ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

School of Information Sciences and Technology
Department of Informatics
Athens, Greece

Master Thesis
in
Computer Science

Exploring Deep Learning Methods for Medical Image Tagging

Foivos Charalampakos

Supervisors: John Pavlopoulos,
Ion Androutsopoulos

September 2022

Foivos Charalampakos

Exploring Deep Learning Methods for Medical Image Tagging

September 2022

Supervisor: Prof. John Pavlopoulos,

Ion Androutsopoulos

Athens University of Economics and Business

School of Information Sciences and Technology

Department of Informatics

Natural Language Processing Group

Athens, Greece

Abstract

Medical image classification is a key technique of Computer-Aided Diagnosis (CAD) systems and has seen remarkable advancements with the use of Deep Learning approaches. This thesis addresses the task of multi-label medical image classification, called medical image tagging, exploring some of the latest research topics and trends in the Deep Learning and Computer Vision fields. The task aims to automatically identify medical terms and its ultimate goal is to help clinicians in the diagnostic process. Towards this goal, we develop several systems that employ Deep Learning neural networks for encoding the images (and extra information that may be available in the data) and combine them with Machine Learning classifiers and Information Retrieval approaches in order to assign the proper terms to the images.

Περίληψη

Η κατηγοριοποίηση ιατρικών εικόνων είναι μια βασική τεχνική των συστημάτων διάγνωσης (Computer-Aided Diagnosis - CAD) και έχει γνωρίσει αξιοσημείωτες προόδους με τη χρήση τεχνικών Βαθιάς Μάθησης (Deep Learning). Η παρούσα διπλωματική εργασία πραγματεύεται το πρόβλημα της ταξινόμησης ιατρικών εικόνων σε πολλές κατηγορίες, που είναι γνωστό και ως επισημείωση ιατρικών εικόνων (Medical Image Tagging), εξερευνώντας μερικά από τα πιο πρόσφατα ερευνητικά θέματα στα πεδία της Βαθιάς Μάθησης και της Υπολογιστικής Όρασης (Computer Vision). Ο άμεσος στόχος του προβλήματος είναι η αυτόματη αναγνώριση ιατρικών όρων που περιγράφουν πιθανά ευρήματα σε μια εικόνα και ο απώτερος σκοπός είναι να βοηθήσει τους ιατρούς στη διαδικασία διαγνώσεων. Προς αυτήν την κατεύθυνση, αναπτύσσουμε συστήματα που χρησιμοποιούν νευρωνικά δίκτυα για την κωδικοποίηση των εικόνων (και επιπλέον πληροφορίας που μπορεί να βρίσκεται στα δεδομένα) σε συνδυασμό με ταξινομητές μηχανικής μάθησης (Machine Learning) και μεθόδους ανάκτησης πληροφορίας (Information Retrieval) προκειμένου να γίνει η αντιστοίχιση των κατάλληλων όρων στις εικόνες.

Acknowledgements

I would like to thank all the people that I cooperated with and I am grateful for the efforts they made in order for this work to be completed. First of all, I would like to sincerely thank my supervisor and professor, Ion Androutsopoulos, who gave me the opportunity to work with him and offered me constant guidance and advice. Next, I would like to thank John Pavlopoulos for his restless advisory offer and ideas regarding the content of this thesis. Last but not least, I would like to wholeheartedly express my gratitude to my family, friends and past professors that believed in me, guided and supported me throughout the years. I would not have gotten to where I am today without them.

Contents

Abstract v

Acknowledgements vii

1 Introduction 1

1.1 Thesis Structure . . . 2

2 Background and Related Work 3

2.1 Background . . . 3

2.1.1 Feature extraction . . . 4

2.1.2 Deep Learning for Image Classification . . . 4

2.1.3 Neural Networks Architectures . . . 9

2.2 Related Work . . . 18

3 Implemented systems 25

3.1 Deep Learning-based classifiers . . . 25

3.1.1 Baseline: Image encoder + FFNN . . . 25

3.1.2 Loss functions . . . 26

3.1.3 Optimization . . . 28

3.1.4 Ensembles . . . 31

3.1.5 Self-supervision . . . 32

3.1.6 Aggregating information from the labels . . . 35

3.1.7 Self-training with Teacher-Student . . . 39

3.2 (Neural) Retrieval-based approaches . . . 39

3.2.1 k-NN . . . 40

3.2.2 Weigthed k-NN . . . 40

3.2.3 Retrieval-augmented classification . . . 41

4 Experiments and results 45

4.1 Data . . . 45

4.1.1 ImageCLEFmedical Caption 2022 . . . 46

4.2 Evaluation measures . . . 48

4.3 Experimental Results . . . 49

4.3.1 Visual encoders . . . 49

4.3.2 Loss functions and optimization . . . 50

4.3.3 Ensembles . . . 52
4.3.4 Self-supervision . . . 52
4.3.5 Information from the labels . . . 53
4.3.6 Preliminary experiments with Teacher-Student self-training . . . 55
4.3.7 k-NN retrieval with majority voting . . . 56
4.3.8 Adjusting the weight of each neighbor . . . 56
4.3.9 Combining retrieval and classifier predictions . . . 56
4.3.10 ImageCLEFmedical Caption 2022 submissions . . . 57

5 Conclusions and Future Work **59**
5.1 Conclusions . . . 59
5.2 Future Work . . . 59

Bibliography **61**

List of Figures **73**

List of Tables **75**

List of Algorithms **76**

1 Introduction

Medical imaging is one of the most critical examinations that are currently used for conducting clinical diagnosis. Patients all over the world are often subjected to various screening examinations that help medical professionals provide a correct diagnosis for a potential medical condition [Wil+12]. Thus, it is clear that clinicians examine and analyze a large amount of medical images (i.e., MRI scans, X-rays and PET/CT scans) in their daily workflow. In many cases, this procedure can be time consuming and may also present difficulties in the case of inexperienced clinicians. In order to avail the examination process of such medical images, Computer-Aided Diagnosis (CAD) techniques can be used in order to assist professionals in the interpretation of medical images. Usually, these techniques revolve around the use of Computer Vision (CV) algorithms. Additionally, recent advancements in the Artificial Intelligence (AI) field and more particular in Deep Learning (DL), have led to increasingly accurate computer vision models that can prove very useful in the automation of medical image examinations as well as further tasks in the biomedical domain in general.

When it comes to medical imaging, there are several tasks where a professional can benefit from the assistance of a computer system during medical examinations. For example, the most prominent tasks are classification (e.g., identify if a condition is depicted in the image) [Sha+21], detection (e.g., detect the an object of interest and its position in the image) [Kij+19] and segmentation (e.g., isolate an object of interest from the rest of the image) [Hes+19]. This thesis discusses the task of multi-label medical image classification, also called medical image tagging. Medical image tagging, aims at identifying and assigning one or more medical terms (tags) that correctly describe possible diagnostic findings on a given image. Furthermore, it can constitute a first step in the automatic development of draft medical reports where the predicted tags could be used as keywords in a coherent textual diagnosis [KPA19a]. Furthermore, medical image tagging can accelerate the diagnostic process by providing possible findings present in an image and help clinicians by reducing their workload. Towards these directions, this thesis focuses on the study and development of systems that conduct medical image tagging using Deep Learning. Moreover, part of the work of this thesis was done in terms of the participation of AUEB's NLP Group[1] in the Concept Detection sub-task of the ImageCLEFmedical 2022 Caption task [Ion+22; Rüc+22]. The competition aimed to encourage the development of medical image tagging systems. We participated by addressing the problem with the use of deep learning architectures

[1] http://nlp.cs.aueb.gr/

for image representation in combination with multi-label classifiers and retrieval-based methods. Following our previous success [KPA19b; Kar+20; Cha+21], we managed to rank first in the campaign. We will describe our methods and present their results as well as additional work done with further experiments.

