (i.e. 3000x3000 pixels). For this reason, the diagnosis task of a single WSI is split into the individual diagnosis of smaller portions of WSI (of a resolution compatible with Neural Networks training). After having obtained a prediction for each of the single tiles of a slide, different methods can be adopted to extract the global diagnosis from the local ones of every single tile. In this work, we will focus only on the neural network part of the task: we will explore and benchmark methods whose performance will be measured on the single tile, without aggregating the results at WSI level.

Chapter 4

Methods

As explained in the previous section, our training process will be split into a pretext training, and a downstream training. In the pretext training, we will train the network on the pretext-task that will allow it to learn how to embed meaningful features from the input space to the latent space. The downstream task will be our original task that we would have carried out, if we did not choose to use SSRL.

In our work, we will explore multiple pretext-tasks, and multiple dataset sizes. In order to make the results of these experiments comparable, we must fix some objects, among which the network architecture.

To achieve this in the pretext-training, we adopted a modular approach to our network. By definition, different pretext-tasks require different outputs (i.e. for an auto-encoding task the output will be of the same shape of the input, while instead for a rotation classification task, the output will have a shape of four, one for each rotation multiple of 90 degrees). For this reason, we cannot simply fix one architecture, since it will not fit all the tasks. Instead, we split the part of the network that remains the same among all the pretext tasks, and we call it backbone. We can also define the Backbone that part of the network whose output is our latent representation, in short, the backbone is the feature extractor of the network. In our experiments, we choose the backbone to be a ResNet18 network, because of its popularity among different papers facilitates comparison with other works. We call the remaining of the architecture Head, and this is the part that changes shape depending on the pretext-task. We can say that the Head is task-specific, and will be discarded after having completed the pretext task.

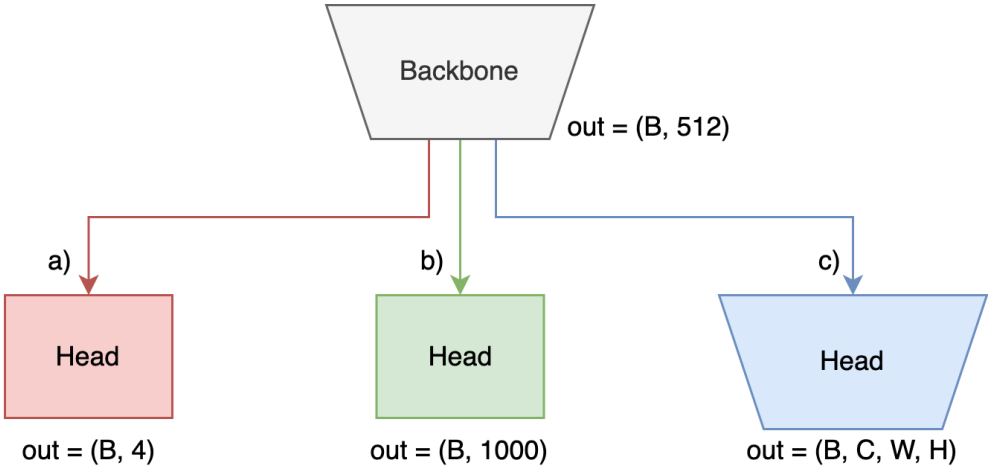

Figure 4.1: A schema of the architecture used in the pretext side of the experiments. On the top side the backbone network. On the bottom side, three network examples with the respective output shapes, for different pretext tasks. On path a we see a Head network for rotation prediction. On path b we see a Head network for Jigsaw Puzzle solving. On path c we see a Head network for Auto-encoding

4.1 Baseline

Our main goal, as stated in 1.5, is to explore SSRL and find out if and which pretext task brings the best improvement for classification of Whole Slide Images (WSIs) containing tumor cells. More importantly, we try to measure if and how much better is any pretext-task with respect to the classic Supervised Learning approach. It comes natural that in order to asses the improvement, we must first measure the baseline performance of the Supervised Learning that we are trying to surpass.

Technically, our baseline consists of a ResNet18 backbone and a fully connected head, with the output shape adapted to each dataset. As previously stated, the baseline consists in training a network (a ResNet18 backbone and a single layer Perceptron head) in a Supervised classification of images.

