Master's thesis in Data Science

Vision Transformers for Analyzing High-Resolution Pathology Images

School of Computer and Communication Sciences

Author
Raphaël Attias

Supervisor
Prof. Dr. Kun-Hsing Yu

Professor
Prof. Dr. Dimitri Van de Ville

Medical Image Processing Lab (MIPLAB)

Ecole Polytechnique Fédérale de Lausanne (EPFL)

Yu Lab (DBMI)

Boston, MA
March 2023

Abstract

Transformers are powerful models that can capture long-range dependencies between data using an attention mechanism. However, applying transformers to medical problems poses challenges due to the high-resolution and complexity of pathology images, as well as the scarcity and noise of labels. In this thesis, we have provided a comprehensive overview of the state-of-the-art transformers method for analyzing high-resolution pathology images, using the glioblastoma dataset IvyGAP and the renal cancer dataset as case studies. We have discussed how to pre-train transformers using self-supervised learning methods that can learn useful representations from unlabeled data. We have also explored how to find regions of interest within a whole slide image using different levels of supervision: self-supervised, weakly supervised and strongly supervised. We have demonstrated that transformers can provide a semantic understanding of the data and outperform convolutional-based models on downstream tasks such as classification and segmentation. Finally, we employ posterior networks to estimate the aleatoric and epistemic uncertainty of ViT predictions and evaluate their usefulness for clinical decision making. We propose a novel method to filter potentially mislabeled data in order to make more accurate and confident predictions. Our experiments show that our methods achieve state-of-the-art results on various glioblastoma tasks and provide meaningful insights into the behavior and limitations of ViTs for medical machine learning.

Keywords: Vision Transformers, Self-Supervised Learning, Uncertainty Estimation, Glioblastoma, Regions of Interest.

Contributions

This thesis advances the field of medical image analysis using transformers in several ways. Besides providing a comprehensive review of transformer methods and their application to medical datasets, we have applied them to the IvyGAP dataset and investigated the issue of potential label noise. To the best of our knowledge, we are the first to perform such an analysis on IvyGAP and to highlight the limitations of this dataset for some tasks that depend on the count of tumor cells. We have also presented evidence for why some labels might be inaccurate or inconsistent based on molecular profiles and histological features. Moreover, we are the first to use posterior networks as a technique to estimate label and prediction uncertainty on IvyGAP. Finally, we have proposed a novel method to filter out potentially mislabeled data points in order to improve the accuracy and confidence of our predictions. We hope that this thesis will inspire further research in this area and contribute to the development of more reliable and robust medical image analysis tools.

Acknowledgements

I would like to express my sincere gratitude to the Yu Lab at Harvard University, in the Department of Biomedical Informatics (DBMI), for hosting me and providing me with an excellent research environment and support. I am especially grateful to Prof. Dr. Kun Hsing Yu for his supervision, mentorship and encouragement throughout this project. He has been a source of inspiration and guidance for me and has taught me a lot about medical machine learning and vision transformers. I would also like to thank Jun Han Zhao from the Yu Lab for his invaluable help and advice on various aspects of the project, from data processing and model implementation to experimental design and evaluation. Lastly, I want to extend a heartfelt thank you to my dear friend Riccardo Cadei, who has been an incredible teammate and friend throughout my Master, and and I am grateful for his unwavering encouragement and dedication

I am also thankful to Prof. Dr. Dimitri Van de Ville from EPFL for supervising my thesis. He has been very supportive and flexible in allowing me to pursue this project at Harvard University and has always shown interest and enthusiasm in my work.

Finally, I would like to thank my family for their constant love, encouragement and understanding during this challenging but fulfilling journey

Contents

List of Figures

1 | Datasets and Motivations

1.1 Introduction

The Yu Lab of the Department of Bioinformatics at Harvard Medical School applies Artificial Intelligence to Quantitative Pathology. The lab analyzes whole slide images and patient outcomes to better understand, predict and evaluate the prognosis of patients with lung (Yu, Lee, et al. 2020), renal (Marostica et al. 2021), breast (Ektefaie et al. 2021), or ovarian (Yu, Hu, et al. 2020) cancer. Whole slide images pose many challenges for their studies. First, they require expensive scanners that produce images with multiple gigabytes in size. This makes their processing difficult, as one needs to extract insights from the whole slide without losing the details at higher magnification. Second, they involve a costly and tedious acquisition process that limits the availability of datasets for some diseases. Moreover, histopathology slides are stained with hematoxylin and eosin, which affects their coloration depending on the staining procedure.

The study of histopathology slides reveals many difficulties in Medical Machine Learning. Datasets are scarce and hard to label for some diseases. In this chapter, we will introduce the datasets used in this thesis and the problems they present. The datasets in this chapter are relevant to all the chapters of this thesis, as the techniques we present are useful for understanding histopathology slides. Specifically, we will focus on the Glioblastoma dataset Ivy Gap (Puchalski et al. 2018, IvyGap), and a dataset of renal cancer slides captured by a cell-phone.

1.2 Glioblastoma Dataset: IvyGAP

Glioblastoma is a type of cancer that is known to be one of the most aggressive. Beginning in the brain is the most lethal malignant brain tumor. It represents 15% of all brain tumors and even for patients receiving aggressive treatment, the median survival is 12-15 months. It is estimated that more than 200'000 individuals in the world will succumb to glioblastoma every year. The five-year survival rate for glioblastoma patients is only 6.8%, and there are no known methods to prevent glioblastoma (*Glioblastoma* n.d.).

In 2019, the Allen Institute for Brain Science, and the Ben and Catherine Ivy Center for Advanced Brain Tumor Treatment have introduced the Ivy Glioblastoma Atlas (IvyGAP), a publicly available atlas of Glioblastoma patients, their segmented whole slide tumor samples and their clinical data. The goal of the IvyGap project is to provide scientists and physicians with online resources dedicated to the understanding of Glioblastoma. Overall, the Ivy GAP cohort consisted of 41 patients (table S1), and their 42 tumors were used to generate 440 tissue blocks, 270 transcriptomes, 11,500 machine learning (ML)-annotated H&E images registered to 23,000 ISH images, 400 MRI scans, and supporting longitudinal clinical information (Puchalski et al. 2018).

In this report we will focus on the set of WSI and their corresponding segmentation map, in particular, we will try to identify the insights that this dataset can provide on Glioblastoma while trying to solve some of the challenges that this dataset presents.

The original images were obtained using a ScanScope scanner and directly saved as SVS images, each of them being approximately 5 GB. The images and their segmentation map were then converted to JPEG 2000 file format and compressed at a rate of 0.8 to 400 MB per

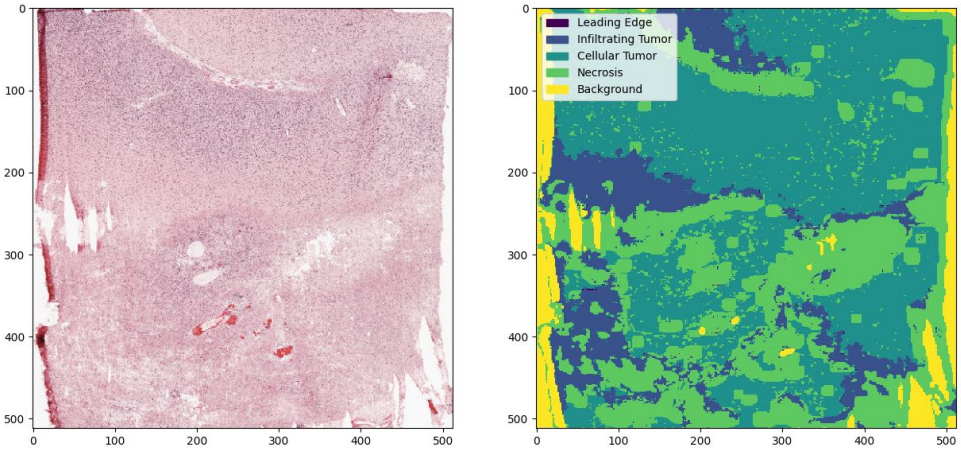
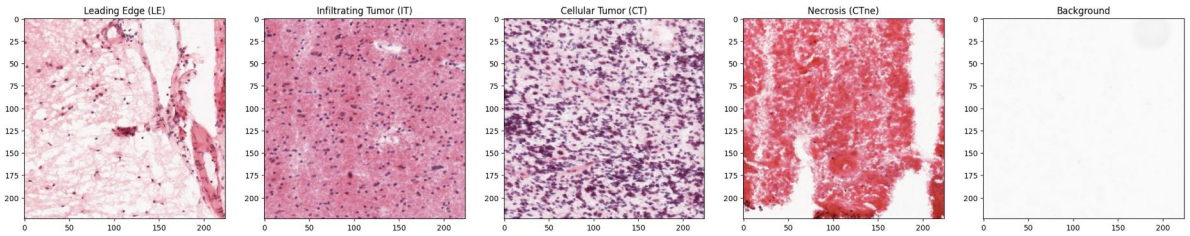


Figure 1.1: Example of WSI from the IvyGAP dataset (left) and the corresponding segmentation ground truth (right). The segmentation map was obtained in a semi-automatic manner.

image. The segmentation maps were obtained using Decision Forests to detect the following features:

- Leading Edge (LE): The outermost boundary of the tumor, where the ratio of tumor to normal cells is about 1-3/100,
- Infiltrating Tumor (IT): It is the intermediate zone between the Leading Edge (LE) and Cellular Tumor.
- Cellular Tumor (CT): Tissue where the ratio of tumor cells to normal cells is about 100/1 to 500/1.
- Necrosis (CTne): Dead or dying tissue, marked by the absence of crisp cytological architecture.
- Background (Background): Background of the slide.



(a) Leading Edge **(b)** Infil. Tumor **(c)** Cellular Tumor **(d)** Necrosis **(e)** Background

Figure 1.2: 5 classes in the subsample IvyGAP dataset

The distribution of the 5 classes of IvyGAP is detailed bellow in table 1.2.

Glioblastoma is a highly aggressive and lethal brain tumor that affects millions of people worldwide. Understanding the molecular and cellular mechanisms of glioblastoma development and progression is crucial for developing effective therapies and improving patient outcomes. However, glioblastoma is a heterogeneous disease that exhibits spatial and temporal variations in gene expression, histology, and response to treatment. Therefore, a comprehensive atlas of glioblastoma that integrates multiple levels of data across different regions of the tumor is needed to reveal its complexity and diversity.

Class	Train	Val	Test
Leading Edge	8.08%	11.1%	1.88%
Infiltrating Tumor	12.1%	7.20%	15.9%
Cellular Tumor	49.5%	39.6%	51.5%
Necrosis	21.9%	34.7%	22.9%
Background	8.15%	7.41%	7.62%

Table 1.1: Distribution of the 5 classes in the IvyGAP dataset

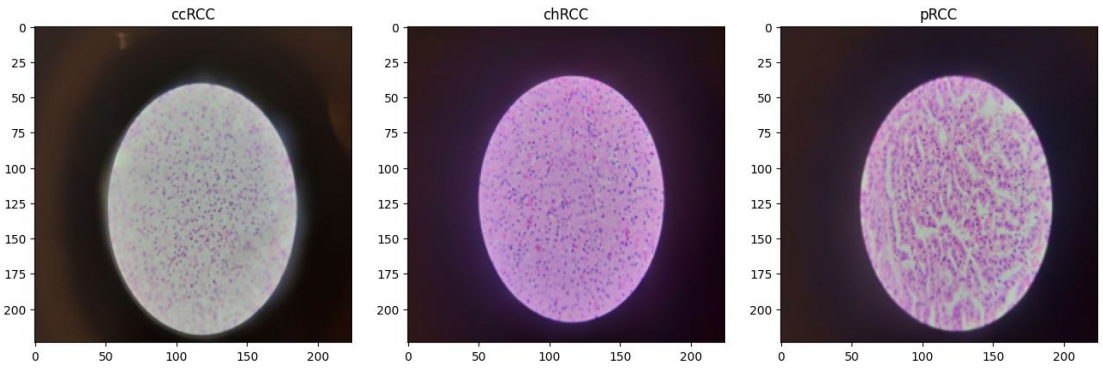
The Ivy Glioblastoma Atlas Project (IvyGAP) is an ambitious initiative that aims to provide such an atlas by combining high-resolution histological images, gene expression data from in situ hybridization (ISH) and RNA sequencing (RNA-Seq), and clinical and genomic information for each donor and tumor. IvyGAP is a valuable resource for researchers, clinicians, and patients who want to explore the anatomic and genomic basis of glioblastoma. By allowing users to visualize and compare gene expression patterns across different regions of the tumor, IvyGAP can help identify novel biomarkers, therapeutic targets, and molecular subtypes of glioblastoma.

However, IvyGAP also poses some challenges for data analysis and interpretation. One of them is the potential presence of mislabeled data due to the semi-automated annotation process that was used to assign gene expression values to different anatomical structures. The annotation process involved using a Random Forest classifier trained on expert-labeled samples to predict the labels for unlabeled samples based on their ISH intensity values. However, this approach may introduce errors due to noise, outliers, or inconsistencies in the training data or the classifier parameters. Therefore, it is important to assess the quality and reliability of the IvyGAP dataset before using it for downstream applications.

1.3 Renal Cancer dataset from a cell-phone

Renal cell carcinoma (RCC) is the most prevalent type of kidney cancer, accounting for 2-3% of cancers in humans. RCC can be divided into three main subtypes based on cell appearance: clear cell RCC (ccRCC), papillary RCC (pRCC), and chromophobe RCC (chRCC). These three subtypes make up more than 90% of all RCC cases. Clear cell RCC is the most dangerous, while papillary and chromophobe RCCs have higher survival rates. The classification of RCC subtypes is important in clinical practice due to the growing use of innovative treatments, which necessitates new methods for identifying subtypes. RCC is associated with several risk factors, such as smoking, certain pain medications, previous bladder cancer, obesity, high blood pressure, exposure to certain chemicals, and a family history of the disease. Treatment options for RCC and TCC include surgery, radiation therapy, chemotherapy, immunotherapy, and targeted therapy. In 2018, approximately 403,300 new cases of kidney cancer were reported worldwide, resulting in 175,000 deaths. The dataset studied at the Yu Lab was directly taken by members of the lab, who manually scanned 4694 slides.

Microscopes are powerful tools that can reveal the microscopic structures and processes of various biological and physical phenomena. However, conventional microscopes are often expensive, bulky and require specialized training and maintenance. Therefore, studying a dataset of pictures taken from a cellphone in a microscope can have various motivations depending on the research question and the domain of application.



(a) Clear cell RCC **(b)** Chromophobe RCC **(c)** Papillary RCC

Figure 1.3: Renal Cell Carcinoma

For example, some researchers may want to study how cellphone microscopes can provide low-cost, portable and accessible alternatives to conventional microscopes for various purposes such as education, diagnosis and research. Others may want to study how cellphone microscopes can enable different contrast mechanisms such as bright-field, dark-field and fluorescence imaging that can enhance the visualization of different features in the specimens1. Yet others may want to study how cellphone microscopes can be integrated with machine learning techniques such as image processing, segmentation and classification that can automate the analysis of the images and provide quantitative measurements.

By studying a dataset of pictures taken from a cellphone in a microscope using machine learning techniques such as convolutional neural networks, edge detection and feature extraction, researchers can exploit the potential of cellphone microscopes for various scientific and practical applications that can benefit society.

Class	Train	Val	Test
ccRCC	25.05%	24.78%	23.32%
chRCC	32.29%	33.16%	32.69%
pRCC	42.65%	42.04%	43.98%

Table 1.2: Distribution of the 3 classes in the Renal Cancer dataset

2 | Self-Supervised Learning for Transformers pre-training on unlabeled dataset

2.1 Introduction

Ever since their introduction, Deep Neural Networks have been shown to be powerful tool in solving Computer Vision tasks, such as image classification, object detection, semantic segmentation and many more. Their impressive performance and flexibility has allowed them to be used in many applications, where more traditional methods would have failed in the past. However, the success of Deep Neural Networks (DNNs) is not only due to their ability to learn complex representations of the data, but also due to the large amount of data they are trained on. In fact, the performance of DNNs is highly dependent on the amount of data available for training; up to a certain point will plateau. This diminishing return, combined with the expensive cost of labeling data, has led to the development of self-supervised learning methods, which allow training DNNs on large unlabeled datasets.

Self-Supervised learning methods are based on the idea that the data itself contains enough information to learn a good representation of the data. This representation can then be used in a variety of downstream tasks, as long as the data is similar to the one used for pre-training. Medical datasets rise above in complexity and labeling cost, and present new challenges in term of patient privacy. This makes self-supervised learning methods a promising approach for medical image analysis, as they allow training DNNs on large unlabeled datasets, while preserving patient privacy.

In this Chapter, we will introduce Self-Supervised Learning methods and their application to medical image analysis. We will see how much pre-training methods can be used to improve the performance of the model on downstream tasks while also providing a better understanding of the dataset. We will compare the performance and costs of pre-training methods on large-scale medical slide datasets, and motivate the use of general pre-trained models on SSL algorithms for downstream tasks.

2.2 Vision Transformer

2.2.1 Introduction

Self-attention-based architectures, particularly Transformers Vaswani et al. 2017, have become the go-to model in natural language processing (NLP). The common approach is to first pre-train on a large corpus of text, and then fine-tune on a smaller, task-specific dataset. Thanks to the scalability and computational efficiency of Transformers, it is now possible to train models of unprecedented size. However, in computer vision, convolutional neural networks (CNNs) remain the dominant architecture. Recently, there have been attempts to combine CNNs with self-attention J. Chen et al. 2021, and some models even replace convolutions entirely. While these models have the potential for efficiency, they have yet to be effectively scaled on modern hardware accelerators due to the use of specialized attention patterns.

In this section, we will introduce the mechanisms of transformers and their recent generalization to computer vision (Dosovitskiy et al. 2021). We will detail the self-attention mechanism and their ability to form strong backbone for computer vision tasks. We will also introduce

the Vision Transformer (ViT) architecture, and its ability to scale to large datasets and large models.

This section will be particularly relevant for this chapter as it will be the backbone of the self-supervised learning methods, in the following chapter for the interpretability of its attention heads and the last chapter when estimating uncertainty of predictions.

2.2.2 Self-Attention using Scaled Dot-Product attention

The goal of Transformers in Shah et al. 2017 is to learn a representation of the input sequence that is invariant to the order of the elements in the sequence. This is achieved by using self-attention, which is a mechanism that allows a model to focus on different positions of the input sequence to compute a representation of that position. Self-attention mechanisms are particularly relevant in problems where sequential computation is expensive, like in natural language processing.

Self-attention, also known as intra-attention, is a mechanism used to analyze a single sequence by relating different positions within it. This method has been proven effective in various tasks such as comprehension, summarization, entailment and learning sentence representations that are independent of a specific task

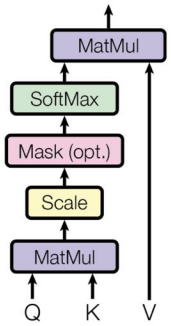


Figure 2.1: Scaled Dot-Product Attention

Mathematical Formulation

In the Scaled Dot-Product Attention mechanism, the *queries*, *keys*, and *values* are used to compute a representation of a sequence. The queries and keys are used to calculate the dot products, which represent the similarity or relevance between them. The values are then used to weight the importance of each element in the sequence based on the dot product similarities. Let Q be the matrix of queries, K be the matrix of keys, and V be the matrix of values. The scaled dot-product attention is defined as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

The dot product is a measure of similarity between two vectors. In the context of attention, the dot product is used to calculate the similarity between the queries and keys, which are used to determine the importance of the values. The dot product is calculated by taking the sum of the element-wise product of the queries and keys. The dot product is then scaled by $1/sqrt(d_k)$,

where d_k is the dimension of the queries and keys, to prevent the dot products from becoming too large and causing issues with the softmax function.

Intuition behind Self-Attention

In essence, the queries are used to determine what information is needed from the sequence, the keys are used to determine where that information is located in the sequence, and the values are used to provide that information. The dot product similarity between the queries and keys is used to weight the importance of the values, which allows the attention mechanism to focus on the most relevant parts of the sequence.

To better understand the self-attention mechanism, consider the following example. Consider the following sentence as an example: "The cat sat on the mat."

When passing this sentence through a self-attention head, each word in the sentence will be represented as a vector, and these vectors will be used as the input to the self-attention mechanism.

The queries, keys and values are all derived from the input word vectors. The queries represent the words we want to attend to, the keys represent the words that the attention mechanism will compare the queries to, and the values are the word vectors that will be used to construct the output representation.

For example, let's say we want to attend to the word "cat" in the sentence. The query vector for "cat" would be used to compare against the key vectors for all the other words in the sentence. The dot product between the query vector for "cat" and the key vectors for the other words would then be calculated. These dot products are then used as weights to weigh the value vectors of the other words. The resulting weighted sum of value vectors represents the output representation of "cat" in the sentence.

2.2.3 Multi-Head Attention

Multi-head attention is an extension of the self-attention mechanism in which multiple attention heads are used to compute different representations of the input sequence. Each head independently computes a weighted sum of the input values, where the weights are determined by a compatibility function of the input queries, keys, and values. These representations are then concatenated and projected to obtain the final output representation. The main intuition behind multi-head attention is that it allows the model to attend to different parts of the input sequence, with each head potentially learning a distinct type of attention. This allows the model to capture more fine-grained information and to attend to different parts of the input in a more nuanced way. Additionally, by having multiple heads, the model can learn a more diverse set of representations, which can be beneficial for downstream tasks. In Chapter 3, we will show how the DINO algorithms can provide segmentation-like map through the attention heads of the ViT backbone, how this can be used to find regions of interest in whole slide images and interpret model predictions.