1.1 Thesis Structure

The rest of this thesis is organized as follows:

Chapter 2

Chapter 2 provides an introduction to image classification, how Deep Learning can be used to solve image classification tasks as well as work related to this thesis.

Chapter 3

Chapter 3 describes the methods that were implemented for the work of this thesis.

Chapter 4

Chapter 4 presents the data that was used in our experiments as well as the details and results of our experimentations.

Chapter 5

Chapter 5 concludes and proposes directions for future work.

2 Background and Related Work

In this section, we will provide insight regarding the image classification task and how machine (and more particular, deep) learning can be used to address it. Furthermore, we will present the neural network architectures that were employed in the experiments of this thesis as well as the approaches that our work was based on.

2.1 Background

The aim of image classification is the recognition of one or more concepts of interest that are present in the image. Essentially, we need to identify certain concepts that constitute the category(-ies) that we want the image to be classified into. In general, the classification task is divided in three main families:

- **Binary classification**: We need to separate the images in two different categories or find whether an image does or does not belong in just one category. For example, we care to tell apart whether a picture contains a cat (or not) or whether a picture contains a cat or a dog. Another example, in the medical domain, is to find whether an image from a medical examination is normal or not [Raj+18].

- **Single-label multi-class classification**: We need to find whether an image belongs to a certain category amongst many others. The categories in this setting are mutually exclusive. For example, we need to find whether an animal image contains a dog, a cat, a bird etc. In the medical domain, an example is that we could want to identify a certain pulmonary medical condition in a chest X-ray [Zon+21].

- **Multi-label multi-class classification:** In this setting, an image can belong to several categories simultaneously. For instance, we can have images that contain both dogs and cats. In the medical domain, a medical exam can contain multiple conditions that should be identified [Irv+19].

The majority of image classification problems are addressed using machine learning approaches. In the remainder of this section, we will discuss how machine and deep learning are applied in image classification and introduce the key neural network architectures that are used in the literature and in this thesis.

2.1.1 Feature extraction

As already mentioned, machine learning is extensively used for image classification problems. One of the most important steps of the classification pipeline is the detection and extraction of (visual) *features* from the raw form of the digital image (pixels). Features are essentially pieces of information about the content of an image. We hope that they can capture properties of the image that will help the classification algorithm distinguish the different categories and classify the images correctly. Features are divided in low-level (e.g., points, edges of different directions, shapes, corners, textures) and high-level (semantically meaningful features, e.g., ears, eyes etc.). For example, a cat classifier should be able to extract higher-level features such as the eyes, ears, legs and the tail in order to be able to classify a cat in an image. Choosing informative and discriminating features is a crucial element that plays a very important role in the effectiveness of the classification algorithm. In the past, computer vision (CV) and signal processing algorithms were used to extract features which were embedded into a vector, called the *feature vector* and then a machine learning algorithm was used w.r.t the feature space in order to classify the vectors into the different categories. The disadvantage of this approach is that feature extraction relied on hand-crafted image filters and low-level components [KVB88]. Feature engineering became extremely time-consuming as both content and size (i.e., how many features are needed) of the feature space, required extensive tuning. However, over the last decade, the rise of deep learning revolutionized image classification and the CV field in general [Dea20]. Contemporary deep neural networks are able to learn which features present in the image are useful for the classification task at hand. Moreover, they can learn to only pay attention to a certain amount of features. This ***feature learning***, also called ***representation learning***, is part of their training process and is performed in an automated way, removing the need for detecting the proper features from raw (image) data 'by hand'.

2.1.2 Deep Learning for Image Classification

Due to the great increase of computational resources and the availability of more and more data, deep learning has made a great leap over the last few years, showcasing significant results in various tasks and numerous data modalities. Deep learning is based on artificial neural networks (ANNs) which were inspired by information processing in biological systems and consist of layers of artificial neurons. The adjective 'deep' refers to the use of multiple layers and the number of neurons on each layer roughly corresponds to the number of (learnable) features. Figure 2.1 illustrates a single biological neuron with annotations that describe a single artificial neuron's function. Simply put, a biological neuron receives input signals from its dendrites from other neurons and sends output signals along its axon, which branches out and connects to other neurons. In the artificial

version, the input signal is represented by the numeric values $x_0, \dots x_n$, and the output signal is the weighted sum

$$\sum_i w_i x_i \tag{2.1}$$

The weight coefficients are learnable and their value and sign (positive or negative) control the influence of the signal. We usually initialize the weight variables by drawing random values from a predefined distribution. The sum is then passed through a non-linear activation function f which allows us to move to a non-linear feature space (i.e., increase the expressivity) instead of having a chain of linear transformations. In practice, a term b which is called bias is added to the sum before passing the result into f. The single artificial neuron with the sign activation function

$$f(x) = \begin{cases} 1, & \text{if } w \cdot x + b > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

is known as the Perceptron [Ros57], a binary linear classifier and the foundation of modern neural networks.

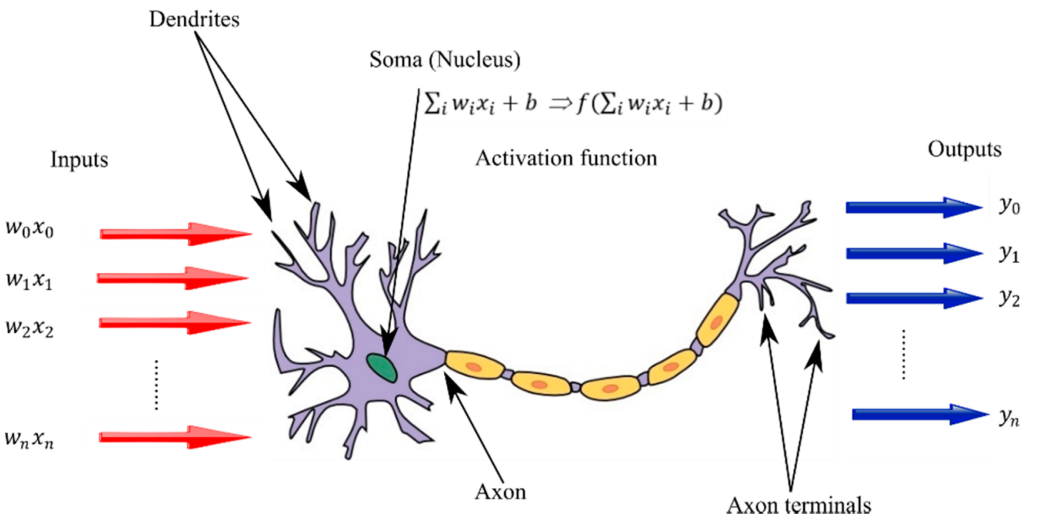

Fig. 2.1: An illustration of a biological neuron with annotations corresponding to the function of an artificial neuron. Figure taken from [KC+21].

But how can we use a neural network to process and classify images? In general, an image is represented as a three-dimensional matrix (tensor) where the first two dimensions correspond to the height and width of the images in pixels and the third dimension is the number of the color channels (usually, we have 3 RGB channels). The neural network

receives the 3D matrix as input and passes it through its layers. The result of the last hidden layer of the network is the feature vector of the input image. This vector is then passed to the last layer(s) of the network, called the classification head, which output(s) the probability(-ies) of the image belonging to a certain category(-ies).