4.2 Pretext Training

We perform one training session for each pretext-task that we analyze, namely Rotation Prediction, Jigsaw Puzzle Solving, Auto-encoding. Each pretext-training, has its own unique configuration and hyper-parameters, as described below.

On top of the tasks above, we also include ImageNET Pre-Training to our approaches.

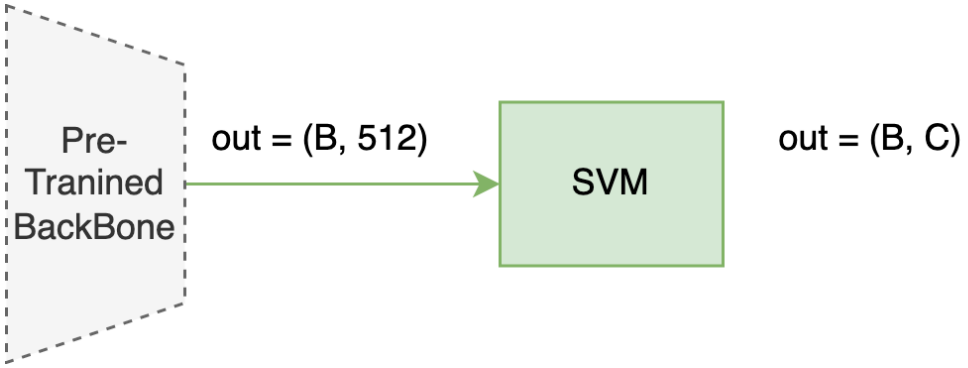

Figure 4.2: The figure shows the Downstream task pipeline. On the left the pre-trained backbone. The dashed contour shows that the weights are frozen. The code outputted by the backbone is then fed to the Downstream SVM which performs classification over the codes.

4.2.1 ImageNET Pre-Training

Although not properly a pretext task, the pre-training on ImageNET represents one of the most common way of initializing a neural network for a downstream task. One of the definitions of pretext-training that we have given above is a smart initialization of the network's parameters, able to extract a meaningful embedding from the input images. For this reason, we consider the pre-training on ImageNET as a weights initialization for our network. This allows us to study how effective the ImageNET initialization is, when applied to a completely different domain such as the biomedical one. Since we are using a ResNet18 backbone (and a Single Layer Perceptron head), we are able to exploit the pre-trained weights configuration available online, without the need to fully train the network on millions of images.

4.2.2 Random Initialization

As the previous subsection, also random initialization is not a proper pretext task. Random initialization in this context means that we are using as a feature extractor for the downstream task, an untrained network, whose weights follow the initialization distribution.

4.2.3 Rotation Prediction

The rotation prediction task is self explicatory. At data loading time, we sample from a uniform random distribution over the 4 multiples of 90 degrees angles, and apply the said rotation to the image at hand. The goal of the task is essentially a classification task over 4 classes, which represent the 4 multiples of 90 degrees

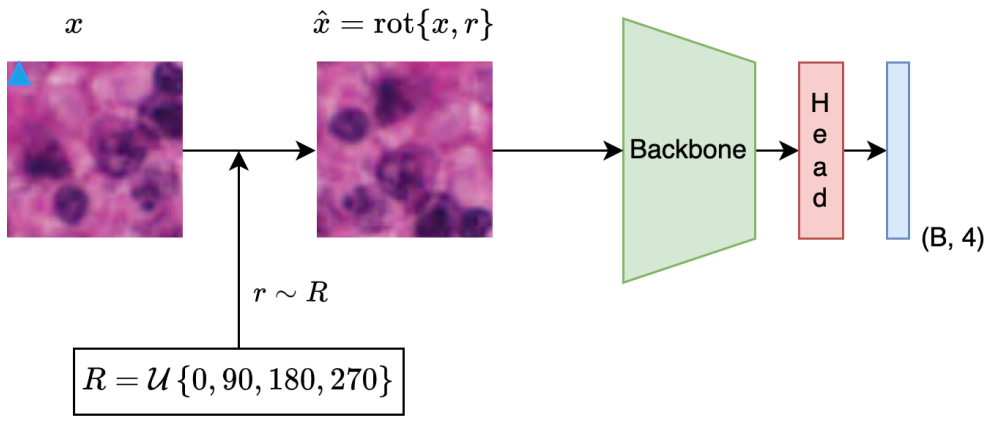

Figure 4.3: In the rotation pretext task, each input is rotated according to a random multiple of 90 degrees, sampled uniformly from the set of multiples smaller than 360 degrees. The rotated image is then fed into the model that predicts the actual rotation angle.

applied to the images. Our expectation from this task applied to the biomedical domain, is that it should not bring great improvement. The reason why is to be found in the essence of the task itself. Predicting the rotation of the picture of a car, can teach the network a meaningful representation of the car object itself, since virtually in any picture of a car, the sub components have a fixed relative position (the wheels are above the ground, and the windshields are on the top part). The shape and dispositions of cells do not have instead a predefined relative positioning of the sub components, for this reason we do not expect the task to bring any improvement with its initialization of the network weights.