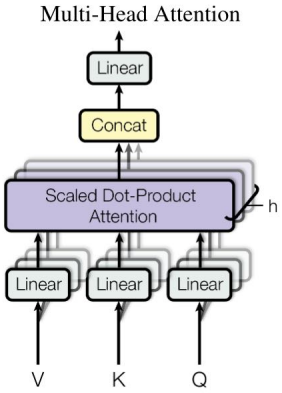


Figure 2.2: Multi-Head Attention

2.2.4 Vision Transformers (ViT): scaling-up to images

First introduced for NLP tasks, Transformers architecture have rapidly shown strong performance and grown in popularity compared to sequential models. The Self-Attention mechanism was motivated by the need to express intra-relevancy within a sentence, while also having the capacity to scale-up to larger set of tokens. Because of Transformers performance and scalability, larger models models are now trainable at scale and there is still no sign of saturating performance.

However, applying self-attention to images poses a challenge as it requires each pixel to attend to all other pixels, resulting in a cost that is quadratic to the number of pixels and is infeasible for realistic input sizes. To address this, Dosovitskiy et al. 2021 introduced minimal modifications to the original Transformers for NLP, and they treated an image as a sequence of patches. Each patch is then considered as a token, turned into an embedding and fed into the Transformer. The authors showed that this approach is able to scale-up to images of 16x16 pixels, and that it can be trained end-to-end for supervised classification.

Architecture of ViT

The traditional transformer for NLP receives a 1D sequence of tokens embedding. To process 2D images, the authors divide the original image $x \in R^{H \times W \times C}$ into a sequence of flattened 2D patches $x_p \in R^{N \times (P^2 \cdot C)}$ where each patch is of size $P \times P$, C is the number of channels and then $N = HW/P^2$.

In comparison to convolutional neural networks (CNNs), Vision Transformer (ViT) exhibits a lower level of image-specific inductive bias. While CNNs rely heavily on assumptions of locality, two-dimensional neighborhood structure, and translation equivariance, which are hard-coded into each layer throughout the entire model, ViT employs a more flexible approach, where only the multilayer perceptron (MLP) layers maintain these assumptions, while the self-attention layers are able to learn global patterns in the image. The utilization of the two-dimensional neighborhood structure in ViT is limited to the initial stages of the model, where the image is divided into smaller patches, and during fine-tuning for images of varying resolutions. Additionally, the position embeddings in ViT do not contain any prior knowledge about the 2D positions of the patches, thus the model is required to learn the spatial relations from the data.

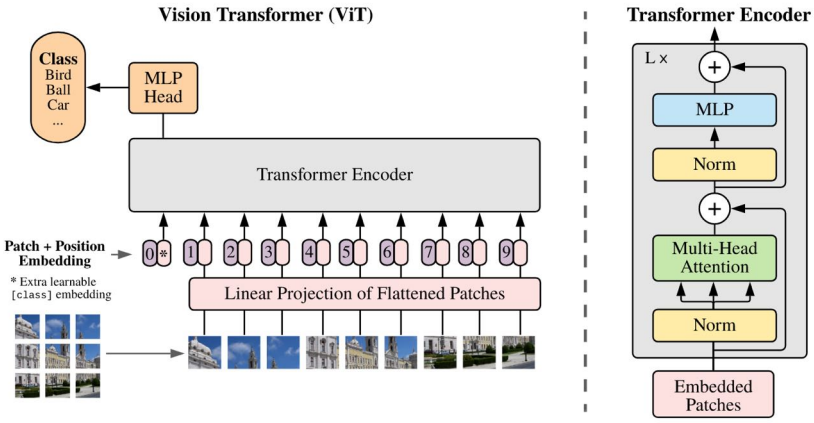


Figure 2.3: Vision Transformer architecture, where the input image is divided into a linear sequence of square patches. Each patch is then projected into a vector of fixed dimensionality. The sequence of patches is then fed into a Transformer encoder.

ViT achieves better performance than CNNs on benchmarks

In their study (ibid.), the largest models, ViT-H/14 and ViT-L/16, are compared to state-of-the-art CNNs from the literature. The first point of comparison is Big Transfer (BiT) (Kolesnikov et al., 2020), which utilizes supervised transfer learning with large ResNets. The second point of comparison is Noisy Student (Xie et al., 2020), a large EfficientNet trained using semi-supervised learning on ImageNet and JFT300M with the labels removed. At that time, Noisy Student holds the state of the art on ImageNet and BiT-L on other datasets reported. All models were trained on TPUv3 hardware. The smaller ViT-L/16 model pre-trained on JFT-300M outperforms BiT-L (which is pre-trained on the same dataset) on all tasks, while requiring substantially less computational resources. The larger model, ViT-H/14, further improves performance, particularly on more challenging datasets such as ImageNet, CIFAR-100, and the VTAB suite. It is worth noting that this model still took substantially less compute to pre-train than prior state of the art, although pre-training efficiency may be affected by multiple parameters, such as training schedule, optimizer, weight decay, etc. Lastly, the ViT-L/16 model pre-trained on the public ImageNet-21k dataset performs well on most datasets and requires fewer resources to pre-train; it could be trained using a standard cloud TPUv3 with 8 cores in approximately 30 days.

	ViT-H/14 (ViT-H/14)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55 ± 0.04**	87.54 ± 0.02	88.4/88.5
ImageNet ReaL	**90.72 ± 0.05**	90.54	90.55
CIFAR-10	**99.50 ± 0.06**	99.37 ± 0.06	-
CIFAR-100	**94.55 ± 0.04**	93.51 ± 0.08	-
TPUv3-core-days	2.5k	9.9k	12.3k

Table 2.1: Comparison of ViT with other models on ImageNet and CIFAR-10 benchmarks. ViT achieves higher accuracy on benchmark datasets while requiring fewer resources to pre-train.

2.3 Self-Supervised Learning

2.3.1 Introduction

Supervised methods may be the most common when we refer to Machine Learning models. These methods make use of labeled data to learn and predict on future unlabeled data. They provide the ability to learn rich patterns provided a large amount of labeled data, and has made deep neural networks a compelling approach in the majority of computer vision and natural language processing tasks.

Self-Supervised Learning (SSL) is a Machine Learning paradigm that aims to learn from unlabeled data. Self-supervised learning is a machine learning approach that involves training a model to perform a task using only unlabeled data. In this type of learning, the model is given a set of inputs and must predict some properties of the inputs, such as the missing pieces in a jigsaw puzzle or the rotation angle of an image. This is in contrast to supervised learning, where the model is given both input data and corresponding labeled outputs, and the goal is to learn a mapping from the inputs to the outputs.

One advantage of self-supervised learning is that it can be used to learn useful representations of the data even when labeled data is scarce. This is because the model is able to learn from the inherent structure of the data itself, rather than relying on explicit labels provided by a human annotator. This can make self-supervised learning particularly useful for tasks where it is difficult or expensive to obtain large amounts of labeled training data.

Self-supervised learning has been applied to a variety of tasks in natural language processing and computer vision, including language translation, image classification, and object detection. It has also been used to pre-train models that are then fine-tuned on a supervised task, resulting in improved performance compared to training on the supervised task alone.

In this section, we will introduce the main self-supervised learning methods, and provide a detail explanation of DINO, a form of knowledge distillation that can provide great pre-trained models.

2.3.2 Motivation and General Approach

Self-supervised learning aims to learn visual features from unlabeled data by using pretext tasks, which are designed to be solvable by neural networks. These tasks allow for the training of networks without the need for expensive and time-consuming human annotations. Common examples of pretext tasks include colorizing grayscale images, image inpainting, and image jigsaw puzzles (Zhang et al. 2016). These tasks share the property of capturing visual features through the weights of the neural networks and generating pseudo labels based on the attributes of the data.

In this process, a pretext task is assigned to a neural network, usually an encoder, which is trained to solve it. The task's pseudo labels are generated automatically from the data's attributes. Once the self-supervised training is complete, the visual features learned can be transferred to downstream tasks, particularly in situations where there is limited data available, to enhance performance and mitigate overfitting. Typically, shallow layers capture general low-level features like edges, corners, and textures while deeper layers capture task-specific high-level features. As a result, only the visual features from the first few layers are transferred during the supervised downstream task training phase.

In the context of self-supervised learning (SSL), a downstream task refers to a task that uses the features learned through the SSL process as input. These tasks can be any type of machine learning task that requires visual features, such as object detection, image classification, or segmentation. In medical tasks, downstream tasks could include tasks like disease classification, lesion segmentation, or anatomical landmark detection. These tasks often require a large amount of annotated data, which can be expensive and time-consuming to acquire. By using the visual features learned through SSL, the performance of these tasks can be improved and the need for large amounts of annotated data can be reduced. Additionally, the learned features can be used to pre-train models for medical tasks, which can help to overcome overfitting and improve performance on small datasets.

Because no human annotations are required for self-supervised learning, it can be used to train models on large amounts of unlabeled data. This can be particularly useful in medical applications, where it can be difficult or expensive to obtain large amounts of labeled data. For example, in the context of medical image analysis, self-supervised learning can be used to train models on large amounts of unlabeled medical images, which can then be used to improve the performance of downstream tasks that require labeled data.

2.3.3 Mathematical Formulation

Neural Networks can learn visual features from labeled or unlabeled data. Both approach have their advantages and disadvantages. In the case of labeled data, the network can learn the visual features that are relevant to the task, but it requires a large amount of labeled data. In the case of unlabeled data, the network can learn visual features from the data itself, but it requires a large amount of unlabeled data. In both cases, the networks have to be trained through a loss minimization process, which is usually done by backpropagation. Hence we can formulate the training process for Supervised and Self-Supervised Learning.

Supervised Learning Formulation

Supervised Learning involves the use of dataset X where each data point x_i has a corresponding human annotated label y_i. The goal of supervised learning is to learn a function f that maps the data x_i to the label y_i. The function f is usually a neural network, and the training process is done by minimizing the loss function L. Let D the dataset of size N, where $D = \{(x_i, y_i)\}_{i=1}^{N}$. The loss value of a neural network over a dataset is defined as:

$$L(D) = \sum_{i=1}^{N} d(f(x_i), y_i) \tag{2.2}$$

The distance function $d(x_i, y_i)$ is usually defined as a distance that will differ depending of the problem (norm L2, norm L1 etc.). The training process is done by minimizing the loss function L through backpropagation.

Self-Supervised Learning Formulation

Recently, several self-supervised learning methods for visual feature learning have been developed without the use of human-annotated labels (Mahendran et al. 2018, Sayed et al. 2019). These methods are also referred to as unsupervised learning. Unlike supervised learning methods, which require a data pair of (x_i, y_i), self-supervised learning is trained with data x_i and its **pseudo**

label P_i, which is automatically generated for a pre-defined pretext task without the need for human annotation. The pseudo label Pi can be generated using the attributes of images or videos such as the context of the images (Zhang et al. 2016) or the temporal information of videos.

Similarly to supervised learning, the goal of self-supervised learning is to learn a function f that maps the data x_i to the pseudo label P_i. The function f is usually a neural network, and the training process is done by minimizing the loss function L. Let D the dataset of size N, where $D = \{(x_i, P_i)\}_{i=1}^{N}$. The loss value of a neural network over a dataset is defined as:

$$L(D) = \sum_{i=1}^{N} d(f(x_i), P_i) \tag{2.3}$$

By training the model on pseudo labels, the network can still learn visual features even though the original dataset was unlabeled. In the process, the network can learn to solve useful tasks, such as image completion, image coloring, or image jigsaw puzzles. The learned visual features can then be transferred to downstream tasks, such as image classification, object detection, or segmentation.

SSL methods can be divided into two categories: contrastive learning and knowledge distillation. In the following sections, we will introduce the main contrastive learning methods and explain the knowledge distillation method DINO.

2.3.4 Learning Image Features

In order to learn image features, neural networks are trained on various pretext tasks, which can be divided into divided intro three categories: generative-based, context-based and semantic free (Jing et al. 2019). Generative-based methods aim to generate images from the input image, while context-based methods aim to predict the context of the input image. Semantic free methods aim to predict the attributes of the input image. In the following sections, we will introduce the main contrastive learning methods and explain the knowledge distillation method DINO.

Generative-Based Feature Learning

Generation-based self-supervised methods for learning image features involve the process of generating images, such as using GANs to generate fake images, super-resolution to generate high-resolution images, image inpainting to predict missing image regions, and image colorization to colorize grayscale images. These tasks do not require human-annotated labels and use the images themselves as pseudo training labels P. Therefore, these methods are considered self-supervised. The Autoencoder in Pathak et al. 2016 was the pioneer work in this area, which learns to compress an image into a low-dimensional vector, which can then be uncompressed back into an image that is similar to the original. Current image generation-based methods follow a similar idea but with different pipelines to learn visual features through the process of image generation.

One specific type of image generation-based method is the use of Generative Adversarial Networks (GANs). GANs consist of two networks, a generator and a discriminator. The generator generates images from latent vectors, while the discriminator tries to distinguish between generated

Figure 2.4: Automatic Inpainting use autoencoders to predict the missing regions of an image, trained with L2 reconstruction loss (Pathak et al. 2016)

and real images. The two networks compete against each other during training, with the discriminator trying to improve its ability to differentiate and the generator trying to produce more realistic images. This competition leads to the improvement of both networks.

In addition to image generation with GANs, there are other types of generative-based feature learning methods such as image generation with super resolution and image generation with colorization.

Super-resolution is a technique that aims to increase the resolution of an image by generating a high-resolution version of it. This can be done by training a model to learn the mapping between low-resolution and high-resolution images, and then using this model to generate high-resolution versions of new low-resolution images. In Ledig et al. 2017, the authors proposed SRGAN, a GAN network for generative a super-resolution image. This approach can be used as a self-supervised feature learning method, where the high-resolution image is used as the pseudo label for the low-resolution image.

Image colorization is another example of generative self-supervised feature learning, it consists of adding colors to grayscale images. The model learns to predict the colors of an image based on its grayscale version and the task can be treated as a self-supervised task where the colorized image is used as the pseudo label for the grayscale image. In recent years, several deep learning-based methods for image colorization have been proposed. A common approach is to use a fully convolutional neural network (CNN) that consists of an encoder for feature extraction and a decoder for color hallucination. The network can be optimized using the L2 loss between the predicted color and the original color. An alternative approach was proposed by Zhang et al. 2016. where they treated the task as a classification task and used class-rebalancing to increase the diversity of predicted colors. When trained on large-scale image collections, this method achieved impressive results and was able to fool human evaluators in 32% of the trials during the colorization test.

Semantic Free Feature Learning

Free semantic labels are labels that have semantic meaning and are generated without human involvement. These types of labels, such as segmentation masks, depth images, optic flows, and surface normal images, can be produced using game engines or hard-coded methods. As they are

automatically generated, the use of synthetic datasets or in combination with large unlabeled image or video datasets is considered as a self-supervised learning approach.

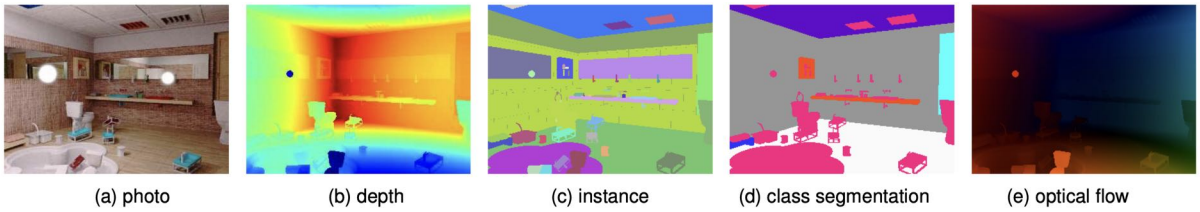


Figure 2.5: An example of an indoor scene generated by a game engine and its corresponding use cases. This approach can lead to a large amount of high quality segmentation maps or depth estimation data. (Shah et al. 2017)

Game engines, such as Airsim (Shah et al. 2017), can produce realistic images and corresponding pixel-level labels by modeling various objects and environments. These engines can generate large-scale datasets with minimal cost and have been utilized to create synthetic datasets with high-level semantic labels, including depth, contours, surface normal, segmentation mask, and optical flow, for training deep learning networks. An example of an indoor scene generated by a game engine and its corresponding labels is shown in Figure 18. However, due to the difference between synthetic and real-world images, it is challenging to directly apply a neural network trained on synthetic images to real-world images. To overcome this, the domain gap must be explicitly addressed when utilizing synthetic datasets for self-supervised feature learning. Through this method, a neural network trained on the semantic labels of synthetic datasets can be effectively applied to real-world images

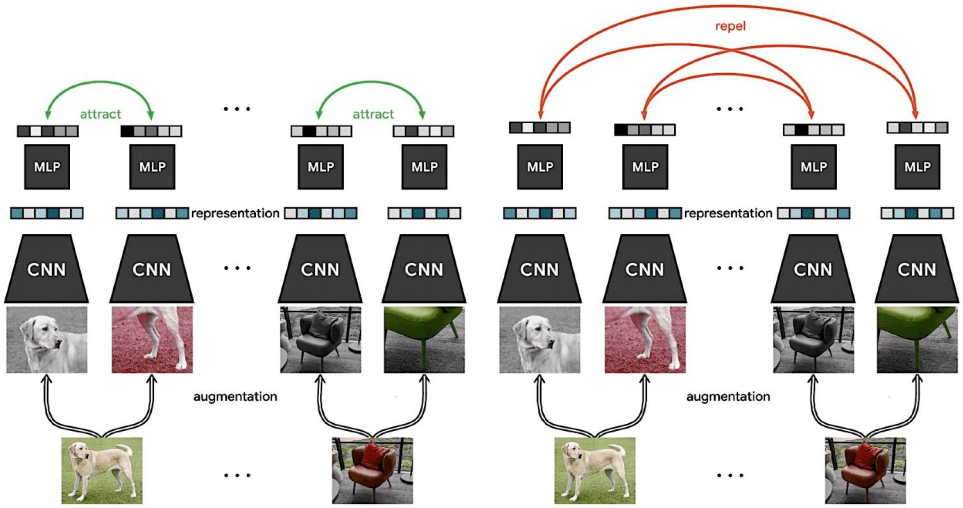
Context-Based Feature Learning

Context-based pretext tasks utilize the context features of images, such as context similarity, spatial structure, and temporal structure, as the supervision signal. Neural networks learn features by solving pretext tasks that are designed based on the attributes of the context of images. Context-based pretext tasks for self-supervised learning utilize the rich spatial context information contained in images, such as the relative positions of different patches within an image. These tasks can include predicting the relative positions of two patches from the same image, recognizing the order of a shuffled sequence of patches from the same image, or recognizing the rotation angle of an entire image. To accomplish these pretext tasks, Neural Networks must learn spatial context information such as the shape of objects and the relative positions of different parts of an object.

One example of this approach is the method proposed by Doersch et al. 2016, which involves training a Convolutional Neural Network to recognize the relative positions of random pairs of image patches extracted from an image. Other methods have been proposed that involve solving more difficult spatial puzzles, such as the image Jigsaw puzzle. In this task, an image is divided into several patches, which are then shuffled and fed to the network, which is trained to recognize the correct spatial locations of the input patches by learning the spatial context structures of images such as object color, structure, and high-level semantic information. The main principle of designing puzzle tasks is to find a suitable task that is not too difficult or too easy for a network to solve, as a task that is too difficult may not converge or a task that is too easy may lead to trivial solutions.

Another approach to context-based feature learning is the contrastive learning approach, which trains the model to predict if two inputs are from the same source or not. As an example,

SimCLR T. Chen et al. 2020 uses a contrastive learning approach to train a neural network to predict if two randomly cropped patches from the same image are from the same source or not. This approach can be used to learn features that are invariant to image transformations such as color, rotation, and translation.



(a) Attract embeddings from the same input image

(b) Repel embeddings from different input images

Figure 2.6: SimCLR (T. Chen et al. 2020) employs contrastive learning to learn image representation without labels. By positively associating images that come from the same input, and repelling images that come from different ones, SimCLR is able to learn contextual and global knowledge of the images.

In this report, we will focus on the DINO, a recent approach to Self-Supervised Learning on Vision Transformers which uses context-based feature learning and knowledge distillation to teach a student neural network to learn features from a representation of unlabeled data. This approach will come to play in the following chapters as a strong approach pre-training for downstream tasks, and finding regions of interest without labels.

2.4 DINO: Self-Distillation with no labels

In *Emerging Properties in Self-Supervised Vision Transformers* Caron et al. 2021, the Facebook AI team has shown the potential of self-supervised pretraining on Visiton Transformers model (ViT) and proposed the DINO algorithm. This approach to self-supervised pretraining produces models achieving performance that are comparable to the best Convolutional Neural Networks, while also providing insights into the image embeddings produced. The motivation is that self-supervised pretraining, in the form of the masked language modeling objective in BERT or language modeling in GPT, played a crucial role in the success of Transformers in natural language processing (NLP). These self-supervised pretraining objectives use the words in a sentence to create pretext tasks that provide a more comprehensive learning signal than the supervised objective of predicting a single label per sentence.

In this section we will provide a deep down into the knowledge distillation approach of DINO, preview the attention maps it could provide (2.4.3) and present how models pretrained by DINO outperforms fully supervised models.