Image classification can be casted as a supervised learning problem, meaning that we define a set of target classes (categories or concepts to identify in an image), and train the network to identify them using labeled example images (labels = 'supervision'). The objective of the training is to minimize a loss function that shows us how close the predicted distribution is to the actual distribution. We 'train' the network using the backpropagation algorithm [RHW86] which computes the gradient of the loss function with respect to the weights matrices with the use of the chain rule. The training process is essentially the (iterative) update of the network's weights with the goal of decreasing the loss objective function i.e., bring the predicted distribution as close as possible to the actual distribution of the categories. To find the optimal values of the weights (i.e., to find the global minimum of the differentiable loss function), the Gradient Descent (GD) optimization algorithm [Cau47; Lem12] is used. Essentially, we take steps in the opposite direction of the calculated gradient of the loss function at the current point, in order to move towards the steepest descent (see Fig. 2.2).

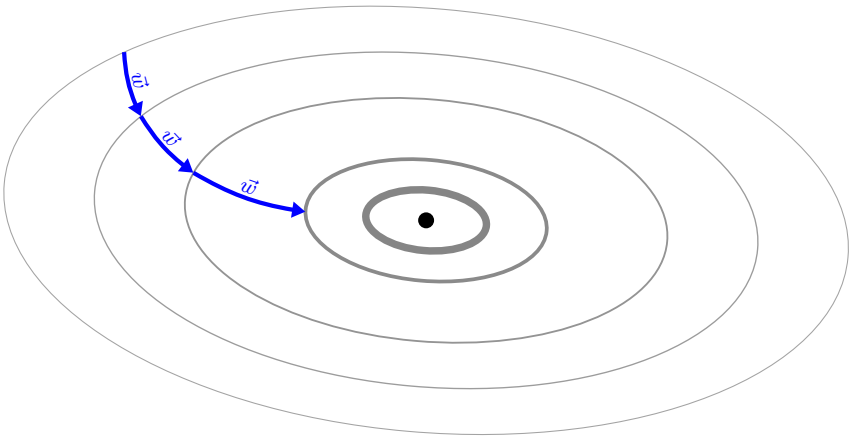

Fig. 2.2: An illustration of how Gradient Descent tries to reach the minimum of the optimization space by iteratively updating the weight parameters.

In practice, a stochastic approximation of the GD optimization, called Stochastic Gradient Descent (SGD) [RM51; KW52], is used. SGD replaces the actual gradient (calculated from all the data points) by an estimate which is calculated from a selected subset of the data. In its earliest adoption, SGD calculated the gradient based solely on one data point and was differentiated from mini-batch GD which used a mini-batch of data examples for the calculations. Nowadays, the 'SGD' term is used to describe mini-batch GD. SGD reduces the very high computational burden, especially in optimization problems with high-dimensional trainable parameters, achieving faster iterations (in trade for a lower convergence rate). Due to the stochasticity of the algorithm and the non-convex optimization space of most

minimization objectives (that contains many local minima), we now try to find a local minimum that is very close (in terms of loss value) to the global minimum of the loss function. Given a loss function $\mathcal{L}$ and a set of parameters $\boldsymbol{w}$, the GD method would perform the following calculation:

$$\boldsymbol{w}_{t+1} \coloneqq \boldsymbol{w}_t - \eta \nabla L(\boldsymbol{w}_t) = \boldsymbol{w}_t - \frac{\eta}{N} \sum_{i=1}^{N} \nabla \mathcal{L}_i(\boldsymbol{w}_t) \tag{2.3}$$

where N is the number of all the data examples and η is a small constant that controls the size of gradient step and is called the learning rate. $\nabla L(\boldsymbol{w}_t)$ denotes the set of partial derivatives of L w.r.t to the parameters set $\boldsymbol{w}$ at time-step t. $\nabla L(\boldsymbol{w}_t)$ is usually a vector or matrix containing partial derivatives. Respectively, SGD will perform the following:

$$\boldsymbol{w}_{t+1} \coloneqq \boldsymbol{w}_t - \eta \nabla L(\boldsymbol{w}_t) = \boldsymbol{w}_t - \frac{\eta}{M} \sum_{i=1}^{M} \nabla \mathcal{L}_i(\boldsymbol{w}_t) \tag{2.4}$$

where M is the number of data examples in one mini-batch and we iterate over the total number of mini-batches. Moreover, a common practice is to add a 'regularization' term to the loss function and practically minimize $\mathcal{L} + \lambda \left\| \boldsymbol{w} \right\|_p$ where $\left\| \boldsymbol{w} \right\|_p$ is the p-norm of the model's parameters and λ is a hyper-parameter that controls the regularization strength. In essence, we increase the loss and 'penalize' the model when the parameters take very high values. Regularization is usually adopted to prevent over-fitting, a phenomenon where a deep learning model of high capacity memorizes the data that it sees during training and fails to generalize well on unseen data.

Through the weights updates, the network implicitly acts as a feature extractor as the values of its weights showcase which features (neurons) are important for the task at hand. Therefore, the goal of these networks is to extract useful features from the images and encode them into feature vectors, creating a representation for each image, also called an image embedding. Because of this, these networks are called image encoders.

In order to train the network, we have to choose the loss function depending on the problem we want to solve. We usually use a form of cross-entropy in order to compare the two distributions mentioned above. When it comes to binary classification, the most common approach is the following: the final layer of the network contains one output neuron with a sigmoid activation function (Eq. 2.5) which outputs a probability in $[0, 1]$ that shows how likely it is that an example belongs to category 1:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

The network is trained using the binary cross-entropy loss function.

$$\mathcal{L}_{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p(y_i)) + (1 - y_i)\log(1 - p(y_i)) \tag{2.6}$$

where N is the number of training examples, y_i is the actual (gold) category of the i-th example and $p(y_i)$ is the output probability of the network for the i-th example. When we have a a single-label multi-class classification problem, the final layer contains as many neurons as the categories we want to identify and a softmax activation function (Eq. 2.7) that converts the logits to a vector of (normalized) probabilities in $[0, 1]$:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.7}$$

The network is trained using the categorical cross-entropy loss function.

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log(p(y_{ik})) \tag{2.8}$$

where K is the number of categories, y_{ik} is the k-th element of the one-hot vector of the gold category distribution of the i-th example. A one-hot vector is a vector filled with zeros except for one element which is 1 and denotes the actual category of the example. $p(y_{ik})$ is the k-th element of the predicted probability vector of the classifier for the i-th example. This approach can also be used with binary classification where the final layer will contain only two neurons (one for each class). Finally, when we are dealing with a multi-label classification problem, the final layer again contains as many neurons as the categories present in our data but we use the sigmoid activation function and the binary cross-entropy loss function. In essence, we reduce the problem to a series of C binary classification tasks (where C is the number of the labels). The model outputs one logit l_c per label. Each logit is then *independently* activated by a sigmoid function producing a probability of occurence $p_c = \sigma(l_c)$. The total loss is obtained by summing the binary loss from C labels (also see Eq. 2.6 for the total loss when $C = 2$).

$$\mathcal{L}_{BCE_{tot}} = \sum_{c=1}^{C} \mathcal{L}_{BCE_c} \tag{2.9}$$

2.1.3 Neural Networks Architectures

We have already seen the basic components of neural networks (i.e., neurons, activation functions etc.) as well as the Perceptron system. In this subsection, we will present the neural network architectures that have been used in the experiments of this thesis.