4.2.4 Jigsaw Puzzle Solving

Solving jigsaw puzzles has been always associated with learning since they have been invented. They are typically used to access and test the solver's visuospatial processing abilities. By teaching a network to solve this task, we aim at teaching the network how to extract a visuospatial representation of the input images.

The architecture of the network has been inspired by [13]. The simplest approach to the task would be to stack the tiles along the batch dimension, and consequently increase the number of filters in the first layers of the architecture. What happens with this approach, is that the network focuses on low level cues to solve the puzzle. Although the above behaviour is what us humans are used to do, it does not require any understanding of the content of the tiles (and the whole puzzle), hence, it will not yield to generalizable image embeddings. The way to avoid this trivial solution

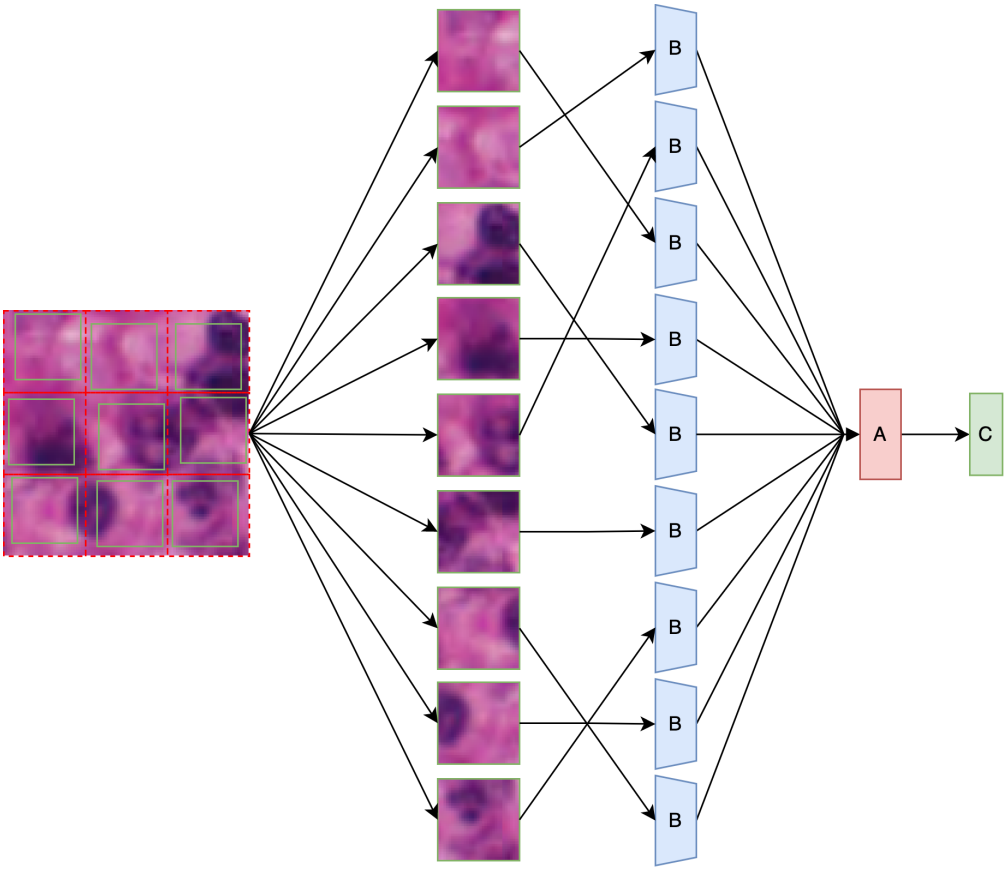

Figure 4.4: The picture shows from left to right the image being tiled, with the tile frames being chosen randomly allowing for some pre-defined jitter, to avoid adjacent tiles. Next the tiles are extracted and scrambled according to a pseudo-random permutation. The tiles are then fed to the siamese backbone. After the backbone is done extracting the features from each individual tile, the features are concatenated, and fed through the head. In the end, the head outputs, for each original puzzle, a vector of size C, where C equals the cardinality of the permutation set.