2.4.1 Student and Teacher networks learn global representations

The DINO framework is similar in structure to recent self-supervised approaches, but also incorporates elements of knowledge distillation. The overall structure of DINO is illustrated in Figure 2 and its pseudo-code implementation is provided in Algorithm 1.

Knowledge distillation is a technique where a student network, g_{θ_s}, is trained to mimic the output of a teacher network, g_{θ_t}, with parameters θ_s and θ_t respectively. Given an input image x, both networks output probability distributions over K dimensions, denoted by P_s and P_t. These distributions are obtained by normalizing the output of the network g with a softmax function. The *temperature* $\tau_s > 0$ controls the sharpness of the distribution, and is used to prevent the student network from overfitting to the teacher network.

$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^{K} \exp(g_{\theta_s}(x)^{(k)}/\tau_s)} \tag{2.4}$$

The student network is trained to match these distributions by minimizing the cross-entropy loss with respect to its parameters θ_s. This is done by minimizing the cross-entropy loss $H(P_t(x), P_s(x))$.

$$\min_{\theta_s} H(P_t(x), P_s(x)) = \min_{\theta_s} -P_t(x) \log P_s(x) \tag{2.5}$$

To begin, DINO generates a set of various distorted views of an image, known as crops. Specifically, from a given image, the algorithm creates a set V of different views, which includes two global views, as well as multiple local views of smaller resolution. All of the crops are processed by the student network, but only the global views are processed by the teacher network. This encourages correspondence between local and global features.

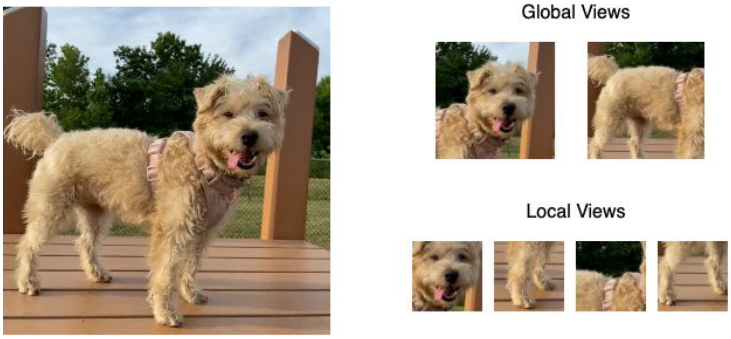


Figure 2.7: The student network is trained to predict the global view of the teacher network.

2.4.2 Student and Teacher networks share architecture and knowledge

The student and teacher networks architecture are identical and are both initialized with the same parameters. The student network is trained to match the teacher network, and the teacher network is updated to match the student network. This is done by using a momentum update rule, where the teacher network is updated with a rate l and the student network is updated with a rate $1-l$. This is done to ensure that the student network does not overfit to the teacher network (collapse) and that the teacher network does not overfit to the student network (diverge). The update rule for the teacher network is given by:

$$\theta_t \leftarrow l\theta_t + (1-l)\theta_s \tag{2.6}$$

with l being the momentum rate, and following a cosine schedule from 0.996 to 1 during training.

Another novel technique in order to avoid teacher collapse is the use of a centering technique. Centering the prediction of the teacher network g_t before the softmax function is applied is done by subtracting the mean of the teacher network's output from the teacher network's output. This is done by maintaining a running average of the teacher network's output, and subtracting this average from the teacher network's output. The update rule for the center is given by:

$$C \leftarrow mC + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_t(x)^{(i)} \tag{2.7}$$

where $m > 0$ is a rate parameter and B is the batch size.

Algorithm 1 DINO PyTorch pseudocode w/o multi-crop

```
gs, gt ← student and teacher networks
C ← center (K)
tps, tpt ← student and teacher temperatures
l, m ← network and center momentum rates
gt.params ← gs.params
for x in loader do
    x1, x2 ← augment(x), augment(x)
    t1, t2 ← gt(x1), gt(x2)
    loss ← 1/2(H(t1, s2) + H(t2, s1))
    loss.backward()
    update(gs)
    gt.params ← l*gt.params + (1-l)*gs.params
    C ← m*C + (1-m)*cat([t1, t2]).mean(dim=0)
end for

function H(t, s)
    t ← t.detach()
    s ← softmax(s / tps, dim=1)
    t ← softmax((t - C) / tpt, dim=1)
    return - (t * log(s)).sum(dim=1).mean()
end function
```

2.4.3 Transformers pretrained with DINO learn attention maps

The DINO framework was built with Transformers in mind, as an answer to similar techniques applied to Natural Language Processing. The writers of DINO (Caron et al. 2021) have tested both ResNet and ViT architectures when evaluating the SSL algorithm. Among the properties of Vision Transformers is the ability to learn attention maps, by evaluating the different heads of the Transformer. For example, the architecture of ViT-8 small provides 6 attention heads that all should focus on different semantic regions of the image. The weights of the attentions

heads then could provide insights on different regions of the image, can be used to visualize the attention maps of the Transformer.

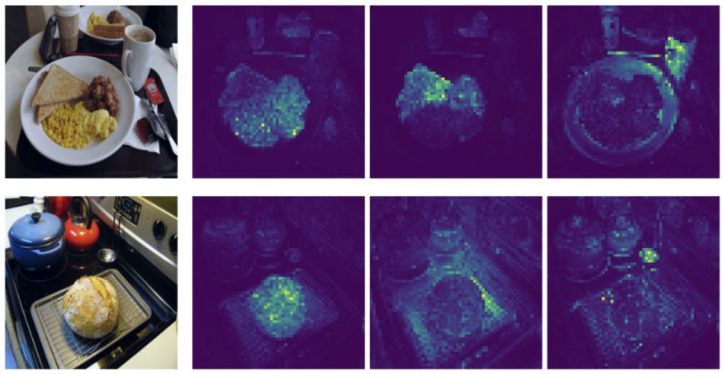


Figure 2.8: Attention maps of the ViT-8 small architecture.

This property of the ViT architecture will be particularly useful to the task of understanding the IvyGAP dataset and finding regions of interests, as it can allow us to find relevant and distinct regions of the whole slide images without the need of segmentation labels or supervised learning.

2.4.4 Transformers pretrained with DINO learns semantic representation for clustering

One of the properties of DINO highlighted by the authors is the quality of embeddings for classification, even using basic clustering methods like *k-NN*. With no finetuning, linear classifier or data augmentation, a ViT-8 model pretrained with DINO can achieve 78.3% accuracy on ImageNet.

To obtain these results, the authors have frozen the weights of the model and did not use the downstream head, in other words for $g = h \circ f$, h was not used. The nearest neighbor classifier is then trained on the frozen features of an image, by finding the most represented class among the k neighbors. The other advantage for such method is that one can obtain a model that produces quality features for clustering without using labels and using a different (and larger) dataset.

Method	Arch.	Param.	im/s	Linear	k-NN
Supervised	ViT-S	21	1007	79.8	79.8
BYOL*	ViT-S	21	1007	71.4	66.6
MoCov2*	ViT-S	21	1007	72.7	64.4
SwAV*	ViT-S	21	1007	73.5	66.3
DINO	ViT-S	21	1007	77.0	74.5

Table 2.2: Performance of different SSL methods on ImageNet. All methods excepted *Supervised* are trained using self-supervised learning, hence without any labels. We observe that DINO features outperforms the other methods on the ImageNet dataset, reaching 74.5% accuracy.

This property of models trained with DINO is particularly interesting for our task, as we can use the features of the model to cluster the images of the IvyGAP dataset. Furthermore it may

allow us to understand better the distribution of patches through a dataset and find confounding labels.

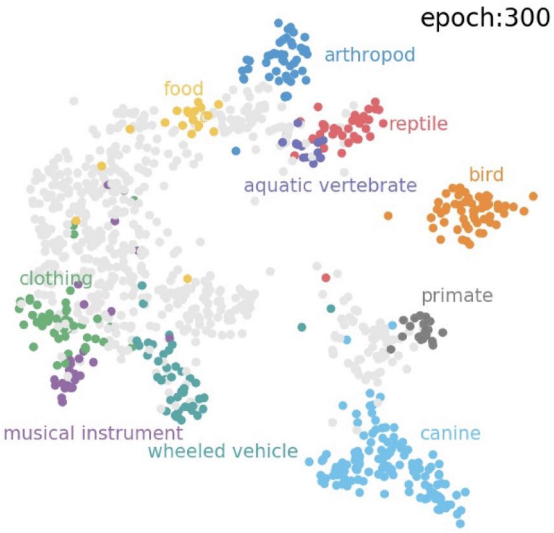


Figure 2.9: Clustering of the ImageNet dataset using the features of a ViT-8 model pretrained with DINO. The images are clustered using the k-NN classifier, and the clusters are then visualized using t-SNE.

2.4.5 Transformers pretrained with DINO can be leveraged for downstream tasks

The authors have evaluated the quality of the embedding and features learn with DINO by evaluating the obtained architecture on downstream task. Let f bet the ViT or ResNet backbone obtained after SSL pretraining, and h an arbitrary head for downstream task, then the model $g = h \circ f$ has been evaluated as a candidate for downstream tasks, against supervised models. In this report we will summarize the results obtained by Caron et al. 2021 on classification tasks across different datasets..

	Cifar10	Cifar100	INat18	INat19	Flwrs	Cars	INet
Supervised	**99.0**	89.5	70.7	76.6	98.2	92.1	79.9
DINO	**99.0**	**90.5**	**72.0**	**78.2**	**98.5**	**93.0**	**81.5**

Table 2.3: Accuracy of the VIT-S/16 model on different datasets, either supervised or pretrained with DINO. We see that the DINO model outperforms the supervised model on all classification datasets.

On ImageNet, a classification dataset of more than 14 millions images and containing 20'000 classes, the model pre-trained by DINO has observed an improvement of 1-2% compared to the baseline of a supervised model. We will evaluate this method on the IvyGAP dataset, and compare the results with the supervised model.

2.5 Self-Supervised pretraining of Vision Transformers for Medical Image Analysis

2.5.1 Introduction

Using the different components introduced in the previous sections, we will now evaluate the performance of the DINO framework on the IvyGAP dataset. In chapter 1, we have introduced the Glioblastoma dataset IvyGAP and the different challenges it poses, along with the Renal Cancer dataset taken from a smartphone. Then we have introduced Self-Supervised Learning as an approach for learning image features without the need for labels in section 2.3.1, and the mechanism of self-attention in Vision Transformers to learn features relationships within an image in section 2.2. Finally we have introduced the DINO framework has a modern approach for Self-Supervised Learning on Vision Transformers in section 2.4. In this section we will evaluate the performance of the DINO framework on the IvyGAP dataset, and compare it to the supervised model.

In this evaluation, we aim to demonstrate the effectiveness of the DINO framework in learning image features from the IvyGAP dataset, which is known to have a limited amount of annotated data. By comparing the performance of the DINO framework with a supervised model, we aim to show that self-supervised learning can be a viable alternative to traditional supervised learning in scenarios where labeled data is scarce. Furthermore, we will analyze the impact of the self-attention mechanism on the performance of the model and how it contributes to the learning of meaningful image features. This evaluation will provide insights into the potential of self-supervised learning and the DINO framework for medical image analysis and its potential applications in the field of computer vision.

2.5.2 Method

Model: Vision Transformers with Classification Head

Vision Transformers (ViT) (Dosovitskiy et al. 2021) are a recent advancement in the field of computer vision that utilize the transformer architecture to perform image classification tasks. The transformer architecture, originally developed for natural language processing, has shown remarkable success in various computer vision tasks such as image classification and object detection.

ViT replaces the traditional convolutional neural network (CNN) architecture used in computer vision with a fully-connected network based on the transformer architecture. This allows ViT to process entire images as sequences of tokens, rather than breaking them down into local patches. The self-attention mechanism in the transformer architecture enables ViT to learn global relationships between the tokens in the image and produce more powerful representations for image classification tasks.

One of the key innovations in ViT is the use of a patch embedding layer, which breaks the image into a fixed-size patch and converts each patch into a dense vector representation. These patch embeddings are then fed into the transformer encoder to capture the relationships between the patches. This allows ViT to process images of varying sizes and aspect ratios, which is a major advantage over traditional CNNs. In recent years, ViT has shown promising results on various benchmark datasets and has surpassed the performance of traditional CNNs in some cases. The ability of ViT to process entire images as sequences and capture global relationships between

image patches has made it a popular choice for computer vision tasks, and further research is being conducted to improve its performance and extend its applications.

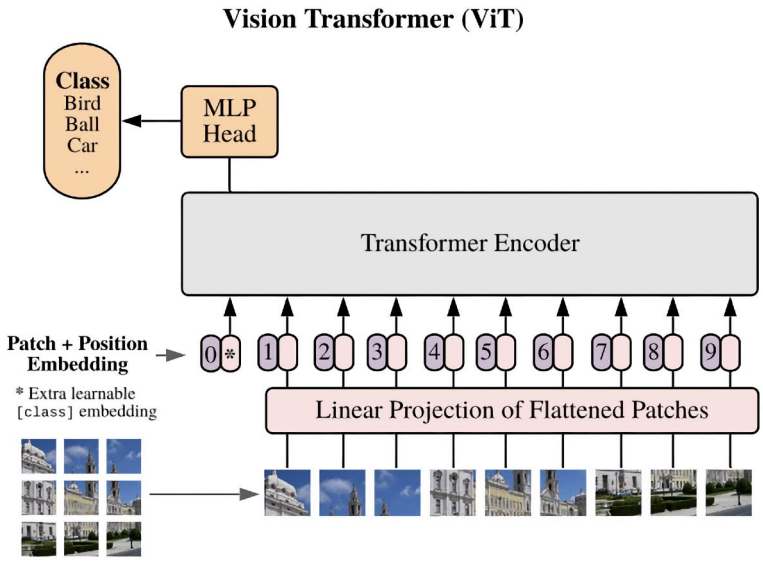


Figure 2.10: An example of a ViT network. The input image is broken down into patches, which are then converted into dense vectors using a patch embedding layer. These patch embeddings are then fed into the transformer encoder, which produces a sequence of tokens. The output of the transformer encoder is fed into the downstream encoder head, which performs the task-specific computation.

The downstream encoder head is the final layer in a ViT network that performs the task-specific computation, such as image classification. The output of the transformer encoder is fed into the downstream encoder head, which typically consists of a few fully-connected layers followed by a softmax activation function to produce a probability distribution over the classes in the dataset. The parameters of the downstream encoder head are typically trained end-to-end with the rest of the network, allowing it to learn task-specific representations that are optimized for the target task. The design of the downstream encoder head can have a significant impact on the performance of the ViT network. For example, the number of fully-connected layers and their size, as well as the activation function used, can all influence the final performance of the network. Thus, careful consideration of the downstream encoder head is important when designing a ViT network for a specific task.

Pretraining: DINO Self-Supervised Learning

Self-supervised learning is a cutting-edge approach in the field of machine learning that has gained significant attention in recent years (Jing et al. 2019). It refers to the process of training deep neural networks on large-scale unannotated data, where the model learns to perform tasks or solve problems without explicit human-provided labels. This method has proven to be highly effective in producing state-of-the-art results in various applications, particularly in computer vision and natural language processing. The ability to leverage unannotated data makes self-supervised learning particularly useful for training models on large-scale datasets, where annotating all the data may not be feasible. Furthermore, the models trained using this approach can be fine-tuned

for specific tasks using relatively small amounts of labeled data, making them ideal for use as pre-trained models.

One of the most promising applications of self-supervised learning is in the area of medical image analysis, where it has been used to improve the accuracy and efficiency of computer-aided diagnosis systems. The ability of self-supervised learning to learn high-level representations of medical images can be used to improve tasks such as classification, segmentation, and registration, leading to better patient outcomes.

The DINO algorithm Caron et al. 2021 is a powerful self-supervised learning approach that leverages the idea of distillation to train deep neural networks. The algorithm consists of two networks: a student network and a teacher network, both with the same architecture but different parameters. The input image is transformed in two ways and passed through both networks, with the output of the teacher network being used to guide the training of the student network. The two networks are trained to produce similar features, with the similarity being measured using a cross-entropy loss. The teacher parameters are updated using an exponential moving average of the student parameters, and the gradients are only propagated through the student network. The DINO algorithm has proven to be highly effective in producing high-quality representations of images, and it has been used in various applications, including computer vision and medical image analysis. In this section, we will leverage these image embeddings by adding a classification head to the pretrained model and evaluating its performance.

2.5.3 Experiment

In order to evaluate the effectiveness of the DINO framework and SSL pretraining, we will compare the performance of a supervised model with a self-supervised model trained using the DINO framework. The evaluation will be done on two datasets of interest for the Yu Lab: the IvyGap dataset for Glioblastoma patch classification, and the Renal Cancer dataset taken from a cell-phone.

We compare a Vision Transformers of patch size 8 (ViT-8) pretrained on a medical dataset (IvyGAP or Renal Cancer) using DINO SSL, to the same model but initialized from random weights. The pretraining was done over 300 epochs using no labels, a batch size of 8, a learning rate tuned between 0.0001 and 0.0005, a weight decay of 0.01, a teacher temperature tuned between 0.01 and 0.05, and a cosine annealing schedule. The downstream classification head was trained for 50 epochs using a batch size of 8, a learning rate of 0.0002, weight decay of 0.01, batch size 16 and a similar cosine scheduler. The experiment was done on Harvard O2 clusters using RTX 8000

Results

The results of the experiment are shown in the table 2.4. The results demonstrate the effectiveness of the DINO framework for self-supervised learning. The DINO pretrained model outperforms the supervised model in terms of accuracy on both the IvyGAP and RCD datasets. The test accuracy of the DINO pretrained model is 0.7339, which is a 6.27% improvement over the supervised model. The validation accuracy of the DINO pretrained model is 0.8229, which is a 3.17% improvement over the supervised model. These results demonstrate the ability of the DINO framework to learn high-quality representations of medical images, which can be used to improve the accuracy and efficiency of computer-aided diagnosis systems.

The improvements in accuracy can be attributed to the ability of the DINO framework to learn robust representations of medical images. The self-supervised training process allows the model to learn high-level features of the images without the need for explicit human-provided labels. This results in a more generalizable model that is able to handle a wider variety of images compared to a supervised model. Furthermore, the fine-tuning process using the downstream classification head allows the model to adapt to the specific task of medical image analysis.

Model	dataset	test acc	train acc	val acc
Supervised	IvyGAP	0.6712	0.8163	0.792
DINO Pretrained	IvyGAP	**0.7339**	**0.9615**	**0.8229**
Supervised	RCD	0.9255	0.9271	0.9531
DINO Pretrained	RCD	**0.9532**	**0.9739**	**0.9616**

Table 2.4: Results of the experiment. The DINO pretrained model outperforms the supervised model in terms of accuracy on both the IvyGAP and RCD datasets.

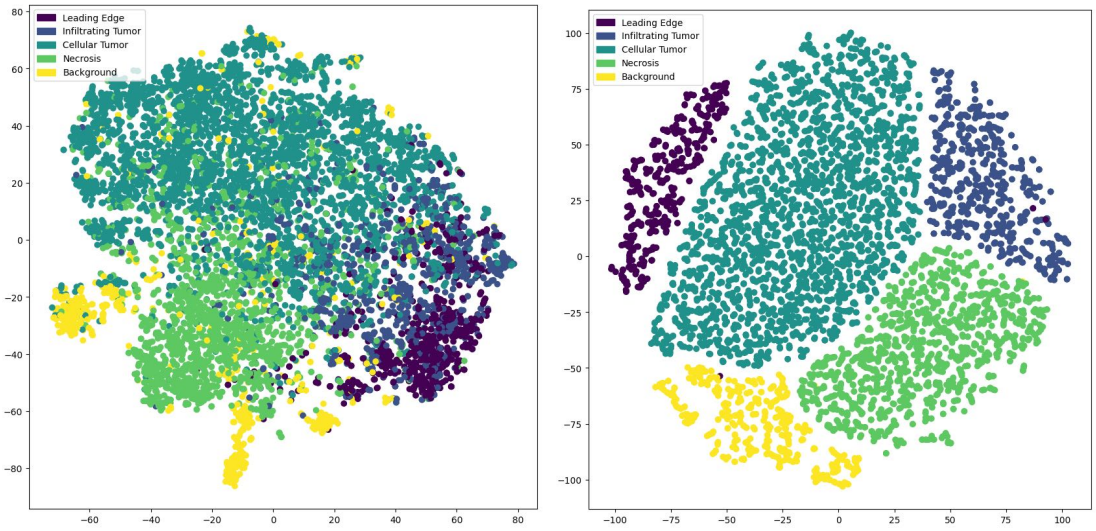
Embeddings visualization

As discussed in section 2.4.4, the DINO framework is able to train Vision Transformers to learn high-quality representations, than can be later used to cluster and visualize the space of images. Using the resulting model from the previous experiment in section 2.5.3, we can perform dimensionality reduction and T-Distributed Stochastic Neighbor Embedding (t-SNE) to visualize the embeddings of the images in the dataset. We will perform this visualization for both the model pre-trained with DINO (which has never seen the labels) and the supervised models. The results of the embeddings visualization will be compared to see how the representations learned by the DINO pretrained model differ from the supervised model. The embeddings of the supervised model are expected to be more compact and separated, as the self-supervised training process encourages the model to learn meaningful features.

The visualizations will be performed on a subset of the images from the IvyGAP and RCD datasets. The t-SNE visualization will provide a 2D projection of the embeddings, making it easier to see how similar images are clustered together

On figure 2.12, we can see that the embeddings of the supervised model are more compact and separated, while the embeddings of the DINO pretrained model are more spread out. This is expected, as the supervised model is trained to learn features that are useful for the classification task, while the DINO pretrained model is trained to learn features that are useful for the self-supervised task. The DINO pretrained model is able to learn more generalizable features, which can be used to cluster images that are similar in appearance but not necessarily in the same class.