Multi-Layer Perceptrons (MLPs)

As their name suggests, MLPs [Ros57; Ros63] are an extension of the Perceptron [Ros57]. An MLP with one hidden layer and a non-polynomial activation can learn almost any bounded continuous function, unlike a linear Perceptron. These networks are composed of multiple (at least three - input, hidden, output) layers stacked together. Each layer contains a number of neurons and a non-linear activation function is applied to the output of each layer. Each neuron is connected with every neuron from the previous layer. That is why MLPs are also called fully-connected networks. For example, if we have an input layer of m neurons and a (first) hidden layer of k neurons, the output vector $\vec{o} \in \mathbb{R}^{k \times 1}$ of the hidden layer is:

$$\vec{o} = f(W^T \vec{x}) \Rightarrow \tag{2.10}$$

$$\vec{o} = \begin{bmatrix} f(s_1) \\ f(s_2) \\ \vdots \\ f(s_k) \end{bmatrix} = f\left(\begin{bmatrix} w_{1,1} & \cdots & w_{m,1} \\ w_{1,2} & \cdots & w_{m,2} \\ w_{1,3} & \cdots & w_{m,3} \\ \cdots & \cdots & \cdots \\ w_{1,k} & \cdots & w_{m,k} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix} \right)$$

where f is the non-linear activation function (e.g., the ReLU function [Fuk75]), $\vec{x} \in \mathbb{R}^{m \times 1}$ is the vector of the input and $W \in \mathbb{R}^{m \times k}$ is a learnable weight matrix that weighs the influence of the neurons (features).

When MLP networks are used for image processing, the pixels of the input image are flattened and fed to the network as a sequence of numbers (see Fig. 2.3). Therefore, such architectures do not take into account the spatial structure of images, treating pixels which are far apart in the same way as pixels that are close together. Moreover, the full connectivity between nodes result in computational intractability when dealing with higher resolution images (e.g., a layer receiving a 1000×1000-pixel image with 3 color channels will have 3 million ($1000 \cdot 1000 \cdot 3$) weights per fully-connected neuron).

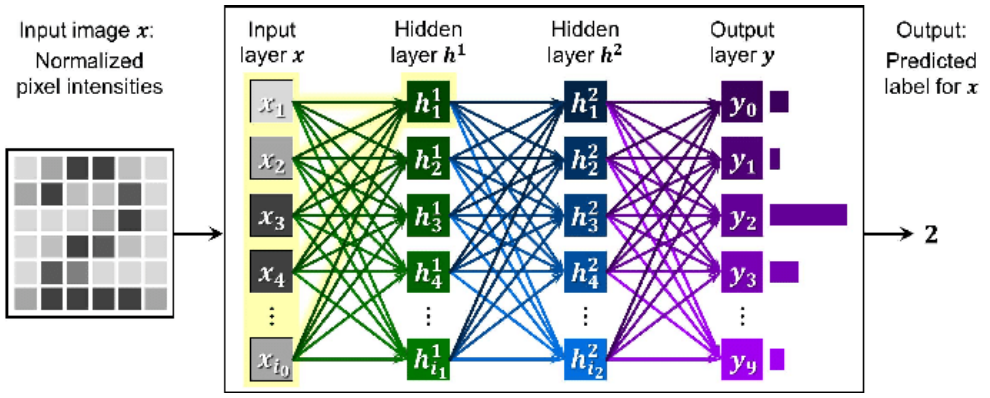

Fig. 2.3: An MLP for handwritten digit recognition. There are four layers (the input layer, two hidden layers and the final/output layer). The final layer receives the image representation vector (i.e., the output vector of the last hidden layer) as an input and outputs a probability distribution showing the probability of the input image belonging in each category. Figure taken from [Che+20b].

Convolutional Neural Networks (CNNs)

CNNs [LeC+89] are one of the most prominent neural architectures for image recognition. These networks consist of layers that apply learnable filters (kernels) to the image using the *convolution* operation by sliding the kernel over the (now non-flattened) 3D image input (see Fig. 2.4). There is also the option of leaving gaps between each slide (i.e., increasing the stride of the convolution). These kernels are the weights of the layer and also produce the features that are learned during training.

To compute the infinite sum of the convolution, we assume that the functions are 0 outside the boundaries of the image and kernels arrays [GBC16].

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.11}$$

where $m \times n$ is the dimension of the image in pixels. In practice, the convolution operation (Eq. 2.11) flips the kernel before the calculation so the cross-correlation operation (Eq. 2.12) is actually used which does not flip the kernel and prevents the calculations of redundant

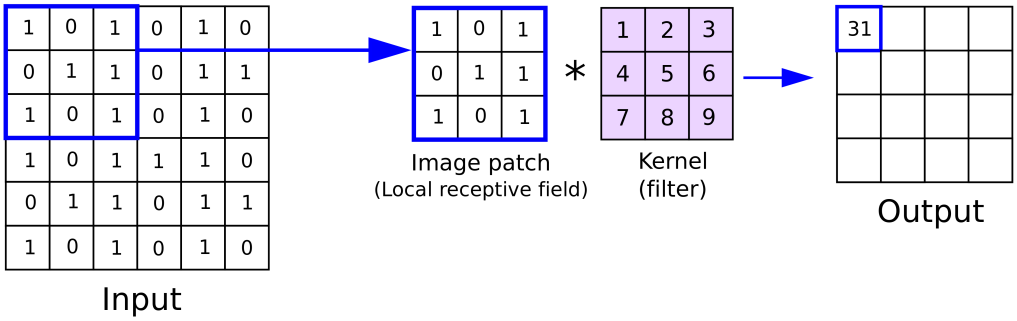

Fig. 2.4: An illustration of the convolution operation. The kernel slides over the image and the dot product between the current patch and the kernel is calculated. In CNNs, the kernel is a weight matrix which is updated during training. The remaining values of the output tensor are calculated likewise. Figure taken from [Rey19].

zeroed values. Because the kernels are learned, we do not mind that they are flipped. A non-linear activation and a bias term are also used in the calculations.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{2.12}$$

$$S(i,j) = f(S(i,j) + b) \tag{2.13}$$

Many kernels can be used in order to detect multiple features from the image and create a better representation. Additionally, padding the borders of the input image can also be used to maintain the initial dimensions throughout the network computations (in Fig. 2.4 the output tensor is smaller than the input due to the kernel slide). The resulting tensor corresponds to a new 'image' and is called a feature map.

Another key attribute of CNNs is pooling layers. Convolutional layers are usually followed by pooling layers that downsample the convolved features and provide translational invariance (by condensing the intermediate feature maps) while also reducing the parameters and increasing the receptive field more rapidly. Pooling layers contain a pooling window

(conceptually similar to the kernel - but not trainable) and stride. The most common variations are tha max and average pooling.

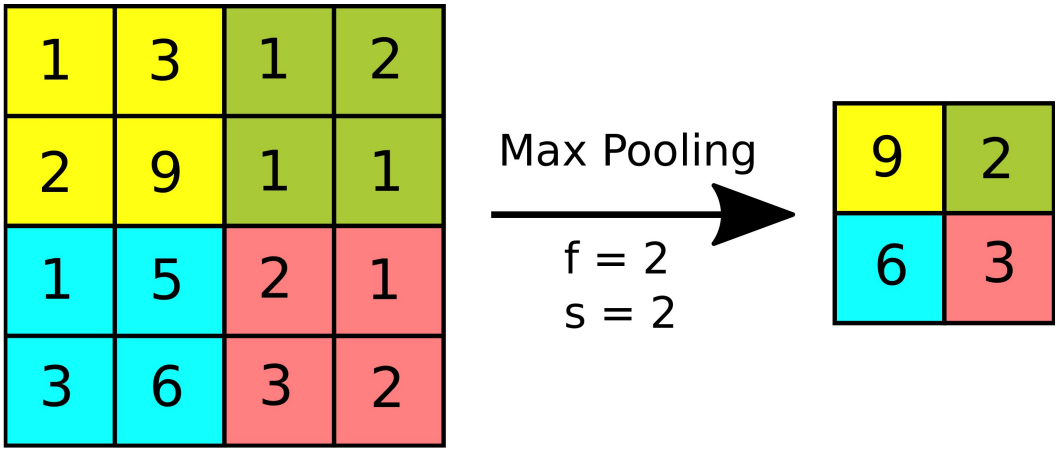

Fig. 2.5: Max pooling operation. f denotes the window size and s the stride. Max pooling down-samples the input by keeping only the max value of the window. Figure taken from [Rey19].