by the network is by feeding the tiles to the network one by one, until the fully connected layer. This way our ResNet backbone is treated as a siamese network, where nine batches (one for each tile of the puzzle) are fed to networks sharing the same set of weights. After the feature extraction part, the embeddings are concatenated to a single vector, and passed through the head of the architecture, where the global context in re-injected in the process, and the network is able to output the solution of the puzzle.

It is worth to further elaborate on the actual output of the network, and what

we mean when we refer to "solving" the puzzle. To start, it is important to mention that after the tiling process of the input, the resulting tiles are scrambled. Scrambling consists in extracting one permutation randomly from a set of pre-computed permutation. These permutations indicate the order of the tiles in the scramble, and are sampled from a uniform distribution over a pre-determined set of scrambles. The characteristic of the set are important to determine the complexity of the task itself. The task is essentially a classification task over a number of classes equal to the cardinality of the permutation set.

1. The cardinality of the set of permutation determines the ambiguity of the problem. In other words, the number of unique permutations in the set translates in the number of classes among which the network have to choose for its final prediction. By increasing the cardinality of the set, we increase the complexity of the pretext task, but we will also increase the performance of the downstream task, and vice-versa.

2. The heterogeneity of the scrambles inside the set of pre-computed permutations is measured by the average Hamming distance of all the permutations. Greater Hamming distances yield better performances both in the solution of the puzzle, and downstream performances.

The network's output are then checked against the ground truth, and a cross entropy loss is used to back-propagate.

4.2.5 Autoencoding

The autoencoder is trained with the objective of reproducing as output, an image which is as much similar to the input as possible. The auto-encoder architecture is made up of an encoder, which takes an image as input, and compresses it into a latent code, and a decoder structure, which tries to re-build the image starting from the code. A correctly trained encoder is able to produce a meaningful representation of the inputs in the latent space, and this is what we leverage by using the auto-encoder task as a pretext. For our purpose, our encoder structure is our backbone, and it is composed by a ResNet18, up until the last average pooling layer. The decoder instead, is a chain of upsamples and convolutions, carefully crafted in a way to recover the original image resolution as output. The auto-encoding pretext-task falls into the Generative pretext-tasks category.

4.2.6 Training Parameters

For all the training, we adopted Adam as optimizer. The learning rate is tuned according to the complexity of each pretext task, more detailed information can be

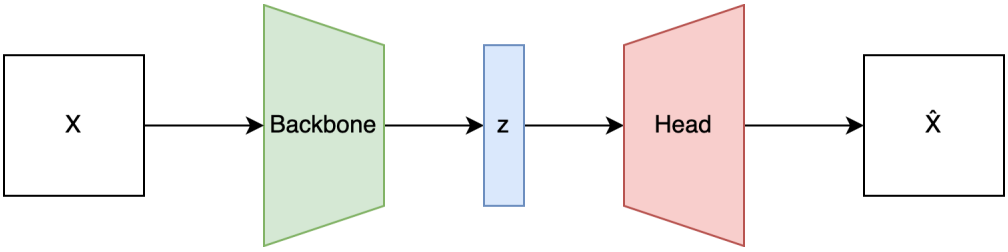

Figure 4.5: The encoder architecture in terms of Backbone and Head. The input is fed to the Backbone-Encoder part, composed of a ResnNet18 model up until the last average pooling layer. The latent code is then fed to the Head-Decoder part, composed of multiple stacks of upsample and convolution blocks. The output of the model is a reconstruction of the input.

found in the table **??**. We also employ weight decay and we set it to 0.0001 for all training instances. In all our training, we include an early stopping criterion, namely the pytorch lightning implementation EarlyStopping callback. This early stopping criterion allows us to monitor a desired (validation) metric, and stops the training after a desired number of epochs without a significant improvement. The parameters of the criterion vary for the various pretext tasks, and are reported in the table 4.2. We make use of a learning rate scheduler, and more precisely, we use a custom configuration of ReduceLROnPlateau from pytorch, for each individual pretext training. We choose this schedule because it allows to lower the learning rate each time the improvement over a given metric is lower than a certain threshold. This particular scheduler works well in combination to the aforementioned early stopping criterion, and they work best when the latter is set to act after a number of epochs which is multiple of the one set for the former.