We can try to understand the embeddings produced by DINO from subfigure 2.12(a), which did not use any labels during SSL training. First, we notice that Infiltrating Tumor embeddings are clustered between Infiltrating Tumor and Cellular Tumor and Leading Edge. This could be because of how these classes are defined. In fact, Leading Edge is the outermost boundary of the tumor, where there is one tumor cell for every 100 normal cells (1). Infiltrating Tumor is the zone between Leading Edge and Cellular Tumor. Finally, Cellular Tumor is where there are 100 to 500 tumor cells for every normal cell. The unclear definition of the class Infiltrating Tumor is evident as there are no clear boundaries between these classes. Moreover, we can see that



(a) ViT embeddings with DINO **(b)** ViT embeddings with DINO + finetuning

Figure 2.11: t-SNE visualization of the embeddings of the images in the IvyGAP dataset. The embeddings of the supervised model are more compact and separated, while the embeddings of the DINO pretrained model are more spread out.

Background and Necrosis are close together, which makes sense as later stages of Necrosis show the slide's background. The Background class also seems to have two clusters. This could be because some samples of the Background class are either completely empty or have some tissue at the edge.

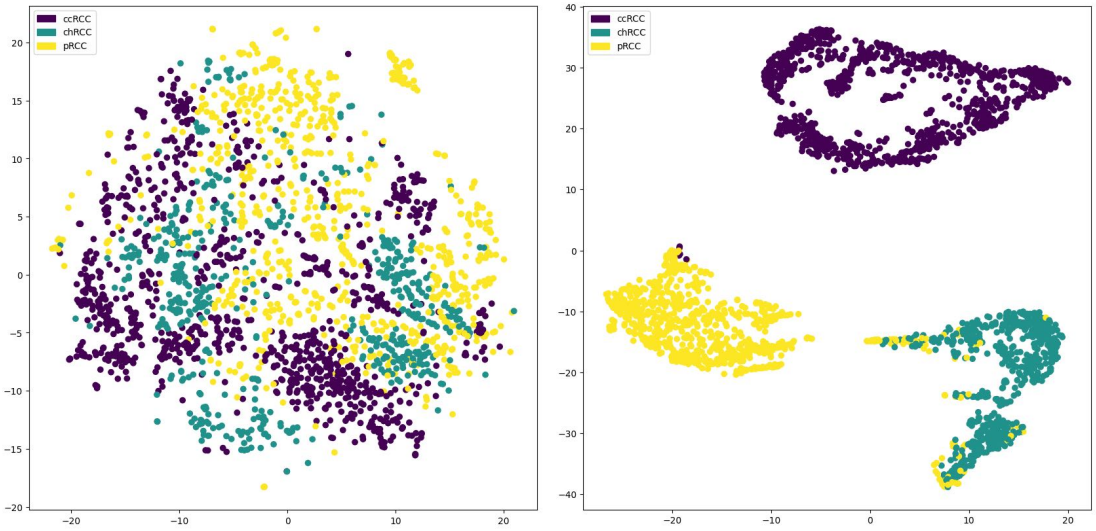
The clusters and boundaries are more evident in subfigure 2.12(b), as the DINO pretrained model was fine-tuned on a downstream task and thus knows the correct labels. We also seem to have lost the semantic meaning that was possible on subfigure 2.12(a), but we gained a more compact and separated embedding space. This is expected, as the DINO pretrained model is trained to learn features that are useful for the self-supervised task, while the supervised model is trained to learn features that are useful for the classification task. The DINO pretrained model can learn more general features, which can be used to cluster images that look similar, but are not necessarily in the same class.

For the Renal Cancer Dataset taken from a cellphone, the model trained with DINO SSL does not seem to provide embeddings as insightful as for IvyGAP. While points from pRCC, chRCC and ccRCC classes seem to be clustered together, there is no clear clustering and separation between classes. This may be explained by the fact that the dataset is taken from a cellphone, which has a lower resolution than the microscope used to take the IvyGAP dataset, and less obvious cues in the appropriate classe.

2.6 Conclusion

In this chapter, we introduced the model architecture that we will use in this thesis, the learning method and the pre-training algorithm. To conclude these sections, we conducted experiments to evaluate their combined performance on classification on IvyGap.

First, we presented Transformer Models and their ability to learn complex global interactions between tokens. We explained how this approach can be applied to images by splitting the input into patches, which describes the general architecture of Vision Transformers. Second,



(a) ViT embeddings with DINO **(b)** ViT embeddings with DINO + finetuning

Figure 2.12: t-SNE visualization of the embeddings of the images in the RCD dataset. The DINO pretrained model fails to provide embeddings as insightful as for IvyGAP.

we discussed Self-Supervised learning as a learning method on large unlabeled datasets. We demonstrated that models pretrained using SSL surpassed supervised models on downstream tasks, such as classification. This is especially important for medical machine learning tasks where data labeling can be costly. We examined an instance of such an algorithm, the DINO algorithm, which uses contrastive learning on Vision Transformers.

Finally, we applied this model to IvyGap and the Renal Cancer Dataset. We obtained semantic embeddings for inputs in IvyGap and the Renal Cancer Dataset. These embeddings helped us understand the possible labeling errors and challenges that we will address in the next chapters.

3 | Findings Regions of Interests in High Resolution Images

3.1 Introduction & Motivation

Whole slide images (WSIs) are digital scans of histopathology slides that can provide high-resolution and comprehensive information for diagnosis and research. However, WSIs also pose significant challenges for analysis due to their large size, complex structure, and heterogeneous content. Therefore, finding regions of interest (ROIs) in WSIs is an important task that can facilitate further processing and interpretation of WSIs.

ROIs are usually defined as local regions that contain disease-related cells or tissues with specific patterns. However, ROIs do not have a clear or consistent definition across different scenarios and applications. Moreover, ROIs may vary in size, shape, location, appearance, and distribution within WSIs. Therefore, finding ROIs in WSIs requires robust and efficient methods that can handle these variations and complexities. The motivation behind this work is twofold. First, the Yu Lab has considered this solution as a tool in case of a surgery of a patient where the fast study of tissue is crucial. Here finding meaningful and insightful areas is a key factor, as it can help the surgeon to make a decision on the development of a disease or tumor. Second, the Yu Lab has been working on an external module for a conventional lab microscope which would allow it to replicate the scanning process of much more expensive industrial microscopes. This module would have multiple parts, such as a camera for a video feed, external motors for the movement of the microscope, and a computer for the processing of the images. The computer would be able to detect ROIs in the images and send the coordinates of the ROIs to the microscope, which would then move to the corresponding location. This would allow the user to quickly scan the tissue and find the most interesting areas.

In this report, we aim to study both supervised and unsupervised techniques for finding ROIs in WSIs using transformer models that can capture global dependencies and contextual information from large-scale data. For supervised techniques, we will look at TransUNet (J. Chen et al. 2021), a transformer-based model that combines a vision transformer (ViT) encoder with a U-Net decoder for medical image segmentation. We will also explore techniques such as transformer interpretability (Chefer et al. 2021a) that can provide insights into how transformers make decisions and what features they learn from WSIs. For unsupervised techniques, we will use transformer weights from DINO (Caron et al. 2021), a self-supervised method that trains vision transformers with an online distillation process and achieves state-of-the-art results on multiple vision tasks. We will investigate how these weights can be used to find ROIs in WSIs without any manual annotation.

Related Work

One of the recent trends in finding ROIs in WSIs is to use deep learning techniques that can learn complex features and patterns from large-scale data. For example, Huang et al. 2020 proposes a fast ROI detection method that combines a deep convolutional neural network (CNN) with image processing techniques such as thresholding, filtering, and morphological operations. The CNN is trained on patches extracted from WSIs to classify them as ROI or non-ROI, while the image processing techniques are used to refine the ROI boundaries and remove noise. This method achieves high accuracy and speed on various types of WSIs. Another example is Nugaliyadde et al. 2020, which investigates the use of RCNN, a deep machine learning technique that consists

of a region proposal network (RPN) and a region-based CNN (R-CNN). The RPN generates candidate ROIs based on anchor boxes, while the R-CNN classifies them as ROI or non-ROI using features extracted by another CNN. This method can detect ROIs with different sizes and shapes using only a small number of labelled WSIs for training. However, both methods require manual annotation of ROIs for training, which can be time-consuming and subjective. To overcome this limitation, Benomar et al. 2021 presents a method that does not rely on manual annotation but uses local information extracted from sub-images of WSIs. The method applies unsupervised clustering algorithms such as k-means and hierarchical clustering to group sub-images based on their similarity in terms of color, texture, shape, and spatial distribution. Then, it selects representative sub-images from each cluster as potential ROIs using criteria such as entropy, contrast, homogeneity, energy, correlation, variance ratio index (VRI), mean square error (MSE), peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), etc. This method can identify ROIs in WSIs of renal cell carcinoma with high accuracy.

3.2 DINO: Attention maps learned through Self-Supervised Learning

3.2.1 Introduction

As we have discussed in chapter 2, we have introduced DINO, a self-supervised image representation method that trains Vision Transformers (ViTs) to capture rich visual features without any labels. DINO leverages a contrastive learning objective that encourages ViTs to produce similar outputs for different augmentations of the same image, while being different from other images. DINO also uses a teacher-student framework with an online distillation mechanism that stabilizes the training and enhances the feature quality.

The attention mechanism in Transformer models allows them to learn how to focus on different parts of an input sequence depending on the context. The attention mechanism computes a score for each pair of input tokens, which represents how much one token attends to another. The scores are then normalized with a softmax function and multiplied by the input embeddings to produce an output vector for each token (Dosovitskiy et al. 2021)

The attention maps are visualizations of these scores, which can show how different tokens relate to each other in terms of relevance, similarity, or dependency. The attention maps can be used to analyze and interpret what the Transformer models learn and how they make predictions. For example, we can use attention maps to see which words or phrases are most important for a classification task, or which tokens are aligned with each other for a translation task. In this section, these attention maps will be particularly useful to find regions of interest in Whole Slide Images.

3.2.2 Transformers Attention Heads allow for the visualization of the attention maps

The paper "Emerging Properties in Self-Supervised Vision Transformers" Caron et al. 2021 proposes a method to visualize attention maps from self-supervised vision transformers (ViTs). The self-attention module from the last block of ViT can be used to compute a saliency map for each image patch, which indicates how much each patch contributes to the final prediction. The saliency map can be obtained by multiplying the output feature vector of each patch by its corresponding attention weight, and then summing over all heads and layers. The paper shows that

this method can produce attention maps that are consistent with semantic segmentation labels, especially for self-supervised ViTs.

Self-attention is a mechanism that allows a model to learn how to relate different parts of an input sequence. In vision transformers, self-attention is applied to a sequence of image patches, which are small regions of an image that are treated as tokens. Each patch has a feature vector that represents its content, and a positional embedding that encodes its location.

Self-attention computes three matrices from the input sequence: query, key, and value. The query matrix represents what each patch is looking for, the key matrix represents what each patch offers, and the value matrix represents what each patch contains. Self-attention then calculates a similarity score between each pair of patches by multiplying their query and key vectors. The scores are normalized by a softmax function to obtain attention weights, which indicate how much each patch attends to another patch. The output sequence is obtained by multiplying the attention weights with the value vectors, and summing over all patches.

The attention map for a given patch is a vector that shows how much each patch contributes to its output feature. It can be computed by multiplying its query vector Q with all the key vectors K, applying a softmax function, and then multiplying with all the value vectors V. This can be written as:

$$\text{AttentionMap}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

where d_k is the dimension of the key vectors.

In this section, we will use the same approach on two datasets: Glioblastoma (IvyGAP) and Renal Cancer (cellphone). We hope that the attention maps will reveal which regions of the images are most relevant for the model's understanding of the slide image, or in other words, which parts of the slide image make them most distinctive. We also expect the different attention heads to focus on a semantically distinct part of the slides, which combined could bring insights to the histopathology slide.

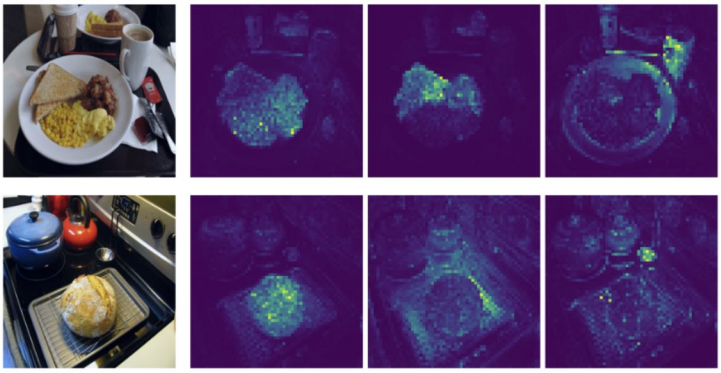


Figure 3.1: Examples of the attention maps generated by a ViT-S/8 model. We can see that the network can separate different part of the images overthought it has been trained with no labels or segmentation maps.

3.2.3 Experiments

Method

As in the experiment on self-supervised learning for robust pretraining (2.5.3), we will train a ViT-S/8 model on the Glioblastoma (IvyGAP) and Renal Cancer (cellphone) datasets. We will then use the attention maps generated by the last self-attention layer to visualize the regions of interest in the images, as done by Caron et al. 2021. As this model has 6 attention heads, we will plot the attention maps for each head separately, along with the averaged attention map for all heads. We expect that the attention maps will show different regions of interest for each head and that the averaged attention map will show a usable region of interest finder.. The model used is the fine-tuned model from 2.5.3. The hyperparameter found is a learning rate of 0.00017, a teacher temperature of 0.006, a weight decay of 0.02, with a ViT-8/S model and batch size of 8. As in the previous experiment, the teacher and student network share the same architecture, and both try to predict the same distribution of data. However the teacher has access to both local and global crops of the inputs, whereas the student only has access to local crops. The model was trained for 300 epochs with a cosine learning rate scheduler, using NVIDIA RTX 8000 GPUs.

Results

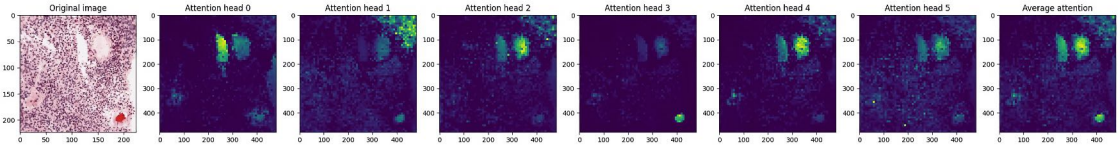
In figure 3.2, we used self-supervised learning with DINO to train a ViT-S/8 model on two medical image datasets: IvyGAP and Renal Cancer. We then visualized the attention maps generated by the last self-attention layer of the model to see what regions of interest it learned to focus on. The attention maps showed that different heads were able to attend to different features of the images, such as tumor cells, blood vessels, or missing tissue. The averaged attention map also showed a clear region of interest for each image, indicating that DINO was able to learn a robust representation of medical images without any labels. This suggests that self-supervised learning with DINO can be a useful technique for pretraining models for medical image analysis tasks, as it can capture relevant information from complex and diverse images.

In figure 3.2(a) and 3.2(b), we see an example of the attention maps generated by the model on the IvyGAP dataset. We can see that the attention maps are able to focus on different regions of the patch, as these areas were relevant to the generation of their embedding. The average attention maps provide a good balance of high interest regions in a patch, while keeping this information encoded in a single image.

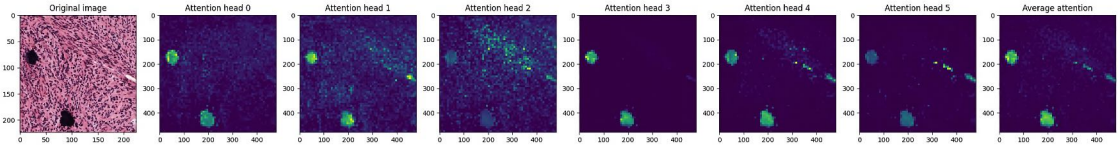
In the same way, we can see in figure 3.2(c) and 3.2(d) that the attention maps generated on the Renal Cancer dataset are able to find relevant features within an image, such as the large nest of tumor cells. We notice that the ocular light distortion in figure 3.2(c) is identified by the attention head 0.

3.2.4 Discussion

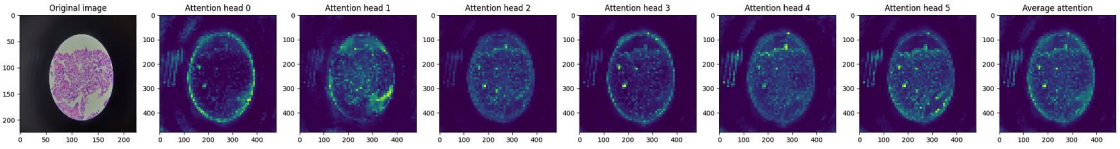
In this work, we have shown that self-supervised learning with DINO can be applied to medical image datasets to learn meaningful and interpretable representations. We have demonstrated that DINO can learn to attend to different features of the images, such as tumor cells, blood vessels, or missing tissue, without any supervision. We have also shown that the attention maps generated by DINO can be used to visualize and understand what regions of interest the model focuses on.



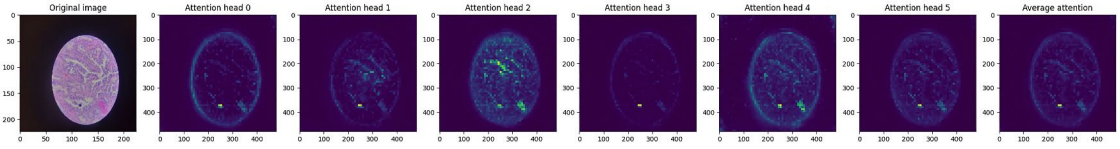
(a) Patch from IvyGAP. We see that the attention maps are able to focus on different regions of the patch that make this sample out of the ordinary in this dataset. Heads 0 and 2 focused on the missing tissue, head 1 on tumor cells, head 3 on blood vessels.



(b) Patch from IvyGAP. Most heads focused on the two aggregations of tumor cells, still head 2 focused on the patch of stretched tumor cells and head 5 on the missing tissue.



(c) Sample from the Renal Cancer Dataset from a cellphone. Head 0, 3 and 5 focused on the large nest of tumor cells while head 1 focused on the background of the slide. The average attention map provides a good overview of the attention maps of all heads.



(d) Sample from the Renal Cancer Dataset from a cellphone. Head 2 focuses on the background of the slide, and the remaining heads on the tumor cells.

Figure 3.2: Examples of attention maps generated by a ViT-S/8 model. We can see that the network can separate different part of the images overthought it has been trained with no labels or segmentation maps.

Our results suggest that self-supervised learning with DINO can be a powerful technique for pretraining models for medical image analysis tasks. By learning from unlabeled data, DINO can capture relevant information from complex and diverse images. This can potentially improve the performance and generalization of downstream tasks, such as segmentation, classification, or detection. Moreover, by generating attention maps that are easy to interpret and interpretability, DINO can provide insights into the model's behavior and reasoning. This can help clinicians and researchers to trust and validate the model's predictions.

One limitation of our work is that we only used a small model (ViT-S/8) and two datasets (IvyGAP and Renal Cancer) to evaluate DINO's performance on medical images. Future work could explore larger models (such as ViT-B/16 or ViT-L/32) and more datasets (such as Chest X-Ray or Brain MRI) to see how DINO scales up with more data and parameters. Additionally, future work could also investigate how to fine-tune DINO pretrained models on downstream tasks using labeled data or weak supervision.

3.3 Transformer Interpretability

3.3.1 Introduction

Transformers are powerful models that use self-attention mechanisms to process text and images. However, understanding how they make decisions is not easy. Existing methods for visualizing transformer attention either rely on the attention maps themselves or use heuristic rules to propagate relevance scores along the attention graph. These methods have limitations such as ignoring class-specific information or being sensitive to noise. In this section, we present a novel method for transformer interpretability by Chefer et al. 2021b that goes beyond attention visualization. Their method uses a gradient-based approach to compute class-relevance scores for each input token or pixel, and then aggregates them into class-relevance maps that highlight the most important regions for each class prediction. We evaluate our method on several transformer models applied to image classification tasks and show that it outperforms existing methods in terms of accuracy and robustness. Transformer Interpretability in this context as a weakly supervised method to find regions of interest. By training a model using class labels for the entire image, we hope to obtain a model that can also produce region maps without the need for segmented images. In this section, we will explain the approach taken by this Gradient Based Method to produce class-relevance maps, and evaluate on IvyGAP and Renal Cancer datasets to find regions of interest that correlate to patch class.