CNNs are variants of MLPs that, following the concept of human receptive fields, exploit spatially local correlation by enforcing a sparse local connectivity pattern between neurons of adjacent layers. Therefore, each neuron is connected to only a small region of the input volume and not to all of it like in MLPs. This is the so-called sparse interactions of CNNs (see Fig. 2.6). Due to their fewer parameters compared to MLPs, CNNs are more robust and less prone to over-fitting. The extent of this connectivity is called the *receptive field* of the neuron. CNNs ensure that the learned convolutional filters produce the strongest response to a spatially local input pattern. Stacking many convolutional and pooling layers leads to non-linear filters that become increasingly global (i.e. responsive to a larger region of pixel space) as the depth of the network increases, so that it first creates representations of small parts of the input and then assembles representations of larger areas from them. CNNs also incorporate the property of parameter sharing because a kernel can be used in all the parts of the input image (see Fig. 2.7). The feature vector that the CNN outputs is usually fed to a classification head, typically a (multi-nomial) logistic regression or an MLP, to get the classification probabilities.

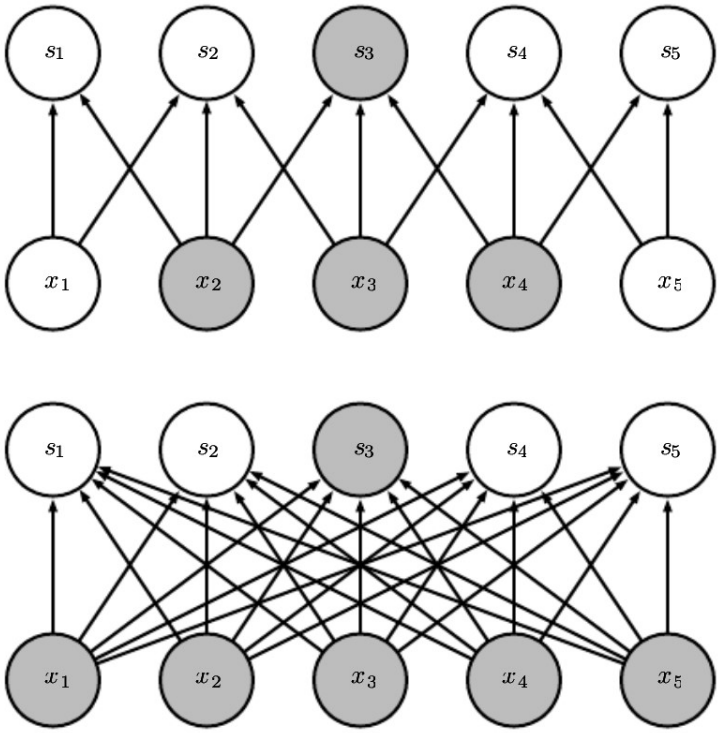

Fig. 2.6: (Top) When $\vec{s}$ is formed by convolution with a kernel of width 3, only three inputs affect s_3. (Bottom) When $\vec{s}$ is formed by matrix multiplication, connectivity is no longer sparse, so all the inputs affect s_3. However, even though direct connections in a CNN are very sparse, units in the deeper layers can be indirectly connected to all or most of the input image. Figure taken from [GBC16].

Vision Transformers (ViTs)

Over the past few years, the Transformer architecture [Vas+17] has dominated the field of Natural Language Processing (NLP), showcasing remarkable results in various languare related tasks. Transformers are based solely on the self-attention mechanism, succeeding Recurrent Neural Networks (RNNs), that were previously used in NLP, as the state of the art (SoTA) in text processing. The self-attention mechanism allows Transformer architectures to process word tokens in parallel (see Fig. 2.8), avoiding sequential processing like RNNs which process sequences timestep after timestep. Transformers use the Query-Key-Value (QKV) self-attention where the result of the operation is computed as a weighted sum of the values. The weight assigned to each value is computed by a function of the query with the corresponding key (Eq. 2.14). Query, Keys and Values are vectors that are linear projections of the initial word embeddings X (e.g., learnable or pre-trained WORD2VEC embeddings [Mik+13]) of the sequence [Vas+17].

$$Att(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.14}$$

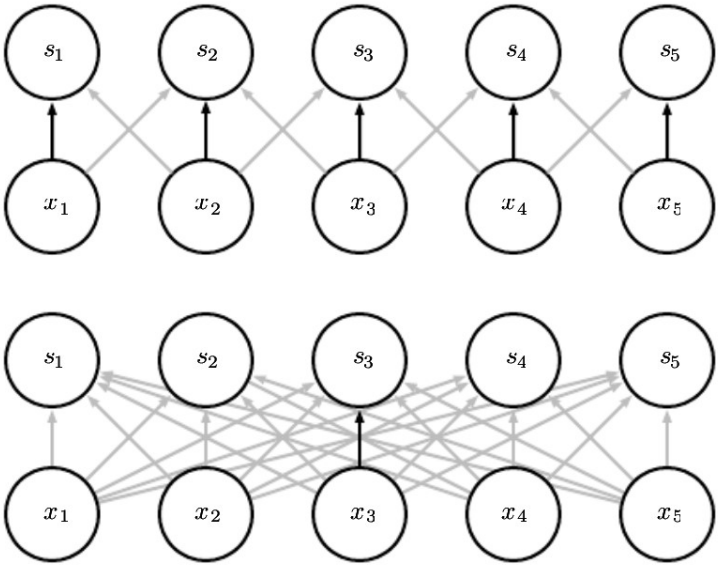

Fig. 2.7: (Top) The black arrows indicate uses of the central element of a 3-element kernel in a convolutional model. Because of parameter sharing, this single parameter is used at all input locations. (Bottom) The single black arrow indicates the use of the central element of the weight matrix in an MLP. This model has no parameter sharing, so x_3 affects only s_3. Figure taken from [GBC16].

where $Q = XW^Q$, $K = XW^K$, $V = XW^V$ and $W^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W^K \in \mathbb{R}^{d_{model} \times d_k}$, $W^V \in \mathbb{R}^{d_{model} \times d_v}$. QK^T is scaled with the square root of Q's and K's dimensionality $\sqrt{d_k}$ (Eq. 2.14) in order to avoid small gradients from the softmax operation due to the large magnitude of dot products. The calculation of the QKV attention is done many times by projecting the queries, keys and values n times with different, learned linear projection matrices W_i^Q, W_i^K, W_i^V and the results are then concatenated and projected again using $W^O \in \mathbb{R}^{nd_v \times d_{model}}$ [Vas+17] (see Fig. 2.8 and Eq. 2.15).

$$\begin{aligned} MultiHead(Q,K,V) &= \text{concat}(h_1,\ldots,h_n)W^O \\ \text{where } h_i &= Att(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.15}$$

Usually, 8 attention heads are used. d_{model} is the dimension of the output embeddings, d_k is the dimension of the Q and K vectors and d_v is the dimension of the V vectors. The output of the self-attention mechanism is then fed to an MLP. Residual connections [He+16] and Layer Normalization [BKH16] are also used (see Fig. 2.9). Furthermore, due