Task	*lr*	*scheduler*	*optimizer*
Baseline	0.003	CosineAnnealingLR	adam
Autoencoding	0.001	ReduceLROnPlateau	adam
Jigsaw	0.001	ReduceLROnPlateau	adam
Rotation	0.001	ReduceLROnPlateau	adam

Table 4.1: The table shows the configuration of learning rate, scheduler and optimizers for the various training tasks

Task	*metric*	*min_delta*	*patience (epochs)*
Baseline	CE	0.005	21
Autoencoding	MSE	0.0005	11
Jigsaw	CE	0.001	11
Rotation	CE	0.001	11‘

Table 4.2: The table reports the configuration of the Early Stopping callbacks for the various training tasks

4.3 Downstream Training

As we previously mentioned in section 1.4, our end goal is the automated diagnosis of cancer, starting from the classification of small sub-images, obtained by tiling the whole slide. Hence, our downstream task can be defined as standard classification. Our downstream classification uses a Support Vector Machine Classifier, trained over the features extracted by our pre(context)-trained backbone network.

To understand until which point SSRL can help overcome the lack of data, we perform the downstream training on different dataset sizes. The aim is to assess how much data is few data. If we consider 100% the dataset size fed to the pretext training, we trained the downstream model with 5%, 10%, 20%, 50% and 100% of data. What we expect to see from these experiments is a slow drop in performance, an possibly an increase in performance with respect to the baseline at 100%.

For all the downstream training, we use a Support Vector Machine Classifier implementation from scikit-learn. At this point in the training we are left with a backbone and head networks pre-trained of the pretext task. Since the head network is, in all pretext task, a portion of network that is needed exclusively to adapt (or decode) the latent code to be compatible with the respective pretext tasks, we can discard it. The backbone instead is the essential block needed to compress and transform the input images. For this reason, when we load the best checkpoint from the previous training, we only load the weights relative to the backbone architecture. Once this is done, we proceed by freezing the weights of the backbone. Freezing weights is a concept that relates to the gradient based training procedures that all neural networks undergo during training. While using the backbone network as a feature extractor, we do not need to further modify the weight schema by effect of back-propagation, hence, it is best practice to freeze the weights, which effectively means no shift will be applied to the weights, even if, for some reason, the gradient is computed.