Related Works

Gradient-based methods are a class of techniques that use gradients to compute saliency maps for neural network visualization. These methods aim to highlight the input regions that have the highest influence on the output prediction for a given class. Gradient-based methods have several advantages over attention-based methods, such as being more robust to noise and being applicable to any neural network architecture. One example of gradient-based method is FullGrad (Full-Gradient Representation, Srinivas et al. 2019), which decomposes the neural network response into input sensitivity and per-neuron sensitivity components, and aggregates them with bias terms to obtain saliency maps. This method has the disadvantage of being class agnostic, in other words we can only see the relevancy score for the predicted class, and not for other potential classes. Another example is Grad-CAM (Gradient-weighted Class Activation Mapping, Selvaraju

et al. 2017), which uses the gradients of any target concept flowing into the final convolutional layer to produce a coarse localization map. Grad-CAM

LRC (Layer-wise Relevance Conservation, Binder et al. 2016) is a method that extends layer-wise relevance propagation (LRP), a popular technique for computing relevancy scores, to neural networks with local renormalization layers. LRC preserves the conservation property of LRP, which ensures that the sum of relevancy scores at each layer equals the prediction score at the output layer. LRC also provides intuitive and meaningful visualizations of relevancy scores for image classification tasks. Transformers use different non-linearities than ReLU, which produce both positive and negative features. This can cause numerical problems if skip connections are not properly handled. For example, LRP tends to break down in such situations. Self-attention layers also pose a difficulty because a simple propagation through them would not preserve the total amount of relevancy.

3.3.2 Method to compute relevancy score

Relevancy Score and LRP method

Relevancy score is a measure of how important an input feature is for a neural network prediction. It can be used to explain and understand how neural networks make decisions, as well as to improve their performance and robustness. However, computing relevancy scores for complex neural network architectures is not trivial. One of the challenges is dealing with non-linearities such as local renormalization layers, which are commonly used in convolutional neural networks.

LRP is a technique that assigns relevance scores to input features based on their contribution to the output prediction. It operates by propagating the prediction backward in the neural network, using a set of purposely designed propagation rules. The propagation rules are based on two principles: conservation and positivity. Conservation means that the total amount of relevance is preserved at each layer of the network. Positivity means that only positive contributions are propagated back, meaning that features that increase the output score receive positive relevance, while features that decrease it receive zero relevance.

The propagation rules can be different for different types of layers and activations. For example, for a linear layer with weights W and biases b, and an input x producing an output $y = Wx + b$, the propagation rule is:

$$R_i = \sum_j \frac{x_i w_{ij}}{\sum_k x_k w_{kj}} R_j$$

where R_i is the relevance score of input feature i and R_j is the relevance score of output feature j.

For a ReLU activation layer with input z producing an output $y = \max(0, z)$, the propagation rule is:

$$R_i = \frac{y_i}{z_i} R_i$$

where R_i is both the input and output relevance score of feature i.

The relevancy map is a visualization tool that shows how much each input feature contributes to a certain output class. It can be obtained by applying LRP to a neural network model and computing relevance scores for each input feature with respect to each output class.

For example, for an image classification task, the relevancy map can show which pixels in an image are most relevant for predicting a certain class label. The relevancy map can help understand how the model makes its decisions and identify potential biases or errors.

Gradient diffusion for transformers

We can now proceed to describe how this mechanism has been generalized by Chefer et al. 2021a for Transformers Models and their self-attention heads. We have a Transformer model M with B blocks. Each block b has self-attention, skip connections, and other layers. The model gets a sequence of s tokens with dimension d. One token is for classification: [CLS]. The model gives a vector y of length C for classification using [CLS]. The self-attention uses a smaller dimension d_h, where h is the number of heads and $hd_h = d$. The self-attention mechanism of a Transformer model is defined as follows:

$$\mathbf{A}^{(b)} = \text{softmax}(\frac{\mathbf{Q}^{(b)} \cdot \mathbf{K}^{(b)}}{\sqrt{d_h}}) \tag{3.1}$$

$$\mathbf{O}^{(b)} = \mathbf{A}^{(b)} \cdot \mathbf{V}^{(b)} \tag{3.2}$$

Let $(\cdot)$ be matrix multiplication. The attention output in block b is $O(b) \in \mathbb{R}^{h \times s \times d_h}$. The query, key and value inputs in block b are $\mathbf{Q}^{(b)}, \mathbf{K}^{(b)}, \mathbf{K}^{(b)} \in \mathbb{R}^{h \times s \times d_h}$, which are projections of an input $x(n)$ for self-attention. The attention map in block b is $\mathbf{A}^{(b)} \in \mathbb{R}^{h \times s \times s}$, where row i has the attention weights for each token with respect to token i. We apply softmax in equation 3.1 so that each row in each head of $\mathbf{A}^{(b)}$ sums to one.

Each attention map $\mathbf{A}^{(b)}$ has gradients $\nabla \mathbf{A}^{(b)}$ and relevance R^{n_b} for a class t, where n_b is the softmax layer in 3.1 of block b, and R^{n_b} is the layer's relevance. We define the output $\mathbf{C} \in \mathbb{R}^{s \times s}$ of our method as the weighted attention relevance:

$$\bar{\mathbf{A}}^{(b)} = I + \mathbb{E}(\nabla \mathbf{A}^{(b)} \odot R^{n_b}) \tag{3.3}$$

$$\mathbf{C} = \bar{\mathbf{A}}^{(1)} \cdot \bar{\mathbf{A}}^{(2)} \cdots \bar{\mathbf{A}}^{(B)} \tag{3.4}$$

with $\odot$ being the Hadamard product and E_h be the mean over the heads. We compute the weighted attention relevance with only the positive values of $\nabla A^{(b)} \circ R^{(n_b)}$ for positive relevance. We add the identity matrix to account for the skip connections in the block and avoid numerical instability. This method gives a matrix $C \in \mathbb{R}^{s \times s}$ for an input sequence of length s. Each row has a relevance map for each token with the other tokens - like the attention in 3.1, 3.4. For classification models, we only use the [CLS] token that explains the classification. The relevance map is from the row $C_{[\text{CLS}]} \in \mathbb{R}^s$ for the [CLS] token. This row has a score for each token's influence on the [CLS] token. Hence, for a vision transformer model, we can obtain the final relevancy map by reshaping the row $C_{[\text{CLS}]}$ to the patch grid size.

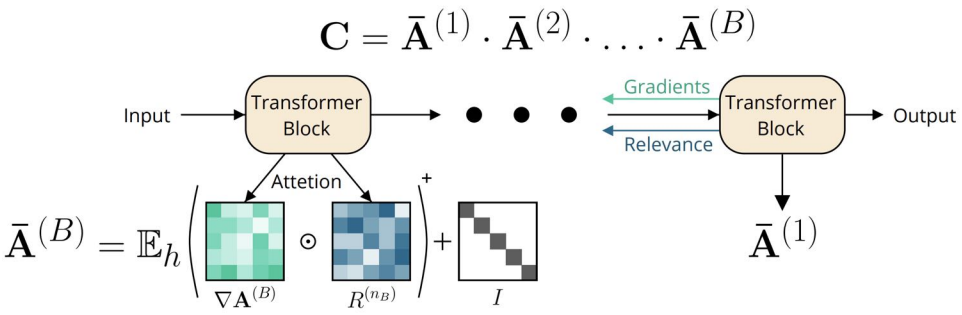


Figure 3.3: Transformer Interpretability architecture. The gradient $\nabla \mathbf{A}^{(b)}$ and relevancy $R^{(n_b)}$ are back-propagated in order to compute the attention and the final relevancy map.

3.3.3 Experiment

Method

In order to evaluate the performance of Transformer Interpretability (Chefer et al. 2021a), we will use a Vision Transformer (ViT, Dosovitskiy et al. 2021) with a classification head. We chose to be consistent with previous experiments, by using a ViT-S model with patch size 8. The loss function used was the cross-entropy loss function. The optimizer used was Adam with a learning rate of 0.001, with weight decay of 0.05. The model was trained for 30 epochs with batch size of 8. The model was trained on the IvyGAP dataset, which contains histopathological images of brain tumors. The dataset contains 224x224 patch images at 20x magnification. We used 38892 patches for training, 5917 for validation and 5983 for testing. The dataset contains 5 classes: Necrosis, Cellular Tumor, Background, Infiltrating Tumor and Tumor Border. The dataset is imbalanced, with the most common class being Cellular Tumor.

Results

The figures 3.4 show some examples of gradient-based relevancy maps for patches from the IvyGAP dataset, which contains histopathological images of brain tumors. The relevancy maps highlight the image features that correlate strongly with different tumor classes, such as necrosis, cellular tumor, background and infiltrating tumor. For instance, we can see that necrosis patches are characterized by the absence of tissue, cellular tumor patches are characterized by the presence of tumor cells and dying tissue, background patches are characterized by the absence of tissue and slide background, and infiltrating tumor patches are characterized by the presence of tumor cells and dying tissue. These relevancy maps can help us understand how the Transformer-based model makes its predictions and what kind of features it learns from the data.

Discussion

The results show that Transformer Interpretability can generate meaningful and intuitive relevancy maps for Vision Transformer models applied to histopathological image classification. The relevancy maps can provide insights into the model's decision-making process and the learned features that are relevant for different tumor classes. This can help us evaluate the model's performance and identify potential sources of error or bias. The relevancy maps can also help us improve the model's robustness and generalization by suggesting ways to augment the data

or modify the architecture. Finally, they can also help us communicate the model's predictions and explainability to domain experts and end-users, such as pathologists and clinicians. This can increase their trust and acceptance of the model and facilitate its adoption in clinical settings.

3.4 TransUNet

3.4.1 Introduction

Medical image segmentation is a challenging task that requires capturing both global and local features of images. Convolutional neural networks (CNNs) have been widely used for this task, but they have limitations in modeling long-range dependencies and preserving spatial information. Transformers, on the other hand, can model global context effectively with self-attention mechanisms, but they may lose fine-grained details due to tokenization and pooling operations. To address these issues, J. Chen et al. 2021 has proposed TransUNet, a hybrid model that combines Transformers and U-Net, a popular CNN architecture for segmentation. TransUNet uses a Transformer as a strong encoder to extract high-level semantic features from tokenized patches of images, and a U-Net as a decoder to recover spatial resolution and enhance local details. TransUNet outperforms various state-of-the-art methods on different medical image segmentation tasks, such as multi-organ segmentation and cardiac segmentation.

TransUNet and UNet are both U-shaped convolutional neural network architectures for image segmentation tasks. The main difference between them is that TransUNet uses a hybrid CNN-Transformer encoder to learn both high-resolution spatial information from CNNs and global context information from Transformers (Dosovitskiy et al. 2021). UNet uses only CNNs for both encoding and decoding, and relays strongly on skip connection and an autoencoder architecture.

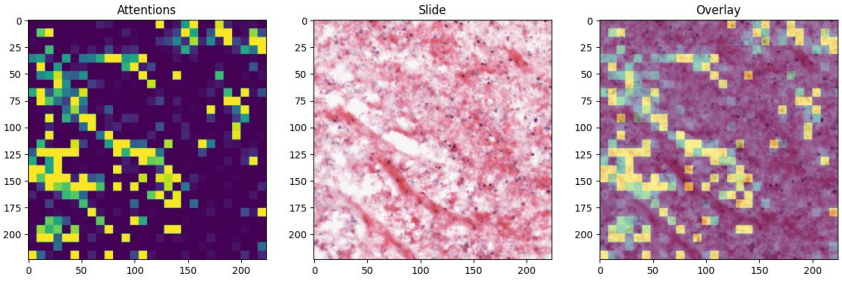
In this section, we will overview a final approach to finding regions of interest, which is to apply TransUNet to perform segmentation on labeled datasets. We will first introduce the main components and design choices of TransUNet, and then compare its performance with UNet and other baselines on the Renal Cancer Dataset and Glioblastoma dataset.

3.4.2 Transformer-CNN Hybrid Architecture

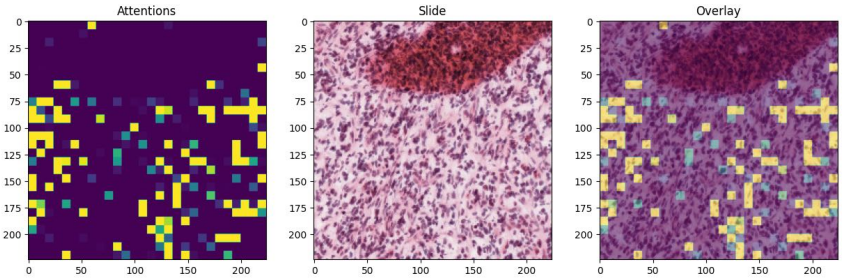
TransUNet is a hybrid architecture that combines Transformers and U-Net for medical image segmentation. The main components of TransUNet are:

A **hybrid encoder** that consists of a convolutional stem followed by a Transformer unit. The convolutional stem extracts low-level features from the input image and reduces its spatial resolution. The Transformer unit consists of multiple Transformer layers that apply self-attention and feed-forward networks to learn high-level semantic features from the patch embeddings. The patch embeddings are obtained by linearly projecting the vectorized patches of the image and adding learned position embeddings to encode spatial information.

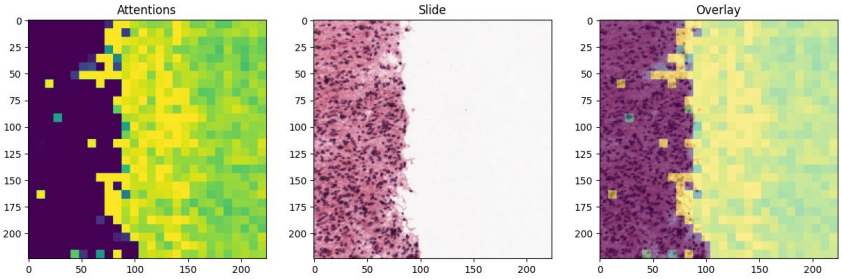
A **decoder** that consists of multiple upsampling blocks followed by a convolutional output layer. The upsampling blocks use transposed convolutions to increase the spatial resolution of the feature maps and concatenate them with corresponding feature maps from the encoder via skip connections. The skip connections help preserve fine-grained details and enhance localization abilities. The convolutional output layer produces the final segmentation map with pixel-wise predictions.



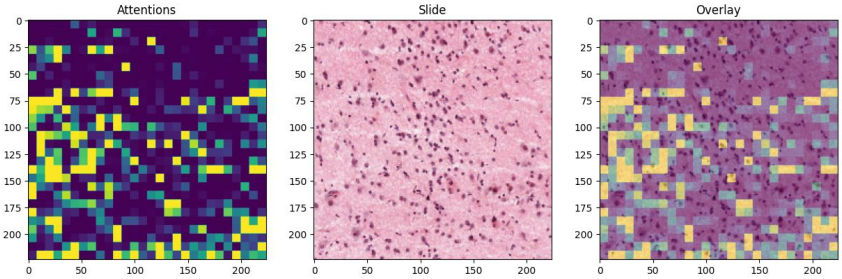
(a) Necrosis patch, and the gradient-based relevancy map. We see that the image features that correlates strongly with Necrosis are the absence of tissue in the patch.



(b) Cellular Tumor patch. The gradient-based relevancy map shows that the image features that correlates strongly with Cellular Tumor are the presence of tumor cells in the patch, and the dying tissue.



(c) Background patch. The gradient-based relevancy map shows that the image features that correlates strongly with Background are the absence of tissue, where we can see the background of the slide.



(d) Infiltrating Tumor patch. The gradient-based relevancy map shows that the image features that correlates strongly with Infiltrating Tumor are the presence of tumor cells in the patch, and the dying tissue.

Figure 3.4: Gradient-based relevancy maps for patches from the IvyGAP dataset.

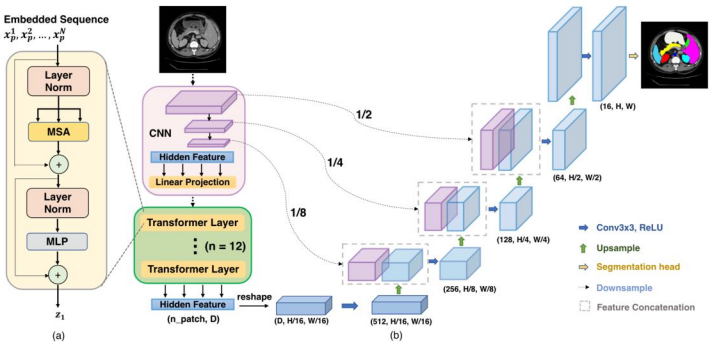


Figure 3.5: TransUNet architecture. The hybrid encoder consists of a convolutional stem and a Transformer unit. The decoder consists of multiple upsampling blocks and a convolutional output layer.

Skip connections that allow some layers in a neural network to be skipped and their outputs to be directly fed to later layers. Skip connections can help improve the performance and training of deep neural networks by: Preserving spatial information and fine-grained details that may be lost due to downsampling or pooling operations. Enabling feature reuse and fusion from different levels of abstraction and resolution. Reducing the vanishing gradient problem by providing alternative paths for gradient flow. There are two main types of skip connections: additive skip connections and concatenative skip connections. Additive skip connections add the output of one layer to the input of another layer, while concatenative skip connections concatenate them along a certain dimension. Additive skip connections are often used in ResNet-like architectures, while concatenative skip connections are often used in U-Net-like architectures.

TransUNet differs from UNet mainly in two aspects: (1) it uses a hybrid CNN-Transformer encoder instead of a pure CNN encoder to learn both high-resolution spatial information from CNNs and global context information from Transformers; (2) it uses an attention-based skip connection instead of a simple concatenation to enhance feature fusion between the encoder and decoder. These design choices make TransUNet more powerful and flexible for medical image segmentation tasks than UNet.

3.4.3 Experiment

Method

We will use the TransUNet model to perform the segmentation of classes over the Whole Slide Images of IvyGAP. We will compare its performance to UNet and report their test metric. The tuned hyperparameters are the learning rate, weight decay and batch size. We have found that the best learning rate for both model was $1e{-}5$, a weight decay of 0.005 along with a batch size of 16. Here we will report the Intersection over Union (IoU) for the evaluation of the segmentation. The IoU is defined as $IoU(A, B) = |A \cup B|/|A \cap B|$. The loss function used was cross-entropy loss, for 50 epochs with Adam optimizer. The model was trained on a NVIDIA RTX 8000 with 48GB of memory. The training time was around 1 hour for each model. We have evaluated the UNet and TransUNet models on 512x512 WSI from IvyGAP, and compared them with the ground truth segmentation map which contains 5 labels. We expect the TransUNet model to outperform the UNet model, as it is can use hierarchical patterns using the convolutional modules and exploit long-range dependencies using the Transformer modules. We used the original dataset split of IvyGAP, which is 474 slides for training, 283 for validation and 189 for testing. Results are shown in Table 3.1.

Results

Model	val/acc_best	test/acc	train/acc
UNet	0.6936	0.7021	0.7072
TransUNet	**0.7495**	**0.7403**	**0.7703**

Table 3.1: Comparison of UNet and TransUNet on three metrics: val/acc_best (the best validation accuracy achieved during training), test/acc (the test accuracy at the end of training), and train/acc (the training accuracy at the end of training). TransUNet outperforms UNet on all metrics, indicating that it can learn more effectively from the data.

The results in Table 3.1 show that TransUNet achieves significantly higher accuracy than UNet on both validation and test sets. This suggests that TransUNet can generalize better to unseen data and handle the variability and complexity of WSI segmentation. Moreover, TransUNet also has a higher training accuracy than UNet, indicating that it can fit the data better and avoid underfitting. The difference between training and validation accuracy is smaller for TransUNet than for UNet, implying that TransUNet has less overfitting and more robustness. The average IoU score for TransUNet is 0.7534, while for UNet it is 0.7010. This means that TransUNet can produce more accurate and consistent segmentation maps that match the ground truth labels. These results demonstrate the relative strenght of TransUNet over UNet for WSI segmentation, and confirm our hypothesis that combining convolutional and transformer modules can enhance the performance of deep learning models for this task.

3.4.4 Discussion

In this section, we have presented TransUNet (J. Chen et al. 2021), a novel deep learning model that combines convolutional and transformer modules for WSI segmentation. We have shown that TransUNet outperforms UNet, a widely used baseline model for this task, on three metrics: validation accuracy, test accuracy and training accuracy. We have also reported the IoU scores for both models, which indicate that TransUNet can produce more precise and consistent segmentation maps than UNet. Our results suggest that TransUNet can leverage the advantages of both convolutional and transformer modules to achieve better performance for WSI segmentation. Convolutional modules can capture local and hierarchical features in images, while transformer modules can model global and contextual information across the whole image. By combining these two types of modules, TransUNet can learn more effectively from the data and generalize better to unseen data.

Our work has several implications and contributions for the field of computer vision and medical image analysis. First, we have demonstrated that transformers can be successfully applied to WSI segmentation, a challenging task that requires high-resolution and fine-grained analysis of complex images. Second, we have provided empirical evidence that our model can outperform a strong baseline model on a large-scale dataset of WSI from IvyGAP. Third, we have shown that our model can produce high-quality segmentation maps that can facilitate the diagnosis and treatment of brain tumors.