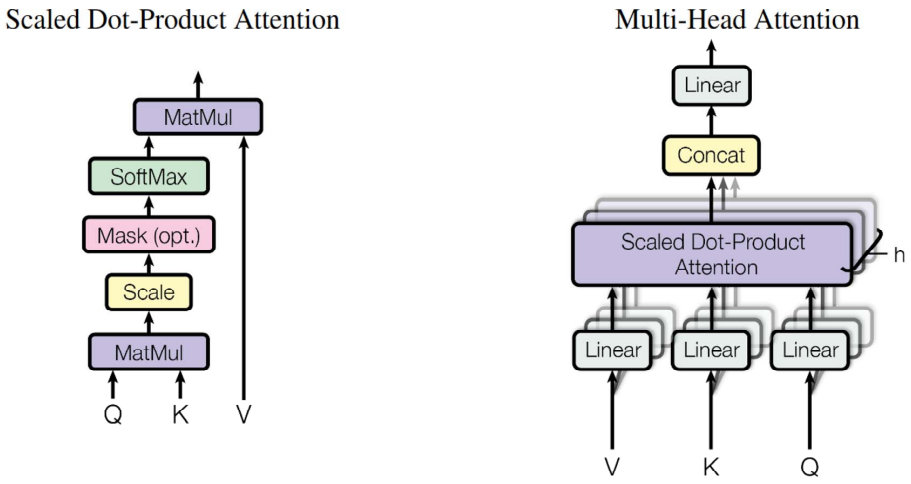

Fig. 2.8: (Left) The Scaled Dot-Product QKV Self Attention. Q, K, V are vectors. (Right) In practice, the operation is repeated many times and the results are concatenated (Multi-Head Attention). Figure taken from [Vas+17].

to the lack of sequentiality, Transformers use positional embeddings [Vas+17] (added to the input word embeddings) in order to know the order of the tokens in the sequence.

The parallel processing of the sequence also allows faster training and that is why there has been extensive work on training very high capacity Transformer models with large amounts of data (e.g., BERT [Dev+19]). These models have extensively been used the last couple of years with great results in NLP tasks.

Due to their success, Transformers have begun to enter the field of CV [Dos+21; Bel+19] as an alternative to CNN encoders, providing on par (or, in some cases, better) performance. Visual Transformers (ViTs) [Dos+21] receive images divided in non-overlapping patches and they handle them in parallel, like NLP Transformers receive word tokens (see Fig. 2.10). The image patches are encoded by a learnable linear layer and the output vector is fed to the Transformer encoder. The self-attention mechanism acts as a generalization of convolutional layers [CLJ20] as the window of the 'kernel' now is the whole sequence. Furthermore, Transformers are able to learn complex spatial relationships between high level image features. More specifically, CNNs are highly effective at recognizing features e.g., facial: eyes, ears, a nose etc. However, they struggle to learn the spatial relationships between these sub-components and other details such as the number of each of these components that should be present. As long as the facial sub-components are present, CNNs are likely to classify the image as containing a face. However, an object that contains multiple ears or more than two eyes will also be classified as a face. CNNs have no concept

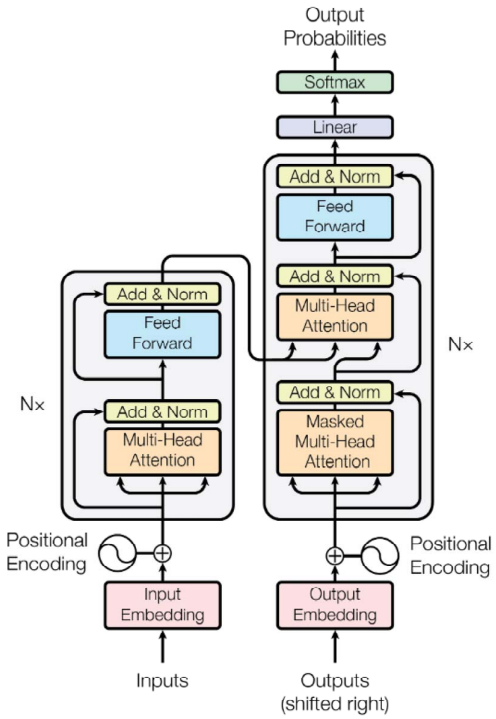

Fig. 2.9: The Transformer architecture. (Left) The encoder of the sequence. (Right) The decoder which predicts the next token (Transformers were initially introduced for machine translation). The decoder has two self-attention components. Masked self-attention which does not 'look' future tokens and a (non-masked) cross-attention mechanism in order to attend over the encoder outputs. Figure taken from [Vas+17].

of the number or relative position of these features. On the other hand, ViTs are harder to train than CNNs as they lack the inductive biases that CNNs inherently have due to weight sharing and translational invariance. Thus, they are way data hungrier than CNNs and require large datasets in order to have good performance.

Many variations of the original ViT have been proposed in the last few years [Liu+21; Tou+21a; Tou+21b].[1] ViTs can be used with CNNs as well, for example if we want to decrease the computational resources that we need. The CNN can be used to process lower level features in the image that will be fed to the Transformer encoder [Wu+21]. Another example is introducing the inductive biases of CNNs into ViTs [dAs+21; Dai+21].

Graph Convolutional Networks (GCNs)

GCNs [KW17] are a type of Graph Neural Networks (GNNs) that were proposed for graph data processing as a variant of CNNs, being able to aggregate and encode features based on the graph's structure (e.g., neighborhoods). The term 'convolutional' is used because weights parameters are typically shared over all locations in the graph. Additionally, the

[1] Also see: https://github.com/lucidrains/vit-pytorch

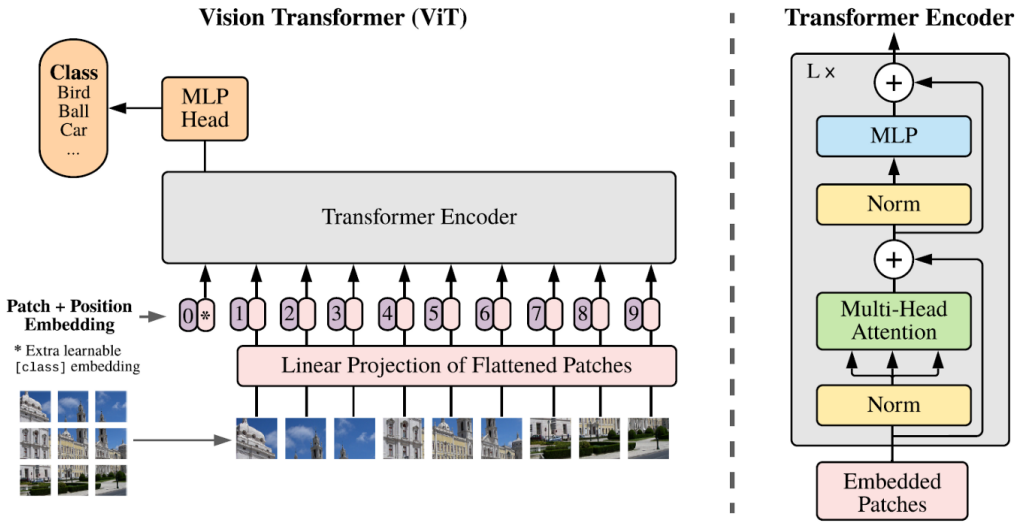

Fig. 2.10: A high-level view of ViT. An image is divided in non-overlapping patches which are linearly projected and fed to the encoder in order to compute the feature vector. The encoder is a Transformer network as described in [Vas+17]. The class ([CLS]) embedding is fed to an MLP in order to be classified. Figure taken from [Dos+21].

locality and receptive field concepts that exist in CNNs, are also present in GCNs as we are encoding information using the local neighborhood of the nodes. GCNs are mainly used to produce node embeddings that can be randomly initialized or calculated from another model, depending on the nodes' modality (e.g., text, image etc.) [Kot20]. These embeddings can then be used to solve various graph-related tasks such as Node Classification and Link Prediction. GCNs aggregate the embeddings of the neighbors of a node by applying a propagation function (e.g., sum, mean etc.) on them:

$$\boldsymbol{h}_{N(v)}^{k} = \text{AGGREGATE}_k \left(\{ \boldsymbol{h}_u^{k-1}, \forall u \in N(v) \} \right) \tag{2.16}$$

where $\boldsymbol{h}_{N(v)}^{k}$ is the summarized embedding of the neighborhood $N(v)$ of node v in the k-th layer of the network.