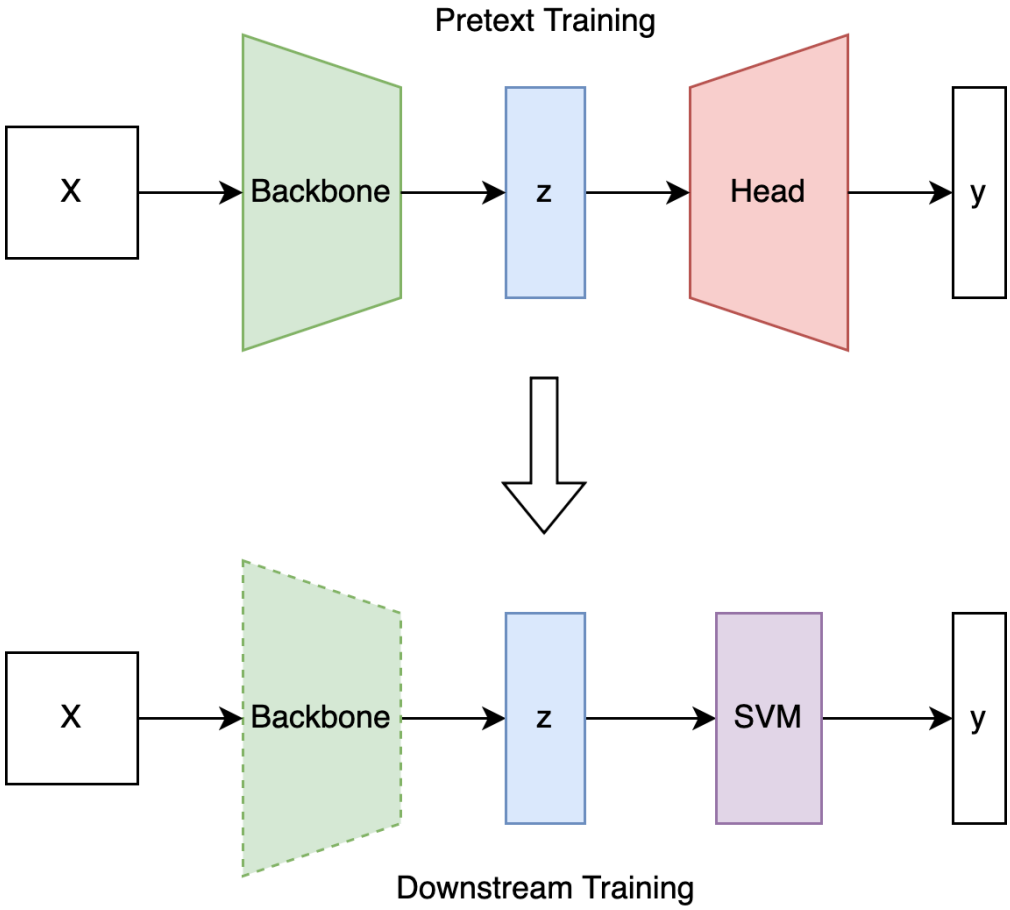

Figure 4.6: The picture shows the transition from pretext (above) to downstream (below). During the pretext task, the objective of the training is to learn a meaningful low-dimensional representation of the input space, with a "pretext". Once the training is over, the head is discarded, and the backbone is frozen. In our downstream training, the frozen backbone is used as a feature extractor, and a SVM is trained to perform classification of the latent representations of the input images.

Chapter 5

Results

5.1 Mesothelioma

For the MESO dataset, results of the various pretext methods are grouped in Table 5.2, while the baseline results are shown in table 5.1. The table shows weighted test F1 scores of the downstream classification, for the different training set sizes, and pretext-tasks. It is clear how the Auto-encoding and Jigsaw pretext tasks perform better than any other methods. When lowering the training set size to the lowest size though, pre-training with an auto-encoding tasks ensures that the downstream performance still outperforms the baseline training, with the full training set size. This is an very important results since it proves that generative pretext-training represent a powerful tool to use when lacking annotations. All the other pretext training, even if they do not perform as well, they still represent a significant improvement over the non pre-trained baseline. The Rotation task shows very similar values to the random task, we think this is because in order to solve the rotation task itself, the network does not need to learn features that are meaningful to the downstream task. The ImageNet pre-trained backbone performs better than expected on images that are far from the original data distribution of the ImageNet dataset. This means that classifying natural images leads the network to encode high-level features such edges and patterns, that still are very useful to classify biomedical images of this kind.

5.2 Colorectal Cancer

For the MESO dataset, results of the various pretext methods are grouped in Table 5.4, while the baseline results are shown in table 5.3. While the general consideration done in the previous section still holds also for the CRC dataset, here we can see a slightly different situation. The gap between the baseline and our best

	Dataset Size (%)				
	5	10	20	50	100
Baseline	0.684	0.778	0.808	0.885	**0.897**

Table 5.1: The table shows the test f1 weighted scores for each of the baseline training applied to the MESO dataset.

	Dataset Size (%)				
Pretext method	5	10	20	50	100
Autoencoding	0.908	0.909	0.927	0.956	**0.957**
Rotation	0.874	0.897	0.899	0.911	0.912
Jigsaw	0.857	0.863	0.894	0.938	0.952
Imagenet	0.825	0.862	0.880	0.917	0.928
Random	0.896	0.900	0.869	0.920	0.913

Table 5.2: The table shows the test f1 weighted scores for each of the pretext training tasks, applied to the MESO dataset.

pretext method is wider than before. At the same time, now the autoencoding task greatly outperforms every other task. This is probably due to the harder classification tasks, as explained in section 3.2.

	Dataset Size (%)				
	5	10	20	50	100
Baseline	0.577	0.612	0.689	0.738	**0.885**

Table 5.3: The table shows the test f1 weighted scores for each of the baseline training applied to the CRC dataset.