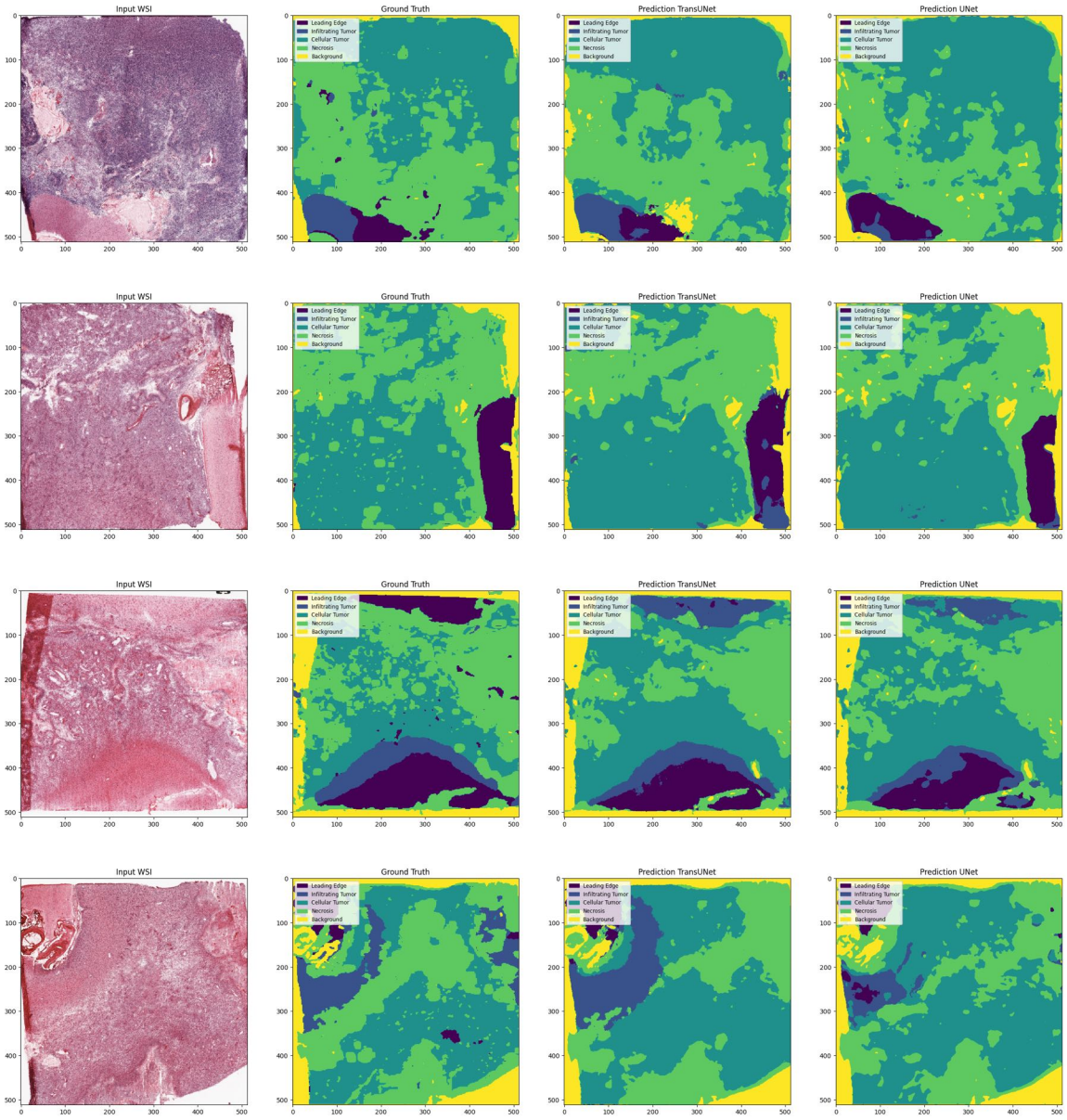


Figure 3.6: Qualitative comparison of segmentation results for TransUNet and UNet on four WSI samples. The first column shows the original WSI images, the second column shows the ground truth labels, the third column shows the segmentation maps produced by TransUNet, and the fourth column shows the segmentation maps produced by UNet. It can be seen that TransUNet generates more precise and smooth boundaries between different tissue types and preserves more details and structures than UNet.

3.5 Conclusion

In this chapter, we have explored various methods to identify regions of interest in whole slide images. This task is essential for pathologists and real world applications, as it can enable faster and more efficient workflows for medical purposes, and uncover new insights from slides. First, we presented an example of a self-supervised method that uses the DINO algorithm applied to Vision Transformers. We showed that the weights of attention heads trained using contrastive learning can generate attention maps that highlight unusual or abnormal features within a whole slide. Second, we introduced a weakly supervised method that uses Transformer Interpretability. This method employs a gradient-based approach to compute the relevancy map of the input given a specific output. In other words, this allows us to train a model for classification and then, given a specific class, we can locate the regions of the image that are most relevant to that class. Finally, we implemented an example of a strongly supervised method that uses TransUNet, which combines convolutional and transformer modules for WSI segmentation. We demonstrated that this method can surpass UNet models on a large-scale dataset of WSI from IvyGAP.

4 | Uncertainty Estimation with Posterior Network

4.1 Introduction

Uncertainty estimation is an important and often overlooked aspect of machine learning. In many real-world applications, it is crucial not only to make accurate predictions, but also to quantify the uncertainty in these predictions. For example, in autonomous driving, if the model is not sure about its prediction, it may result in a safer decision being taken, such as slowing down or stopping the vehicle. Similarly, in medical diagnosis, a doctor needs to be aware of the uncertainty in a prediction made by a machine learning model in order to make a more informed decision.

Given the importance of uncertainty estimation in machine learning, it is worth exploring the various methods that have been developed to address this challenge. In this report, we will take a closer look at two popular methods for uncertainty estimation in machine learning: Bayesian Neural Networks Blundell et al. 2015 and Ensemble methods Ovadia et al. 2019. Bayesian Neural Networks are a type of neural network that incorporate Bayesian methods to estimate the uncertainty in the predictions of the model. Ensemble methods, on the other hand, are based on the idea of combining the predictions of multiple models to produce a more accurate and uncertain prediction.

However, despite the success of these traditional methods, there is still room for improvement. In particular, Bayesian Neural Networks can be computationally intensive, and Ensemble methods can be difficult to implement and can increase the complexity of the model. To address these limitations, we will introduce Posterior Network, a state-of-the-art model for uncertainty estimation in machine learning. Posterior Network (Charpentier, Zügner, et al. 2020) combines the benefits of Bayesian Neural Networks and Ensemble methods, and provides a powerful tool for uncertainty estimation. In the following sections, we will explore the details of Posterior Network, its performance on various datasets, and how it compares to other methods for uncertainty estimation in machine learning.

Furthermore, we will also explore the use of Posterior Network for dataset cleaning. Dataset cleaning is a process in which the model is used to identify and remove outliers from the dataset. This is particularly relevant to the work of the Yu Lab as the Glioblastoma dataset IvyGAP contains an unknown amount of mislabeled samples. We will evaluate how Posterior Network can be used to estimate the confidence of a model prediction, but also the uncertainty of labeling of data, and show that an uncertainty-aware model outperforms models trained on mislabeled samples.

4.2 Uncertainty Theory and Related Work

Uncertainty estimation in machine learning is a complex and multi-disciplinary field that draws upon the theories and methods from mathematics, statistics, and computer science. At its core, uncertainty estimation involves quantifying the level of confidence in the predictions made by a machine learning model. To achieve this, one must first understand the sources of uncertainty in the model, which can be classified into two main categories: aleatoric uncertainty and epistemic uncertainty.

Aleatoric uncertainty captures the inherent noise or randomness in the data and is inherent to the problem being solved. For example, in a prediction task for the height of a person,

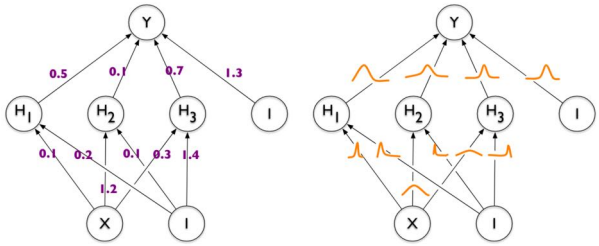


Figure 4.1: A Neural Network with fixed weights (left) and a Bayesian Neural Network (right). The parameters of the neural network are treated as random variables and are given a prior distribution.

the measurement of a person's height might have some random error, leading to a degree of uncertainty in the prediction. Aleatoric uncertainty can be modeled using probabilistic models such as Gaussian distributions, where the variance of the distribution represents the level of uncertainty in the prediction.

Epistemic uncertainty, on the other hand, captures the uncertainty in the model's parameters due to limited data. This type of uncertainty can be reduced as more data is collected, or as the model is improved to better capture the underlying patterns in the data. Epistemic uncertainty can be modeled using Bayesian methods, where the model's parameters are treated as random variables and their distribution represents the level of uncertainty in the model.

Models like Bayesian Neural Networks and Ensemble methods aim to estimate and reduce the epistemic uncertainty by combining the predictions of multiple models or modeling the parameters as random variables. However, these methods can be computationally intensive, and can be difficult to implement. Furthermore they do not address the question of data and labeling uncertainty, i.e aleatoric uncertainty.

4.2.1 Bayesian Neural Networks

Bayesian Neural Networks (BNNs) Blundell et al. 2015 are a type of deep learning model that combine the expressive power of neural networks with the probabilistic framework of Bayesian inference. This integration provides a natural way to estimate the uncertainty in the predictions made by the model, making BNNs an important tool for uncertainty estimation in machine learning. At its core, a BNN is a feedforward neural network with a probabilistic interpretation. In a standard neural network, the parameters of the model are treated as fixed, deterministic values that are learned from the training data. In contrast, in a BNN, the parameters are treated as random variables and are given a prior distribution that represents our prior knowledge about their values. During training, the model uses the training data to update its beliefs about the parameters and compute the posterior distribution, which represents the model's updated knowledge about the parameters. The posterior distribution can then be used to make probabilistic predictions about the target variable and estimate the uncertainty in these predictions.

The key to making a neural network Bayesian is to introduce a prior distribution over the model parameters. The choice of prior distribution will depend on the specific problem and the available prior knowledge. Common choices include Gaussian distributions, which are suitable for problems where the parameters are expected to have smooth and continuous values, and Dirichlet distributions, which are suitable for problems with categorical parameters. The posterior distribution

is then computed using Bayes' theorem (4.1), which states that the posterior is proportional to the product of the likelihood function and prior, given the training data. The likelihood function represents the probability of observing the training data given the model parameters, and is typically estimated using maximum likelihood estimation.

$$P(\theta|D) \propto P(D|\theta)P(\theta) \tag{4.1}$$

The ability to assign distributions to model parameters and predictions provides a crucial advantage in estimating uncertainty. By computing the variance of these distributions, we can accurately gauge our level of uncertainty, particularly when working with small datasets where standard neural net training is prone to overfitting. In addition to providing a way to estimate uncertainty, BNNs have several other advantages over standard neural networks. Firstly, they are more robust to overfitting, as the model is less likely to overfit to the training data when the parameters have a prior distribution. Secondly, they are more interpretable, as the posterior distribution provides a way to quantify the uncertainty in the parameters and the predictions. Finally, they provide a way to incorporate prior knowledge into the model, which can be useful when there is limited data or when the data is noisy (Gal et al. 2016).

The main disadvantage of BNNs is that they can be computationally expensive, as computing the posterior distribution requires evaluating the likelihood function for a large number of parameter values. To address this issue, several methods have been proposed to approximate the posterior distribution, including variational inference and Monte Carlo methods. These methods are designed to be fast and computationally efficient, while still providing a good approximation of the posterior distribution.

In conclusion, Bayesian Neural Networks are an important tool for uncertainty estimation in machine learning, providing a natural way to estimate the uncertainty in the predictions made by the model. By combining the expressive power of neural networks with the probabilistic framework of Bayesian inference, BNNs offer several advantages over standard neural networks, including robustness to overfitting, interpretability, and the ability to incorporate prior knowledge. Despite their computational complexity, recent advances in approximation methods have made BNNs a practical and valuable tool for a wide range of machine learning applications.

4.2.2 Dropout and Ensemble Methods

Ensemble methods and dropout are two important techniques used in machine learning for improving the performance of models and handling uncertainty. These techniques can be used separately or in combination to build models that are robust, accurate, and provide reliable predictions. In this report, we will discuss these techniques in detail and explore how they can be used to estimate uncertainty in machine learning models.

Ensemble

Ensemble methods are a class of machine learning algorithms that build multiple models and combine their predictions to make a final prediction. The idea behind ensemble methods is that by combining the predictions of multiple models, we can produce a final prediction that is more accurate than the prediction of any single model. Ensemble methods are widely used in machine learning and have been shown to be very effective in improving the performance of models in a wide range of applications.

There are several types of ensemble methods, including bagging, boosting, and stacking:

1. **Bagging**: Bagging (bootstrapped aggregating) is a simple ensemble method that involves building multiple models, each trained on a random subset of the data, and combining their predictions by taking a majority vote or by averaging. Bagging is effective in reducing the variance of the model, which makes it less sensitive to outliers and fluctuations in the data.
2. **Boosting**: Boosting is an ensemble method that builds a sequence of models, each of which tries to correct the mistakes made by the previous model. The models are weighted according to their accuracy, and the final prediction is made by combining the predictions of all the models. Boosting is effective in reducing the bias of the model, which makes it more accurate in generalizing to new data.
3. **Stacking**: Stacking is an ensemble method that involves building multiple models and combining their predictions by training a final model to make the final prediction. The final model is trained on the predictions of the individual models, rather than on the raw data. Stacking is a flexible ensemble method that can be used with any type of model and can be used to combine models of different types.

Ensemble methods are powerful techniques that can be used to improve the performance of machine learning models and handle uncertainty. By combining the predictions of multiple models, ensemble methods can produce a final prediction that is more accurate than the prediction of any single model. Furthermore, ensemble methods can be used to reduce the variance and bias of the model, which makes the final prediction more reliable.

Dropout

Dropout is a regularization technique used in machine learning to prevent overfitting and improve the generalization of models. Dropout works by randomly dropping out (i.e., setting to zero) a fraction of the neurons in the network during each training iteration. This means that the network is forced to learn multiple representations of the data, as different neurons are dropped out at different times. The final prediction is made by averaging the predictions of all the neurons in the network.

It is a simple and effective technique that can be used to improve the generalization of deep neural networks. By forcing the network to learn multiple representations of the data, dropout can prevent overfitting and improve the performance of the model on new data. Furthermore, dropout can be used in combination with other techniques, such as ensemble methods, to further improve the performance of the model.

Uncertainty Estimation with Ensemble Methods and Dropout

Ensemble methods and dropout can be used in combination to estimate uncertainty in machine learning models. By building multiple models and combining their predictions, ensemble methods can provide a measure of the variability of the predictions, which can be used as an indicator of the uncertainty in the predictions. For example, if the predictions of the individual models in an ensemble are highly varied, this suggests that the model is uncertain about the correct prediction, and the final prediction should be viewed with caution.

Similarly, dropout can be used to estimate the uncertainty of deep neural network models by using dropout at test time. At test time, the network is used multiple times with different dropout masks, and the final prediction is made by averaging the predictions of all the runs.

The variance of the predictions can be used as a measure of the uncertainty in the predictions.

In practice, ensemble methods and dropout can be used in combination to build models that provide both accurate predictions and reliable estimates of uncertainty. For example, a deep neural network can be trained with dropout to prevent overfitting and improve the generalization of the model. Then, multiple models can be built using different subsets of the data, and their predictions can be combined using an ensemble method to produce a final prediction. The variance of the predictions from the individual models can be used as an estimate of the uncertainty in the final prediction.

4.2.3 Confident Learning

Confident learning(CL) is a data-focused approach that centers on the quality of labels, as opposed to just the predictions of the model (Northcutt et al. 2022). Unlike traditional methods that concentrate on the model's predictions, CL takes into account the quality of the data by identifying and removing label errors in the dataset. The approach involves pruning noisy data, determining the amount of noise through probabilistic thresholds, and prioritizing examples to train with a higher degree of confidence. In this way, CL provides a comprehensive solution for ensuring the quality of labels, which is essential for accurate machine learning. In other words, when Bayesian and Ensemble methods focused on estimating epistemic uncertainty, Confident Learning provides a way to find and reduce the aleatoric uncertainty in the dataset.

The method used by Confident Learning is to estimate the joint distribution between the observed labels (prone to error and noise) and the latent labels (predicted by the model). Confident Learning only takes two inputs to find samples that may be mislabeled: the noisy labels and the predicted class probabilities by the model. Indeed, this approach is model agnostic as long as the model can provide class probabilities. CL can directly find noisy labels by following three steps:

1. **Estimate the joint distribution**: The first step is to estimate the joint distribution between the observed labels and the latent labels. Given a noisy label $\tilde{y}$ and a potentially unknown true label y^*, then $Q_{\tilde{y},y^*}$ is the matrix of joint distributions $p(\tilde{y}, y^*)$. We can empirically estimate this matrix by counting the number of samples that are predicted as $\tilde{y}$ and have a true label y^*. This empirical estimate is noted as $\hat{Q}_{\tilde{y},y^*}$.

2. **Filtering out the noisy labels**: The second step is to filter out the noisy labels. The noisy labels are those that are predicted as $\tilde{y}$ but have a true label y^* that is different from $\tilde{y}$. Using the matrix $\hat{Q}_{\tilde{y},y^*}$, we can count the set of samples that are not in the diagonal to estimate the labeling error.

3. **Training without the noisy labels**: The third step is to train the model without the mislabeled samples, that have been pruned in the previous step. To keep the model's exposure to each class equal, CL use class weighting. The sample are reweighted with the class weight $\hat{Q}_{y^*}[i]/\hat{Q}_{\tilde{y},y^*}[i][i]$, where $\hat{Q}_{y^*}[i] := p(y^* = i)$ is the prior of the latent label i, and $\hat{Q}_{\tilde{y},y^*}[i][i] := p(\tilde{y} = i, y^* = i)$ is the discrete joint probability of the noisy and true label being both equal to class i.

This pipeline is illustrated in the following figure 4.2. First, the model θ is trained on the noisy labels $\tilde{y}$. The model produces noisy predicted probabilities and classes. Cleanlab, the python framework developed for Confident Learning, is used at that stage to estimate the joint probability between the noisy and true labels, by counting the number of samples that are

predicted as $\tilde{y}$ and have a true label y^*. In this example, we see in the confusion matrix that 56 samples were predicted as fox when they were actually a dog. One way to remove is to rank the samples by predicted probabilities in the wrong class, e.g we could keep the n-th samples that predicted fox instead of dog that have the lowest "fox" probability, or the greatest "class" probability. Finally we obtain a cleaner dataset, which can be used again for a new round of Confident Learning.

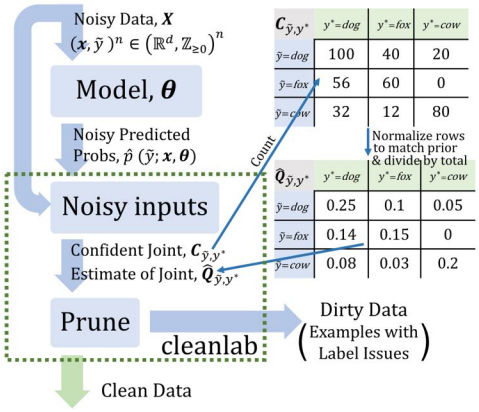


Figure 4.2: Confident Learning estimating the joint distribution between the observed labels and the latent labels.

In the context of her capstone project at the Yu Lab, graduate student Liza Mathews evaluated Confident Learning and its performance in improving model accuracy in the Glioblastoma dataset IvyGAP. She found that while Confident Learning was great at finding mislabeled samples in IvyGAP, removing those samples from the dataset had a marginal impact on the validation and test accuracy of this model. In the next section, we will evaluate Posterior Network (Charpentier, Zügner, et al. 2020, Charpentier, Borchert, et al. 2022), a Bayesian Neural Network-like approach to estimate distributions of predicted probabilities.

4.3 Posterior Network: OOD detection without OOD data

4.3.1 Introduction & Motivations

Quantifying uncertainty in predictions made by neural networks is crucial for ensuring the reliability of Machine Learning systems, especially in sensitive domains such as robotics, finance, and medicine. Knowing the level of uncertainty in their predictions enables AI systems to adapt to new situations and avoid making decisions in uncertain or dangerous conditions. Despite its importance, traditional neural networks often exhibit overconfident predictions, even when dealing with data that is significantly different from the training data.

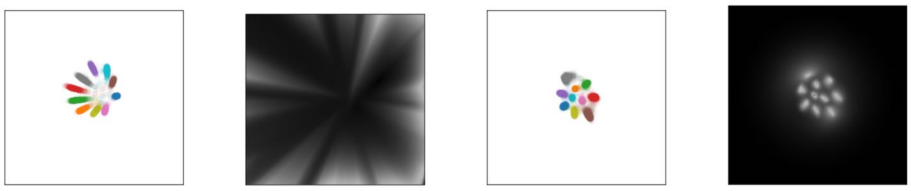
It is important to differentiate between two types of uncertainty in neural network predictions: aleatoric uncertainty and epistemic uncertainty. **Aleatoric uncertainty** arises from the inherent randomness of the data, such as the 50/50 chance of flipping a fair coin or the labeling process. **Epistemic uncertainty**, on the other hand, stems from the model's lack of knowledge of the data.

The estimation of uncertainty in neural networks is a rapidly growing research area, with various approaches being developed, including Bayesian Neural Networks (4.2.1), ensembles, and drop-out (4.2.2). Bayesian Neural Networks learn a distribution over the weights, while ensemble methods use a collection of sub-models to estimate the mean and variance of the class

probability distribution. Drop-out, too, has demonstrated remarkable performance in uncertainty estimation. However, these methods rely on implicit distributions for predictions and require a computationally expensive sampling phase for uncertainty estimation during inference time.

Traditionally, machine learning models would be trained on both in-distribution (ID) and out-of-distribution (OOD) samples, such as MNIST and FashionMNIST, to identify similar OOD samples during inference. However, Charpentier, Zügner, et al. 2020 have found that denying access to explicit OOD data during training can result in poor performance using these approaches. Surprisingly, these models exhibit increasingly confident predictions even for samples that are significantly different from the observed data.