Aggregating the embeddings of only local-neighborhoods, makes GCNs scalable and fast for large graphs and multiple iterations of training allow the embeddings to characterize a (more) global neighborhood [GF18]. The output of a GCN's layer $l + 1$ (when the *mean* aggregation function is used) can be written in matrix form as:

$$H^{l+1} = f(H^l W_0^l + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} H^l W_1^l) \tag{2.17}$$

where A is the adjacency matrix of the graph, $W_0^l \in \mathbb{R}^{d^{l+1} \times d^l}$ and $W_1^l \in \mathbb{R}^{d^{l+1} \times d^l}$ are learnable parameter matrices and D is the degree matrix which is used for normalizing A in order to account for the degree of each node and in effect avoid large numbers in the embeddings of nodes with large degrees. W_0^l is the weight matrix for self-loops and W_1^l is the weight matrix for the set of neighbors of i, N_i. Using a separate weight matrix for modeling self-loops can lead to over-fitting in cases where there are nodes with very few neighbors or generally, when the graph has very few nodes [KW17]. To address this, we can add the identity matrix I to A and use a single weight matrix. In essence, we add self-loops to the adjacency matrix A and Eq. 2.17 becomes:

$$H^{l+1} = f(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^l W_1^l) \tag{2.18}$$

where $\hat{A} = A + I$ and $\hat{D}_{ii} = \sum_{j \in N_i} \hat{A}_{ij}$

All these aforementioned architectures as well as certain novel normalization and regularization techniques that have been proposed during the last decade, such as Dropout [Sri+14], Batch/Layer/Group Normalization [BKH16; IS15; WH18] and Residual Connections [He+16], have allowed deep learning to make a great impact in the field of AI, achieving incredible results.

2.2 Related Work

Current research approaches on medical image classification heavily revolve around ***transfer learning***, a method that allows us to exploit knowledge from one setting to improve generalization to another one. Transfer learning is particularly useful when one setting has much more data than the other [GBC16]. In general, transfer learning is a two-step process where we first train a high-capacity model using a very large dataset. Essentially, given the large amount of data, we provide the model with general knowledge about a domain. This step is also known as the pre-training step. Over the last years, the most common choice for pre-training in the CV field is the use of the ImageNet dataset [Den+09], a large general-purpose corpus that contains images of various animals and objects organized according to WordNet's hierarchy. ImageNet is a labeled dataset, meaning that its images are labeled with their true category. Thus, supervised pre-training can be performed. Obviously, ImageNet can also be used for unsupervised (or self-supervised) pre-training by using only the images and discarding the labels. By pre-training, we hope to

reach a good initial point for building a model for our specific task (also called downstream task). The second step is called fine-tuning and involves the process of using a classifier (an MLP or a linear logistic regression classifier) and train it on a target dataset using the features of the pre-trained network. We can either fix ('freeze') the parameters of the pre-trained network or further train them along with the parameters of the classifier that is added on top. Transfer learning can prove extremely useful when we deal with small target datasets. In such cases, the general knowledge that the backbone image encoder has obtained can help it generalize better and be more robust to over-fitting.

Despite the large difference between natural and medical images, image classification models pre-trained on ImageNet (and fine-tuned on the target medical dataset) have achieved results close to predictions from clinicians [EKN17]. Following this paradigm, CheXnet [Raj+17] that uses a DenseNet-121 [Hua+17] image encoder pre-trained on ImageNet and fine-tuned on the ChestX-ray 14 dataset [Wan+17] to classify chest X-rays to 14 different was proposed. Using variations of CheXnet, Kougia et al. [KPA19b; KPA20] as well as Karatzas et al. [Kar+20] managed to win the 1st position at the ImageCLEFmed concept detection task on 2019 [Ion+19] and 2020 [Ion+20] respectively. They used a DenseNet-121 pre-trained on ImageNet as the image encoder along with a Feed-Forward Neural Network (FFNN) as the classifier. Their work also included ensembles of different instances of these models.

ImageNet is also used as a benchmark dataset for image classification models. From an architectural point of view, most top-performing approaches rely on huge ViT models [Zha+22; Dos+21; Din+22; Tu+22; Li+22c] that are pre-trained in very large image datasets. This showcases that ViTs are the current trend in CV research. Another line of work consists of the use of hybrid CNN-ViT [Xu+21; Dai+21] that have proven competitive. The top-performing CNN is an EfficientNet-L2 [TL19] when iteratively trained with a student-teacher approach [Pha+21] (a form of knowledge distillation [HVD15] that utilizes large unlabeled datasets).

A more recent advancement in the CV field, involves the use of ***self-supervised learning***. Self-supervised learning is a form of unsupervised learning that is used to learn representations of the data without the use of labels by solving a pre-training task. As such, self-supervised methods allow us to (pre-)train networks using large datasets that do not require costly human annotation. Self-supervised learning has gained a lot of attention both in CV and NLP [Dev+19] as it performs comparably with supervised learning, creates robust representations and it is affordable as it does not require labeled data. However, self-supervision algorithms generally require large training times. Several pre-training tasks have been proposed [DGE15; NF16; Wu+18] and the most successful until now is the instance discrimination task in which joint embedding architectures are usually employed. In this setting, visual encoders (with shared weights) are trained to project visually similar images close together in the embedding space and dissimilar images far away. This form

of self-supervised learning is called contrastive learning [Che+20a; He+20; Dwi+21] as the network is trained to 'contrast' instances. This is achieved with the use of a contrastive loss function [OLV18; Soh16] which calculates the similarity between different image samples (see Eq. 2.19).

$$\mathcal{L}_{NCE}^{q} = -\log \frac{\exp(\langle q, k_{+} \rangle / \tau)}{\sum_{j=0}^{N} \exp(\langle q, k_{j} \rangle / \tau)} \tag{2.19}$$

where $\langle \cdot, \cdot \rangle$ denotes the cosine similarity $\langle x, y \rangle = \frac{x^T y}{||x|| ||y||}$, τ is a temperature hyper-parameter that scales the result, q is the query image and k_{+} is the similar (positive) image we try to bring close to q in the embedding space. The denominator contains the dissimilar (negative) images that we try to move away from q. While contrastive methods yield satisfactory performance, they require large amounts of contrastive pairs in order to work well and avoid representation collapse (i.e., the encoders produce constant or non-informative embeddings). These pairs can be sampled from a memory bank [He+20], or given by the current (large) batch of data [Che+20a], with a significant memory requirement.