	Dataset Size (%)				
Pretext method	5	10	20	50	100
Autoencoding	0.893	0.917	0.936	0.961	**0.957**
Rotation	0.683	0.724	0.766	0.806	0.817
Jigsaw	0.704	0.741	0.778	0.857	0.922
Imagenet	0.781	0.814	0.851	0.888	0.906
Random	0.705	0.766	0.808	0.855	0.877

Table 5.4: The table shows the test f1 weighted scores for each of the pretext training tasks, applied to the CRC dataset.

Chapter 6

Discussion

6.1 Trade-off between memory size and speed

Device	*Size*	*Peak Transfer Rate*
DISK	1000 GB	0.5 GB/s (Read)
RAM	16 GB	14 GB/s
VRAM	12 GB	20 Gb/s

Table 6.1: The table shows the difference in size and speed of the various memories of the device used to run experiments. The trade-off is obvious, and shows how faster memory, being way more expensive, is typically more scarce than cheaper slower memory.

Having big datasets is often a problem since it yields to a trade off between memory usage and speed. The best way to approach a Computer Vision is to load all the dataset in RAM, and load one batch at a time to VRAM (Video RAM). VRAM is much faster than RAM, but is also in limited size. For this reason, there could be many different configurations in which we could run out of memory:

- *Dataset size exceeds DISK size*: When the size of all the images in the datasets is greater than the available DISK space, the only available option is to store the dataset on a remote server, or cloud, with enough DISK space. This approach comes with the drawback of having to include in the overall computation the

bandwidth and latency of the network connecting the computer to the remote folder.

- *Dataset size exceeds RAM Size*: When the size of all the images in the datasets is greater than the amount of system memory available, it means that not all the images can be loaded in RAM. This also means that when loading images to VRAM, we will have to first read them from the much slower storage memory (DISK). Another solution to this problem would be to simply upgrade the RAM of the system, since this kind of memory has become not so expansive these days.

- *The forward activations of the model exceeds VRAM size*: Before starting the training, the CPU loads into the VRAM the model's parameters. Right after the training started, a batch of data is loaded into the VRAM. Both this two chunks of data do not represent a major problem. To give a quick example, given that all data is represented by 32bit floating points, a ResNet 50 with a batch size of 64 and an input size of 3x224x224 pixels would require 0.13 GB of VRAM. During the training of the model, and more precisely during a forward pass of the batch, the system will need to store all the intermediate forward activations of all the layers, to be able to compute the gradients during the subsequent backward propagation step. These forward activations can easily fill up the VRAM and cause the training to stop if the model and/or the batch size are too big. The solution to this problem is either to lower either the model size, or the batch size or both. Alternatively, a more expensive solution is to purchase another GPU to add to the first one, hence increasing the VRAM size, and running the training algorithm in parallel between the two GPUs.

6.2 Random Initialization's Success

The results clearly shows, that the features extracted by a randomly initialized backbone, still allow the downstream model to achieve a reasonable amount of performance on the data. The reason this happens we think is to be found in the nature of the convolutional layers inside the architecture. We think that even if it uses random weights, a stack of convolutional layers are able to extract a code that is good enough to perform downstream classifications on. In the case of Colorectal Cancer, we even see the Random entry rise above the Rotation entry. In this case, we think that the rotation pretext task not only is not useful for the downstream classification, but that the mapping the backbone learns from solving the rotation task is so far away from a representation useful to solve the classification task, that is counterproductive.

6.3 Further Development

Given the encouraging results obtained by our work, quite a few development directions arise for the future. Primarily, the great success of generative pretext-training methods suggest that further research on more complex models could be beneficial. As such, considering other approaches like Denoising or Variational AEs, or even GANs could yield even better results than those obtained here.

The essential step needed to validate the results clinically, is to further the development to include the possibility of aggregating the tile-level results to slide level. This would enable a researcher to obtain a prediction over a WSI, effectively getting even closer to the automatic diagnosis goal stated in section 1.4.

Another step towards further validation of our work would be increasing the absolute size of datasets. Because of constraints on our resources, the work was not validated on large absolute datasets sizes. This is essential as feeding the network with a considerable amount of images could consolidate even more our findings (or uncover weak spots that we did not see in the current settings).

Chapter 7

Conclusion

The results obtained from the experiments show that the self supervised learning approach represents a promising avenue for the biomedical domain. It is apparent that every domain requires a careful study of a meaningful pretext tasks that can lead to the learning of meaningful features. The results reflected our expectations in the fact that the generative approaches yield better results than every other approach. As expected, the rotation prediction pretext task does not yield interesting performances, because solving that task in a biomedical domain, does not require the learning of meaningful features from the data. This is mostly due to the fact that the orientation of cells and tumors in the WSI do not follow the same rules that the orientation of a physical macro-object follow in a everyday picture.