In response to these limitations, Charpentier, Borchert, et al. 2022 propose the Posterior Network (PostNet), which assigns high epistemic uncertainty to out-of-distribution samples, low overall uncertainty in regions close to observed data of a single class, and high aleatoric and low epistemic uncertainty in regions close to observed data of different classes. PostNet leverages normalizing flows to learn a distribution over Dirichlet parameters in the latent space. They enforce the densities of individual classes to match the number of training samples in that class, aligning with the intuition that Dirichlet parameters correspond to the number of observations per class.



(a) Data labels - PriorNet(b) Uncertainty - PriorNet (c) Data labels - PostNet (d) Uncertainty - PostNet

Figure 4.3: PriorNet (Malinin et al. 2018) Posterior Net (Charpentier, Borchert, et al. 2022).

PostNet offers several advantages over traditional models, as it does not require any OOD samples for training, the specification of target prior distributions, or costly sampling for uncertainty estimation during inference time. This makes PostNet a promising approach for uncertainty estimation in machine learning. In figure 4.3, we can see a comparison of the predicted uncertainty on MNIST between the PriorNet (Malinin et al. 2018) and the Posterior Net (Charpentier, Borchert, et al. 2022). In both networks, the last step before inference was set to be a latent space of dimension 2 for visualization purposes. In subfigure (b) and (d) are visualized the uncertainty estimation of PriorNet and PostNet, where dark colors correspond to high uncertainty and light colors correspond to low uncertainty. We can see that beyond the training data, the PriorNet is overly confident in the overall uncertainty of the classes, whereas PostNet uncertainty evaluation is much more similar to the actual representation of the classes. Furthermore, we can note that PriorNet was trained with OOD samples, while PostNet was not.

4.3.2 Theory and Formulation

Previous approach to uncertainty estimation

First, we will describe mathematically the motivations behind uncertainty estimation, and how aleatoric and epistemic uncertainty plays a role in predicting the correct class for a given sample. Consider a sample x and a label y, in the context of this report x could be a patch from a whole slide image and y could be the patch class. As we have discussed previously, Bayesian Neural

Networks (4.2.1) models the distribution of the parameters w of the model θ, trained on a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ for a given sample x, in other words the predictive uncertainty of a model for classification is $P(w|x, \mathcal{D})$. Using Bayes' rule (4.1), we can write the posterior distribution of the parameters w as:

$$P(w|x, \mathcal{D}) = \int \underbrace{P(w|x, \theta)}_{\text{Data}} \underbrace{P(\theta|\mathcal{D})}_{\text{Model}} d\theta \tag{4.2}$$

The estimate of the uncertainty of the model's weights (LHS) can be described by an uncertainty tied to the data and the model (RHS). We will formally denote the **aleatoric uncertainty** as $P(w|x, \theta)$, and the model uncertainty $P(\theta|\mathcal{D})$. As this integral is not explicit for neural networks, approaches like Monte-Carlo dropout (4.2.2) and Bayesian Neural Networks (4.2.1) rely on sampling multiple models to have an estimate of the uncertainty:

$$P(w|x, \mathcal{D}) \approx \frac{1}{M} \sum_{i=1}^{M} P(w|x, \theta^{(i)}) \tag{4.3}$$

where $\theta^{(i)}$ is a sampled model from the epistemic distribution q of the categorical distribution prediction. In other words, given a sample x, in order to estimate the uncertainty of the model's weights, we can sample an ensemble of categorical distributions $P(w|x, \theta^{(i)})$, using models $\theta^{(i)}$ sampled from the epistemic distribution q of categorical distribution predictions. Each $P(w|x, \theta^{(i)})$ is a categorical distribution over the classes. This makes Monte-Carlo dropout methods expensive as we need to train M models to have the estimate of the uncertainty. Furthermore, for a given x, this approach can be modeled on a simplex. Each $P(w|x, \theta^{(i)})$ produces a set of probabilities for each class given x. On a simplex, each vertex is a different class, and the accumulation of the M categorical distributions provides a way to estimate and visualize the uncertainty of the model's weights. In this particular example, the uncertainty is low as most models of the ensemble are close to the top vertex.

(a) Ensemble (b) Distribution

Posterior Network approach to uncertainty estimation

We will know describe the framework of Posterior Network for uncertainty estimation. While we will change the notation to match Charpentier, Zügner, et al. 2020, we will maintain the goal of the previous subsection of estimating the predictive uncertainty of model for classification. Given an input $x^{(i)}$, there is uncertainty of the class prediction $y^{(i)}$ and the uncertainty of the model distribution prediction $\mathbf{p}^{(i)} = [p_1^{(i)}, ..., p_C^{(i)}]$. In other words, there is the aleatoric uncertainty of the prediction made for sample $x^{(i)}$, and the epistemic uncertainty of the model distribution prediction. For example, a sample $x^{(i)}$ could have the correct label prediction $y^{(i)}$ by the model, but $\mathbf{p}^{(i)}$ could be close to equal to a uniform distribution. This would be a low aleatoric uncertainty but high epistemic uncertainty.

The approach to unify aleatoric and epistemic uncertainty notation is to define the **epistemic distribution** $q^{(i)}$ of the categorical prediction $\mathbf{p}^{(i)}$. This is equivalent to saying that $\mathbf{p}^{(i)}$ the distribution of predictions for a sample $x^{(i)}$ is sampled from $q^{(i)}$. If we sample enough distribution of predictions $\mathbf{p}^{(i)}$ from $q^{(i)}$, we can now define the **aleatoric distribution** of the class prediction $y^{(i)}$ as $y^{(i)} \sim Cat(\bar{\mathbf{p}}^{(i)})$, where $\bar{\mathbf{p}}^{(i)}$ is the expected mean of distribution of predictions $\mathbf{p}^{(i)}$. Since $\mathbf{p}^{(i)}$ is sampled from $q^{(i)}$, we write $\bar{\mathbf{p}}^{(i)} = \mathbb{E}_{q^{(i)}}[\mathbf{p}^{(i)}]$.

Regarding the previous subsection, $\mathbb{E}_{q^{(i)}}[\mathbf{p}^{(i)}]$ is analogous $P(w|x, \mathcal{D})$ and $P(w|x, \theta^{(i)})$ would be sampled from $q^{(i)}$.

Going back to Posterior Network, the authors have modeled the epistemic distribution $q^{(i)}$ as Dirichlet distribution with parameter $\alpha^{(i)} = [\alpha_1^{(i)}, ..., \alpha_C^{(i)}]$, where $\alpha^{(i)}$ is the result of a model (potential a deep encoder) f_θ such that $f_\theta(x^{(i)}) = \alpha^{(i)}$. We obtain:

1. The epistemic distribution: $q^{(i)} = Dir(\alpha^{(i)})$
2. The aleatoric distribution: $\bar{p}_c^{(i)} = \frac{\alpha_c}{\alpha_0}$ where $\alpha_0 = \sum_{c=1}^{C} \alpha_c$
3. The class prediction: $y^{(i)} = \text{argmax}_c \, \bar{p}_c^{(i)}$

Bayesian Update

Given data points $\{y^{(j)}\}_{j=1}^n$, how do we perform an update of the parameter $\mathbf{p}$ of the categorical distribution $y \sim Cat(\mathbf{p})$? We will use Bayes' rule and properties of the Dirichlet distribution in order to find the update rule.

In this case, Bayes's rule for the update of the parameter $\mathbf{p}$ is:

$$\mathbb{P}(\mathbf{p}|\{y^{(j)}\}_{j=1}^n) \propto \mathbf{P}(\{y^{(j)}\}_{j=1}^n|\mathbf{p}) \cdot \mathbb{P}(\mathbf{p}) \tag{4.4}$$

First, recall that since $\mathbf{p}$ is drawn from q, and $q = Dir(\alpha)$, we can introduce a prior Dirichlet distribution, in other words, $\mathbb{P}(\mathbf{p}) = Dir(\beta^{prior})$ with $\beta^{prior} \in \mathbb{R}_+^C$. Second, we can estimate the likelihood $\mathbf{P}(\{y^{(j)}\}_{j=1}^n|\mathbf{p})$ by empirical counting the number of times each class c is observed in the data $\{y^{(j)}\}_{j=1}^n$. This latter quantity is denoted $\beta_c^{data} := \sum_j \mathbf{1}_{y^{(j)}=c}$. Finally, we can now use the property that the Dirichlet distribution to provide the distribution of the LHS. Since the LHS of the equation has a prior Dirichlet $\mathbb{P}(\mathbf{p})$ and the data points follow a categorical distribution, we have that $\mathbb{P}(\mathbf{p}|\{y^{(j)}\}_{j=1}^n) = \text{Dir}(\beta^{prior} + \beta^{data})$. We have found the bayesian update rule for the categorical distribution using the parameters of the prior and a count of the labels.

Learning distributions within a latent space

As we have discussed in the previous subsection, we can provide the distribution and an update rule to the categorical distribution if the labels y are explicit. Posterior Network does not learn directly from the labels but from a latent space, with the intent of having more flexibility in describing the distribution space. The input x is passed through an encoder f_θ and results into a latent space, i.e $f_\theta(x) = z \in \mathbb{R}^{latent}$. The posterior predictions are modeled from normalizing flow ϕ. Hence, we obtain a probability distribution $\mathbb{P}(z|c; \phi)$ which models the probability that z belongs to the class c in the latent space, which is modeled after a normalizing flow. Going back to the bayesian update 4.4, how can we define β^{data} in terms of elements z in the latent space

Let x^i be an input, we recall that $z^{(i)} = f_\theta(x^{(i)})$ is the mapping of the input within the latent space. Then we have that $\mathbb{P}(z^{(i)}|c;\phi)$ is the probability that we observed $z^{(i)}$ in the class c and generated by the distribution ϕ. Hence if N_c is the number of ground truth observations in class c, we obtain that $N_c \cdot \mathbb{P}(z^{(i)}|c;\phi)$ is the number of observations of class c that would also be mapped at $z^{(i)}$ in the latent space. This is analogous to the quantity β_c^{data} in the previous subsection, hence we can define:

$$\beta_c^{(i)} = N_c \cdot \mathbb{P}(z^{(i)}|c;\phi) = N \cdot \mathbb{P}(c) \cdot \mathbb{P}(z^{(i)}|c;\phi) \tag{4.5}$$

Using Bayes's rule, we can expend the RHS:

$$\beta_c^{(i)} = N \cdot \mathbb{P}(c) \cdot \mathbb{P}(z^{(i)}|c;\phi) = N \cdot \mathbb{P}(c) \cdot \frac{\mathbb{P}(c|z;\phi)\mathbb{P}(z;\phi)}{\mathbb{P}(c)} \tag{4.6}$$

$$= N \cdot \mathbb{P}(c|z;\phi)\mathbb{P}(z;\phi) \tag{4.7}$$

Going back to the subsection 4.3.2, we recall that the goal of most uncertainty models is to estimate the distribution of predictions for a sample $x^{(i)}$, i.e $\bar{\mathbf{p}}^{(i)} = \mathbb{E}_{q^{(i)}}[\mathbf{p}^{(i)}]$. We have found that $\mathbf{p}^{(i)}$ follows a Dirichlet distribution $\text{Dir}(\beta^{prior} + \beta_c^{(i)})$. Since the mean of a Dirichlet distribution is explicitly know, we can compute the estimator of the distribution of predictions for a sample $x^{(i)}$:

$$\mathbb{E}_{\mathbf{p}\sim\text{Dir}(\alpha^{(i)})}[p_c] = \frac{\beta_c^{prior} + N \cdot \mathbb{P}(c|z;\phi)\mathbb{P}(z;\phi)}{\sum_c \beta_c^{prior} + N \cdot \mathbb{P}(z^{(i)};\phi)} \tag{4.8}$$

From Charpentier, Zügner, et al. 2020 we can observe intuitive observations from the above equation. First for out-of-distribution data, we would have $\mathbb{P}(z;\phi) \simeq 0$, hence the aleatoric distribution p_c would converge to a flat prior distribution, for example if $\beta^{prior} = 1$, then $p_c = 1/c$ for out-of-distribution sample.

Second, we observe that increasing the size of the dataset N to infinity brings p_c to be equal to $\mathbb{P}(c|z;\phi)$, which is the class posterior.

4.3.3 Architecture overview and loss function

Architecture

Figure (4.4) gives an overview of the Posterior Network. It depicts three example inputs, $x^{(1)}$, $x^{(2)}$, and $x^{(3)}$, that are passed through the encoding neural network f_θ, resulting in their respective latent space coordinates $z^{(i)}$. The normalizing flow component is responsible for learning flexible (normalized) density functions $P(z|c;\phi)$, which are evaluated at the positions of the latent vectors $z^{(i)}$. These densities are then used to parameterize a Dirichlet distribution for each data point. The confidence in the Dirichlet distributions is determined by the density values, with higher densities indicating higher confidence. The sample $x^{(3)}$ is mapped to a point with (almost) no density, resulting in a predicted Dirichlet distribution with high epistemic uncertainty. Sample $x^{(2)}$ is ambiguous and could depict either the digit 0 or 6, resulting in a Dirichlet distribution with high aleatoric uncertainty but low epistemic uncertainty. In contrast, the unambiguous sample $x^{(1)}$ has low overall uncertainty. Both the encoding network f_θ and the normalizing flow, parameterized by ϕ, are fully differentiable, allowing for their parameters to be learned jointly in an end-to-end fashion.

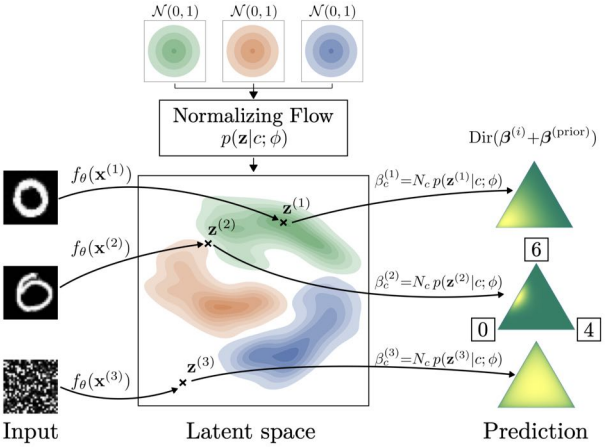


Figure 4.4: Posterior Network architecture

Uncertainty aware loss function

Recall that Posterior Network learns an epistemic distribution $q^{(i)}$ for each sample $x^{(i)}$, such that $q^{(i)}$ is as close as possible to the true posterior distribution of the categorical distribution $_{\text{I}}{}^{(i)}$ which is estimated from the training data. The parameters that we are trying to optimize are the parameters of the encoding network f_θ and the parameters of the normalizing flow ϕ. The objective function of Posterior Network consists of two components: the expected negative log-likelihood of the data given the model predictions and the entropy of the approximate posterior distribution over the model parameters. The first component evaluates the accuracy of the model predictions, while the second component acts as a regularizer that avoids overfitting and promotes smoothness. The objective function can be expressed as:

$$\min_{\theta,\phi} L = \min_{\theta,\phi} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{q^{(i)}}[CE(\mathbf{p}^{(i)}, y^{(i)})] - H(q^{(i)}) \tag{4.9}$$

where $\mathbf{p}^{(i)}$ is the predicted categorical distribution, $y^{(i)}$ is the one-hot encoded true class, $q^{(i)}$ is the approximate posterior distribution over the model parameters θ, and $H(q^{(i)})$ is the entropy of $q^{(i)}$. Optimizing this objective function approximates the true posterior distribution for the categorical distribution $\mathbf{p}^{(i)}$. The first component corresponds to the Uncertain Cross Entropy loss (UCE), which is known to increase confidence for observed data. The second component is an entropy regularizer. Because entropy regularizer employs the product logarithm of the density with the density, the term H is bounded by 1 and is minimized when the values of $\mathbf{p}^{(i)}$ are close either to 1 or 0, with maximum at 0.5. Hence minimizing $-H$ comes to maximizing H, where most values of the density $\mathbf{p}^{(i)}$ are close to 0.5, promoting class diversity.

4.4 Posterior Network for Uncertainty Prediction on IvyGAP

4.4.1 Introduction & Motivation

As we have discussed, Glioblastoma is a highly aggressive and heterogeneous brain tumor that poses significant challenges for diagnosis, prognosis, and treatment. Accurate segmentation of glioblastoma from magnetic resonance imaging (MRI) scans is crucial for clinical decision making and planning of surgical resection and radiotherapy. However, manual segmentation is

time-consuming, labor-intensive, and prone to inter- and intra-observer variability. Therefore, automated segmentation methods based on deep neural networks are a natural contender for segmentation and classification of these patches. However, these methods often suffer from over-confidence and lack of generalization, especially when faced with unseen or rare cases. Therefore, it is important to estimate the uncertainty of the segmentation predictions, which can provide useful information about the reliability and confidence of the model, as well as identify potential errors and outliers. Posterior Network is a novel framework for uncertainty estimation that leverages variational inference and implicit amortized flow to approximate the true posterior distribution over the model parameters given the data. Posterior Network can capture both aleatoric and epistemic uncertainty, as well as provide calibrated and sharp predictive distributions. In this paper, we apply Posterior Network to the task of glioblastoma segmentation on the IvyGAP dataset, which contains high-resolution MRI scans of 50 patients with glioblastoma. We evaluate Posterior Network on the IvyGAP dataset, in order to draw insights of which classes and samples may be problematic. In a later section, we will discuss how we can use the uncertainty estimates to clean the dataset and improve the accuracy of the model.

4.4.2 Experimental Method

In order to obtain a model calibrated for uncertainty estimation on IvyGAP, by Charpentier, Zügner, et al. 2020 there are two steps. First, train PostNet using a desired encoder f_θ and hyper-parameters, with the intent to minimize the Brier Score. The Brier Score is a metric that measures the square difference between two distributions. By using this metric as our uncertainty calibration metric, we ensure that the densities $\mathbf{p^{(i)}}$ are as closed as possible of the one hot encoded true class $y^{(i)}$. Second, we will need an Out of Distribution dataset to assert that we have a calibrated classifier of In Distribution vs Out of Distribution samples. In their work, the authors of Posterior Network have used CIFAR10 for In Distribution samples and SVHN for Out of Distribution samples. In our case, we will use IvyGAP dataset and Renal Cancer Dataset for Out of Distribution data, as this dataset can be out of distribution in terms of the tumor type.

The hyper-parameters tuned in this tasks are the dimension of the latent-space l, in range 2 to 8, the number of layers of the normalizing flow f in range 1 to 4, the learning rate lr in range 0.0002 to 0.00001 with a maximum of 40 epochs. To assert the best model, we used a weighted sum of the Brier Score, Epistemic Uncertainty and Aleatoric Uncertainty. This allow the model to provide accurate prediction by minimizing the Brier Score, but also to maximize its confidence. We used the following weighted sum:

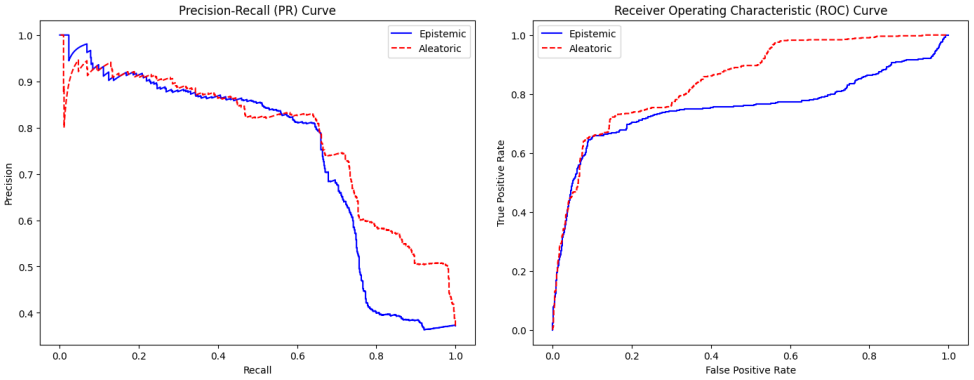
$$\text{Calibration Metric} = \frac{1}{3} \cdot \text{Epistemic Confidence} + \frac{1}{3} \cdot \text{Aleatoric Confidence} - \frac{1}{3} \cdot \text{Brier Score}$$

The model architecture is a Vision Transformer with patch size 8 (Dosovitskiy et al. 2021), and the sweep was done using multiple RTX 8000 GPUs with 48GB of memory with a batch size of 8.

Results

Tunning the hyper-parameters of the model, we obtained a model for which the Brier Score is 0.278 (lower is better) and an accuracy of 0.828 on our test set. The results are in the range of Charpentier, Borchert, et al. 2022, where they found a Brier Score of 0.228 and accuracy of 0.8485 on CIFAR10 for In-Distribution data. To validate this model, we will perform

OOD detection using the same procedure as the authors of PostNet, which is to say evaluating the OOD detection using an external dataset. While they used CIFAR10 and SVHN for In-Distribution and Out of Distribution data, we will use the IvyGAP dataset and the Renal Cancer Dataset for In-Distribution and Out of Distribution data. The results are shown in Table 4.1.)



(a) PR curve for the calibrated model **(b)** ROC curve for the calibrated model

Figure 4.5: AUC scores for aleatoric and epistemic confidence

The resulting area under PR curve and ROC curve are shown in table 4.1. We see that as binary classifier of OOD vs ID data, PostNet can learns without OOD data if a new external dataset is similar to the data seen during training. In the next section, we will look closer at the uncertainty evaluation per class on IvyGAP and try to draw insights using this calibrated model.