Another point worth noting is the high performance of the random methods, which seems counterintuitive at first. Our interpretation of this result is that the SVM model trained for the downstream classification, is still able to learn a mapping from the latent code to the class labels, even if the latent code has been extracted by a randomly initialized neural network. This could be because even if the weights are randomly initialized, the spatial characteristics of a convolutional filter (and moreover a stack of them) could still be able to map the image to a space which retains most of the images features.

We also believe that the reason why the generative models yield such great performances downstream is that, opposed to the contrastive methods, they are required to learn the distribution of the whole image. The reason behind this ability could be that for the auto-encoder, each pixel is dependent of the surrounding context, where instead in a jigsaw puzzle solving task, the context for all the pixels belonging to the same tile is always the same.

Our work clearly shows how the supervised learning performances drop drastically when the amount of supervision decreases under the 50%, while instead the self-supervised learning methods are able to retain performance, and the decrease is sensibly slower. Our work shows promising results, with the (generative-based)

self-supervised learning methods surpassing the classical supervised learning also when compared at 100% of training data. This means that on top of being a viable approach when labelled data is completely missing, self-supervised learning could also be used as a pre-training for regular supervised learning task, as a way to increase it's performance.

Bibliography

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. «Backpropagation Applied to Handwritten Zip Code Recognition». In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541 (cit. on p. 1).

[2] Zhengyu He. «Deep Learning in Image Classification: A Survey Report». In: *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*. 2020, pp. 174–177. DOI: 10.1109/ITCA52113.2020.00043 (cit. on p. 1).

[3] Gunter Röth. «Tutorial 1: NVIDIA's platform for Deep Neural Networks». In: *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 2015, pp. XXXVII–XXXIX. DOI: 10.1109/DSAA.2015.7344778 (cit. on p. 1).

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. «ImageNet: A large-scale hierarchical image database». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 1).

[5] Keiron O'Shea and Ryan Nash. «An Introduction to Convolutional Neural Networks». In: *CoRR* abs/1511.08458 (2015). arXiv: 1511.08458. URL: http://arxiv.org/abs/1511.08458 (cit. on p. 5).

[6] Li-Pang Chen. «Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar: Foundations of machine learning, second edition: The MIT Press, Cambridge, MA, 2018, 504 pp., CDN, ISBN 9780262039406». In: *Statistical Papers* 60 (July 2019), pp. 1793–1795. DOI: 10.1007/s00362-019-01124-9 (cit. on p. 8).

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. «ImageNet: a Large-Scale Hierarchical Image Database». In: June 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 8).

[8] Bram Wallace and Bharath Hariharan. «Extending and Analyzing Self-Supervised Learning Across Domains». In: vol. abs/2004.11992. 2020. arXiv: 2004.11992. URL: https://arxiv.org/abs/2004.11992 (cit. on p. 9).

[9] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. «Self-supervised Learning: Generative or Contrastive». In: *CoRR* abs/2006.08218 (2020). arXiv: 2006.08218. URL: https://arxiv.org/abs/2006.08218 (cit. on p. 10).

[10] Kouki Inai. «Pathology of mesothelioma». In: *Environmental health and preventive medicine* 13 (Apr. 2008), pp. 60–4. DOI: 10.1007/s12199-007-0017-6 (cit. on p. 13).

[11] Greta Alí, Rossella Bruno, and Gabriella Fontanini. «The pathological and molecular diagnosis of malignant pleural mesothelioma: a literature review.» In: *Journal of thoracic disease* 10 Suppl 2 (2018), S276–S284 (cit. on p. 13).

[12] Francesco Ponzio, Enrico Macii, Elisa Ficarra, and Santa Di Cataldo. «Colorectal Cancer Classification using Deep Convolutional Networks - An Experimental Study». In: Jan. 2018, pp. 58–66. DOI: 10.5220/0006643100580066 (cit. on p. 16).

[13] Mehdi Noroozi and Paolo Favaro. «Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles». In: *CoRR* abs/1603.09246 (2016). arXiv: 1603.09246. URL: http://arxiv.org/abs/1603.09246 (cit. on p. 22).