Table 4.1: AUC scores for aleatoric and epistemic confidence

Confidence Type	AUC-PR	AUC-ROC
OOD Detection Aleatoric	0.7808	0.8502
OOD Detection Epistemic	0.7405	0.7550

Uncertainty distribution on IvyGAP

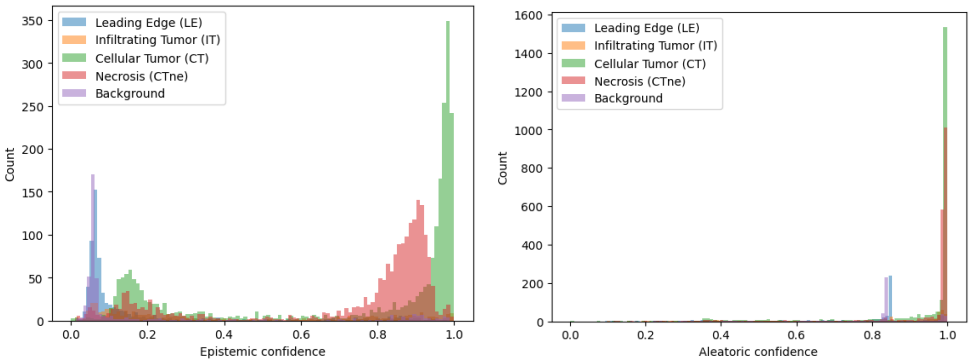
Furthermore, we can now visualize for each class within IvyGAP the aleatoric and epistemic distribution. The results are shown on figure 4.6. As a reminder, visualizing the epistemic uncertainty distribution of a calibrated model would tell us which classes the model has low or high confidence on its prediction. Similarly, the aleatoric uncertainty distribution will provide insights on the model's confidence on the original data and potentially reveal issues with its labels.

By analyzing the uncertainty distributions for each class in IvyGAP, we can gain a better understanding of the model's performance and identify potential areas for improvement. For example, if we observe high epistemic uncertainty for a particular class, this may indicate that the model requires more data or that the data for that class is particularly noisy or difficult to classify. If we observe high aleatoric uncertainty for a particular class, this may suggest that the data for that class is inherently ambiguous or that the labels are not well-defined. In either case, this information can be used to guide future data collection efforts or to refine the labeling process.

Class	Median Alea. confidence	Mean Alea. confidence	Median Epist. confidence	Mean Epist confidence
Cellular Tumor (CT)	**0.9986**	0.9221	**0.9416**	0.7169
Necrosis (CTne)	0.9910	**0.9357**	0.8558	**0.7180**
Infiltrating Tumor (IT)	0.8878	0.7925	0.2539	0.4099
Leading Edge (LE)	0.8479	0.7665	0.0719	0.1793
Background	0.8355	0.8325	0.0597	0.2158

Table 4.2: Median and mean values of aleatoric and epistemic confidence for each class, with the best value in bold.

In table 4.2 we can see that Cellular Tumor and Necrosis reached a high level of aleatoric and epistemic confidence, which can be expected as these classes are well defined in the dataset and are most present within the dataset. On the other hand, the classes Infiltrating Tumor, Leading Edge and Background have a lower aleatoric and epistemic confidence. Leading Edge (LE) is the class with the lowest mean Epistemic and Aleatoric Confidence. As we have discussed in the chapter 1, this may be due to the number of samples present in the dataset, but also ambiguity with Infiltrating Tumor. In figure 4.7, we have shown the confusion matrix of predictions made by the PostNet model. We observe a similar pattern that comes to confirm our findings. Infiltrating tumor is confused with Cellular Tumor and Leading Edge, and Leading Edge is confused with Infiltrating Tumor.



(a) Epistemic uncertainty distribution **(b)** Aleatoric uncertainty distribution

Figure 4.6: AUC scores for aleatoric and epistemic confidence

In this section, we have seen how we could calibrate our uncertainty estimator by minimizing the Brier Score as long as a curated Calibration Score. We have also seen how we could use the calibrated model to perform OOD detection, using the Renal Cancer Dataset as an external dataset. We have found a similar performance to the original paper in terms of model accuracy and uncertainty estimation. Finally, we have visualized the performance of the uncertainty estimator on the IvyGAP dataset, and we have found that the model has a high confidence on the classes Cellular Tumor and Necrosis, and a lower confidence in the classes Infiltrating Tumor, Leading Edge and Background. This can be explained by the number of samples present in the dataset, but also by the ambiguity between these classes.

In the following section, we will look at how we can filter out the data that is most uncertain, and how we can use this data to improve the model's performance.

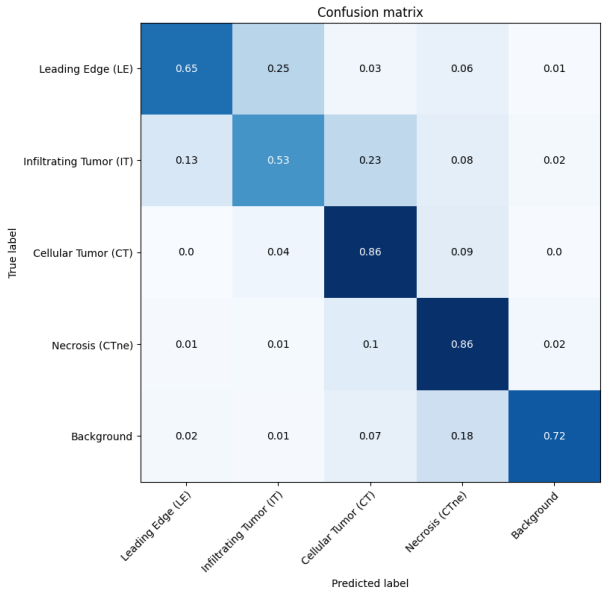


Figure 4.7: Confusion Matrix for the predictions of the calibrated model, normalized for each row.

4.5 Posterior Network for Dataset Cleaning

4.5.1 Introduction

In the previous section, we have seen how samples from IvyGAP could be prone to errors in their labels. We have calibrated a model for uncertainty prediction, which has allowed us to identify the samples that are most uncertain in terms of label confidence (aleatoric) or prediction confidence (epistemic). In this section, we will leverage this Uncertainty estimator to filter out the samples that are most uncertain, and we will use this data to improve the model's performance. More generally, we will describe a pipeline for uncertainty estimation which could be used for models applied to challenging datasets. In our experiments, we have seen an improvement in model accuracy on validation of up to 1.5 points, when removing the 15% most uncertain samples from the dataset. By removing the most uncertain samples from the dataset, we can create a cleaner and more reliable training set for our models. This is especially useful in situations where the dataset is noisy or has a high degree of label uncertainty.

4.5.2 Method

Following the results of the calibrated uncertainty estimator PostNet in section 4.4.2, we can now describe and motivate a pipeline for dataset filtering, with the intent of improving the model's performance on validation. First, a PostNet model is trained for uncertainty prediction, to minimize the Brier Score on validation. Once we obtain this calibrated model, we evaluate the aleatoric uncertainty of each sample in the dataset, as we have done in Figure 4.6. We then remove the p-% of samples with the highest aleatoric uncertainty (i.e lowest aleatoric confidence). This approach assures that we are removing the samples that are most likely to be mislabeled and do not fit within the distribution of predicted probabilities in the latent space. Finally, we train a new model on the filtered dataset, and we evaluate its performance on validation.

We can summarize this pipeline as follows:

1. Train a PostNet model for uncertainty prediction, to minimize the Brier Score on validation.
2. Evaluate the aleatoric uncertainty of each sample in the dataset.
3. Remove the p-% of samples with the highest aleatoric uncertainty.
4. Train a new PostNet model on the filtered dataset.

Hyperparameter tuning is done at step 1 in order to find a calibrated model, as in the previous section the latent space, the number of radial flows, the learning rate and weight decay are examples of the impactful parameter to tune. The percentage of samples to remove is a hyperparameter that can be tuned at step 3, using a linear search. In our experiment, we observed that 10 to 15% of the samples with the highest aleatoric uncertainty were leading to the best performance on validation.

4.5.3 Results

The results presented in figure 4.8 show the accuracy achieved by the model on the validation set, after removing a percentage of the most uncertain samples, evaluated by the PostNet model's aleatoric uncertainty predictions. The results demonstrate that the proposed pipeline for dataset filtering based on uncertainty estimation can improve the model's performance on validation. As seen from the results, the accuracy on validation varies with the percentage of samples removed, with the best performance achieved when removing 12% of the most uncertain samples. The accuracy achieved after removing 12% of the samples was 0.841, which is the highest accuracy achieved in the experiment.

We applied the pipeline described in the previous section to the IvyGAP dataset, and we evaluated the performance of the model on the validation set. We tested different values of p, i.e., the percentage of samples with the highest aleatoric uncertainty to be removed from the dataset. The results of this experiment are summarized in Figure **??**.

We observed that, as expected, the accuracy on validation generally improved as we removed more uncertain samples from the dataset. However, after a certain point, the performance started to deteriorate, suggesting that we might have removed too many samples from the dataset. In our experiment, we observed that removing the top 15% of the most uncertain samples led to the best performance on validation, with an accuracy improvement of up to 1.5 points, as shown in the following table:

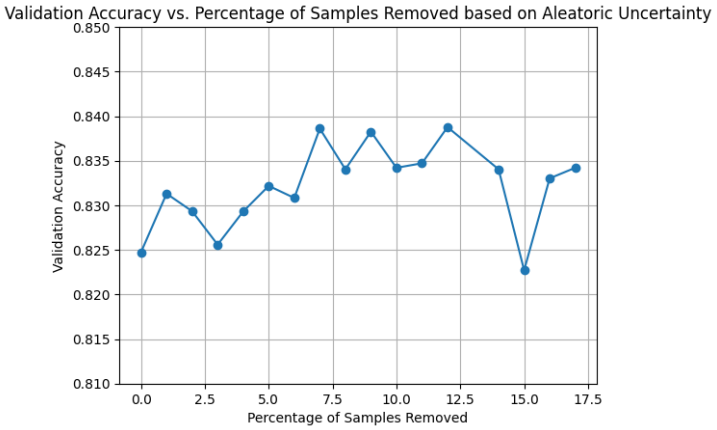


Figure 4.8: Performance of the model on validation for different values of p, i.e., the percentage of samples with the highest aleatoric uncertainty to be removed from the dataset.

These results suggest that the proposed pipeline can be applied to models trained on challenging datasets to improve their performance. By using a calibrated uncertainty estimator to identify and remove the most uncertain samples, we can create a cleaner and more reliable training set for our models. This approach can be especially useful in situations where the dataset is noisy or has a high degree of label uncertainty. The proposed pipeline can be implemented with relatively simple steps, making it easy to apply to different models and dtasets. Overall, the results suggest that uncertainty estimation can be a powerful tool for improving the performance of machine learning models.

4.6 Conclusion

In this section, we have presented a pipeline for dataset filtering based on uncertainty estimation. We have shown that by using a PostNet model to estimate the aleatoric uncertainty of each sample in the IvyGAP dataset, we can identify and remove the most uncertain samples that might degrade the performance of our model. We have demonstrated that this approach can improve the accuracy of our model on validation by up to 1.5 points, compared to using the original dataset.

Our results indicate that uncertainty estimation can be a useful tool for dataset filtering and quality control. By removing the samples with high uncertainty, we can reduce the noise and ambiguity in the dataset and make it more suitable for training our model. This can potentially lead to better generalization and robustness of our model on unseen data.

One limitation of our approach is that it relies on a separate PostNet model to estimate the uncertainty of each sample. This adds an extra computational cost and complexity to our pipeline. Moreover, the PostNet model itself might not be perfectly calibrated or accurate in its uncertainty estimates. Future work could explore alternative ways to estimate uncertainty without using a separate model, such as using dropout or ensembles.

Another limitation of our approach is that it does not take into account the diversity or importance of each sample in the dataset. By removing samples based solely on their uncertainty, we might lose some valuable information or introduce bias in the dataset. Future work could incorporate other criteria for selecting samples to remove or keep in the dataset, such as their similarity to other samples or their relevance to a specific task.

In conclusion, we have proposed a novel pipeline for dataset filtering based on uncertainty estimation. We have applied this pipeline to the IvyGAP dataset and shown that it can improve the performance of our model on validation. We hope that our work will inspire further research on using uncertainty estimation for dataset filtering and quality control.

5 | Conclusion

In this thesis, we have provided a comprehensive overview of the state-of-the-art transformers method for analyzing high-resolution pathology images. We have discussed the challenges of applying transformers to medical problems, using the glioblastoma dataset IvyGAP as a case study.

First, we have introduced in Chapter 1 the datasets of interest: IvyGAP for glioblastoma and the Renal Cancer Dataset. The former is a large-scale dataset of glioblastoma images that poses difficulties for analysis due to the heterogeneity of tumor regions and the noisy labels. The latter is a small dataset of renal cancer images produced by the Yu Lab, where each sample was captured by a cellphone camera using a microscope.

Second, we have presented in Chapter 2 and motivated the Vision Transformer (ViT) architecture that can capture long-range dependencies between data using an attention mechanism with reduced computational complexity compared to convolutional-based models. Then, we have presented Self-Supervised Learning (SSL) methods as a way to learn useful representations from unlabeled data and focused on DINO, a recently proposed method based on contrastive learning and distillation. We have shown how this representation can provide a semantic understanding of the data and how it can be used to pre-train transformers for downstream tasks. In particular, ViT model pre-trained with DINO on IvyGAP outperformed the model pre-trained with supervised learning on the same dataset.

Third, we have explored in Chapter 3 how we could find insightful regions of interest (ROIs) within a whole slide image using self-supervised learning, weakly supervised learning and strongly supervised learning. We have leveraged DINO to pre-train a ViT without the need for labels and obtained ROIs by evaluating the attention map without ground truth. We have also used gradient-based explainability methods to get insightful relevance maps over a whole slide image when training a model for multi-class classification. Finally, we have used TransUNet, a hybrid model that combines ViTs and U-Nets, to find ROIs in a whole slide image.

Fourth, we have discussed in Chapter 4 the different methods to estimate the uncertainty of prediction and labeling in medical image analysis. We have presented the Posterior Network architecture that allows detecting out-of-distribution (OoD) samples without the need for OoD dataset during training. Furthermore, we explained how one could calibrate a model for prediction and labeling uncertainty. Finally, we have proposed a novel method to improve confidence and accuracy of predictions made on IvyGAP by filtering out samples with low labeling confidence. This approach is particularly useful for medical image analysis where the labeling process is often noisy and requires human expertise.

To conclude, this thesis provides a comprehensive overview of the transformers method in analyzing high-resolution pathology images, focusing on the challenges and opportunities of applying this method to medical problems. Through the case study of IvyGAP, we have shown how self-supervised learning can be used to pretrain transformers without the need for labeled data and how to find regions of interest within a whole slide image. Moreover, we have presented methods to estimate the uncertainty of prediction and labeling and proposed a novel method to improve confidence and accuracy. We hope that this thesis will inspire further research in this area and contribute to the development of more accurate and robust medical image analysis tools.

Bibliography

Benomar, Mohammed Lamine et al. (12/2021). "Identifying Regions of Interest in Whole Slide Images of Renal Cell Carcinoma". In: *Research on Biomedical Engineering* 37.4, pp. 785–802. DOI: 10.1007/s42600-021-00178-9.

Binder, Alexander et al. (04/2016). *Layer-Wise Relevance Propagation for Neural Networks with Local Renormalization Layers*. arXiv: 1604.00825 [cs].

Blundell, Charles et al. (06/2015). "Weight Uncertainty in Neural Network". In: *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, pp. 1613–1622.

Caron, Mathilde et al. (05/2021). *Emerging Properties in Self-Supervised Vision Transformers*. arXiv: 2104.14294 [cs].

Charpentier, Bertrand, Oliver Borchert, et al. (03/2022). *Natural Posterior Network: Deep Bayesian Uncertainty for Exponential Family Distributions*. arXiv: 2105.04471 [cs, stat].

Charpentier, Bertrand, Daniel Zügner, and Stephan Günnemann (10/2020). *Posterior Network: Uncertainty Estimation without OOD Samples via Density-Based Pseudo-Counts*. DOI: 10.48550/arXiv.2006.09239. arXiv: 2006.09239 [cs, stat].

Chefer, Hila, Shir Gur, and Lior Wolf (04/2021a). *Transformer Interpretability Beyond Attention Visualization*. arXiv: 2012.09838 [cs].

— (06/2021b). "Transformer Interpretability Beyond Attention Visualization". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, pp. 782–791. DOI: 10.1109/CVPR46437.2021.00084.

Chen, Jieneng et al. (02/2021). *TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation*. arXiv: 2102.04306 [cs].

Chen, Ting et al. (10/2020). *Big Self-Supervised Models Are Strong Semi-Supervised Learners*. arXiv: 2006.10029 [cs, stat].

Doersch, Carl, Abhinav Gupta, and Alexei A. Efros (01/2016). *Unsupervised Visual Representation Learning by Context Prediction*. DOI: 10.48550/arXiv.1505.05192. arXiv: 1505.05192 [cs].

Dosovitskiy, Alexey et al. (06/2021). *An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale*. DOI: 10.48550/arXiv.2010.11929. arXiv: 2010.11929 [cs].

Ektefaie, Yasha et al. (11/2021). "Integrative Multiomics-Histopathology Analysis for Breast Cancer Classification". In: *NPJ breast cancer* 7.1, p. 147. DOI: 10.1038/s41523-021-00357-y.

Gal, Yarin and Zoubin Ghahramani (10/2016). *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. arXiv: 1506.02142 [cs, stat].

Glioblastoma (n.d.). https://www.gliocure.com/en/patients/glioblastoma/.

Huang, Junzhou et al. (11/2020). *Fast Regions-of-Interest Detection in Whole Slide Histopathology Images*. IntechOpen. DOI: 10.5772/intechopen.94238.

Jing, Longlong and Yingli Tian (02/2019). *Self-Supervised Visual Feature Learning with Deep Neural Networks: A Survey*. arXiv: 1902.06162 [cs].

Ledig, Christian et al. (05/2017). *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*. DOI: 10.48550/arXiv.1609.04802. arXiv: 1609.04802 [cs, stat].

Mahendran, Aravindh, James Thewlis, and Andrea Vedaldi (07/2018). *Cross Pixel Optical Flow Similarity for Self-Supervised Learning*. DOI: 10.48550/arXiv.1807.05636. arXiv: 1807.05636 [cs].

Malinin, Andrey and Mark Gales (2018). "Predictive Uncertainty Estimation via Prior Networks". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc.

Marostica, Eliana et al. (05/2021). "Development of a Histopathology Informatics Pipeline for Classification and Prediction of Clinical Outcomes in Subtypes of Renal Cell Carcinoma". In: *Clinical Cancer Research* 27.10, pp. 2868–2878. DOI: 10.1158/1078-0432.CCR-20-4119.

Northcutt, Curtis G., Lu Jiang, and Isaac L. Chuang (08/2022). *Confident Learning: Estimating Uncertainty in Dataset Labels*. DOI: 10.48550/arXiv.1911.00068. arXiv: 1911.00068 [cs, stat].

Nugaliyadde, Anupiya et al. (2020). "RCNN for Region of Interest Detection in Whole Slide Images". In: *Neural Information Processing*. Ed. by Haiqin Yang et al. Communications in Computer and Information Science. Cham: Springer International Publishing, pp. 625–632. DOI: 10.1007/978-3-030-63823-8_71.

Ovadia, Yaniv et al. (12/2019). *Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift*. DOI: 10.48550/arXiv.1906.02530. arXiv: 1906.02530 [cs, stat].

Pathak, Deepak et al. (11/2016). *Context Encoders: Feature Learning by Inpainting*. DOI: 10.48550/arXiv.1604.07379. arXiv: 1604.07379 [cs].

Puchalski, Ralph B. et al. (05/2018). "An Anatomic Transcriptional Atlas of Human Glioblastoma". In: *Science (New York, N.Y.)* 360.6389, pp. 660–663. DOI: 10.1126/science.aaf2666.

Sayed, Nawid, Biagio Brattoli, and Björn Ommer (04/2019). *Cross and Learn: Cross-Modal Self-Supervision*. DOI: 10.48550/arXiv.1811.03879. arXiv: 1811.03879 [cs].

Selvaraju, Ramprasaath R. et al. (10/2017). "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Venice: IEEE, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

Shah, Shital et al. (07/2017). *AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles*. DOI: 10.48550/arXiv.1705.05065. arXiv: 1705.05065 [cs].

Srinivas, Suraj and Francois Fleuret (12/2019). *Full-Gradient Representation for Neural Network Visualization*. arXiv: 1905.00780 [cs, stat].

Vaswani, Ashish et al. (12/2017). *Attention Is All You Need*. DOI: 10.48550/arXiv.1706.03762. arXiv: 1706.03762 [cs].

Yu, Kun-Hsing, Vincent Hu, et al. (08/2020). "Deciphering Serous Ovarian Carcinoma Histopathology and Platinum Response by Convolutional Neural Networks". In: *BMC medicine* 18.1, p. 236. DOI: 10.1186/s12916-020-01684-w.

Yu, Kun-Hsing, Tsung-Lu Michael Lee, et al. (08/2020). "Reproducible Machine Learning Methods for Lung Cancer Detection Using Computed Tomography Images: Algorithm Development and Validation". In: *Journal of Medical Internet Research* 22.8, e16709. DOI: 10.2196/16709.

Zhang, Richard, Phillip Isola, and Alexei A. Efros (10/2016). *Colorful Image Colorization*. DOI: 10.48550/arXiv.1603.08511. arXiv: 1603.08511 [cs].