There are several other self-supervised methods that fall under the instance descrimination task and do not rely on contrastive samples (i.e., they do not rely on negative pairs) in order to prevent representation collapse. Several methods have employed clustering of samples based on a similarity measure [Car+18; Car+20] and can be viewed as contrastive learning at the level of clusters (still requiring a lot of negative comparisons to work well), distillation with the use of a momentum encoder or stop-gradient operations in one of the branches [Gri+20; CH21] to avoid the collapse, as well as information maximization and decorrelation between the content of the produced embeddings [Erm+21; Zbo+21] that tries to prevent an informational collapse. These approaches generally utilize variations of standard losses such as cross-entropy (see Eq. 2.8) or MSE as

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y - \hat{y})^2 \tag{2.20}$$

where y: gold, $\hat{y}$: predicted

Additionally, due to the rise of ViTs, several pre-training tasks that are specific to the architecture have been proposed. An example is the masked image modeling task [Bao+22; Xie+22], where random visual tokens of the input image are masked and the network tries to predict them. It was proposed following BERT's [Dev+19] great success in the NLP field which uses masked language modeling as a pre-training task (i.e., predicting masked words in sentences). A similar line of work is masked autoencoders [He+22] that try to

predict masked tokens through an encoder-decoder architecture. Furthermore, ViTs have been used with success in architecture-agnostic self-supervised tasks [CXH21; Car+21; Li+22a], producing competitive results. Last but not least, current research also includes cross-modal self-supervised learning using both images and text. These works usually (but not exclusively [Lu+19]) follow a contrastive pre-training approach where we try to embed the image and its true caption close together and embed (different) random captions irrelevant to the image far away [Yao+22; Li+22b; Rad+21; Yu+22].

Self-supervised learning can be also applied to graph-structured data in order to learn node representations in an unsupervised manner. Unsupervised node representation methods utilize the graph's structure. Most of these methods [PRS14; GL16; HYL17a] are based on random walks of the graph which accumulate context from further neighbors and capture higher order dependencies between the nodes and not only information from first order neighbors that may not always be sufficient. For instance, NODE2VEC [GL16], one of the most, if not the most, well-known random walk method, generates a fixed number w of random walks with length l from each node u of the graph. However, by visiting very far neighbors, we can end up aggregating all the node embeddings, resulting in over-smoothing (losing the context of the 'neighborhoods'). Due to this problem, NODE2VEC uses two hyper-parameters p and q to adjust the probability of creating low-order and high-order neighborhoods and let the random walks switch between Breadth-First search (BFS) and Depth-First search (DFS). During training, NODE2VEC applies WORD2VEC's SKIPGRAM paradigm which maximizes the predicted probability $p(u|v)$ of observing the actual neighbors $u \in N(v)$ of each node v of the graph in a random walk R, where $N(v)$ is the neighborhood of v. $p(u|v)$ is calculated using the dot product of the node embeddings z_u and z_v of u and v respectively and are produced by an embedding look-up table. Equivalently, the model minimizes:

$$\mathcal{L}_{N2V} = -\sum_{u \in V} \sum_{v \in N_R(u)} \log(p(v|u)) = -\sum_{u \in V} \sum_{v \in N_R(u)} \log\left(\frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)}\right) \quad (2.21)$$

In practice, negative sampling is used for large graphs instead of the computationally inefficient *softmax* operation. Through training, the look-up embedding matrix $\boldsymbol{Z}$, which contains an embedding for every node u in the training dataset, is learned.

GRAPHSAGE [HYL17a] advances the idea of NODE2VEC by replacing $\boldsymbol{Z}$ with a deep graph encoder (e.g., a GCN) providing inductive representation learning (i.e., embedding calculation for unseen nodes). The encoder function is then trained using a random walk-based objective that encourages nearby nodes to have similar representations, while enforcing the representations of disparate nodes to be distinct:

$$\mathcal{L}_{GS}(z_u) = -\log(\sigma(z_u^T z_v)) - Q\mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-z_u^T z_{v_n})), \quad \forall u \in V \tag{2.22}$$

where v is a node that co-occurs near u on a fixed-length random walk, σ is the sigmoid function (Eq. 2.6) and P_n is a negative sampling distribution with Q denoting the number of negative samples.

In another line of work on unsupervised node representation learning, DEEP GRAPH INFOMAX (DGI) [Vel+19] aims at maximizing the mutual information between patch representations and high-level summaries of $\mathcal{G}$ without relying on random walk objectives. A patch representation denotes a vector $\vec{h}_u$ which summarizes a patch of the graph centered around node u rather than just the node itself. The representation is again produced by an deep encoder like a GCN. DGI's objective is to learn a node encoder $\mathcal{E}$ which will produce high-level representations for each node u. To do so, it seeks to obtain local representations that capture the global information content of the entire graph, represented by a summary vector $\vec{s}$. $\vec{s}$ is calculated by a readout function $\mathcal{R} : \vec{s} = \mathcal{R}\left(\{\vec{h}_u | u \in \mathcal{G}\}\right)$. For instance, $\mathcal{R}$ could calculate the sum or the average of the node embeddings of the graph. In order to maximize the local mutual information, a discriminator $\mathcal{D}$ is employed to decide if node u should be contained within the summary $\vec{s}$. $\mathcal{D}\left(\vec{h}_u, \vec{s}\right)$ represents the probability score assigned to this patch-summary pair and should be higher for patches contained within $\vec{s}$. An example for $\mathcal{D}$ is the cosine similarity. Additionally, negative samples for $\mathcal{D}$ are provided by pairing the summary $\vec{s}$ from $\mathcal{G}$ with representations $\vec{\hat{h}}_u$ of an alternative graph $\hat{\mathcal{G}}$. $\hat{\mathcal{G}}$ can be obtained from another graph in multiple-graphs dataset. For a single graph, a corruption function $\mathcal{C}$ is used as $\hat{\mathcal{G}} = \mathcal{C}(\mathcal{G})$. An example could be a random row-wise shuffling of the nodes of $\mathcal{G}$. Finally, the loss function to be minimized is a noise-contrastive type objective combined with a standard BCE loss between the positive and the negative examples which maximizes mutual information between $\vec{h_u}$ and $\vec{s}$:

$$\mathcal{L}_D = \frac{1}{N+M}\left(\sum_{i=1}^{N} \mathbb{E}_{\mathcal{G}}\left[\log\left(\mathcal{D}\left(\vec{h}_u, \vec{s}\right)\right)\right] + \sum_{j=1}^{M} \mathbb{E}_{\hat{\mathcal{G}}}\left[\log\left(1 - \mathcal{D}\left(\vec{\hat{h}}_u, \vec{s}\right)\right)\right]\right) \tag{2.23}$$

Due to the scarcity of human-annotated medical imagery as well as the difficulty of acquiring new labeled data, self-supervised learning can be used in order to create more domain-related image representations compared to those created by ImageNet pre-training by leveraging unlabeled medical data that is more easily accessible. Azizi et al. [Azi+21] used SimCLR [Che+20a] and the CheXpert dataset (without its labels) [Irv+19] for contrastive pre-training and reported big improvements compared to supervised pre-training using ImageNet. Furthermore, they proposed an algorithm that made use of exam cases found in medical datasets for contrastive pre-training. Similar to this work, Sowrirajan et

al. [Sow+21] used MoCo [He+20] for pre-training in order to improve Chest X-ray [Irv+19] classification performance. Another interesting work as well as the predecessor of CLIP (Contrastive Language–Image Pre-training) [Rad+21], is ConVIRT [Zha+21], a cross-modal contrastive learning pipeline where an image and a text encoder are jointly trained to contrast images with text reports as previously described. The objective is to maximize the agreement between true image-text pairs versus random pairs between the image and text modalities using a bidirectional contrastive objective. [Zha+21] reported improvements compared to ImageNet pre-training and image-only contrastive pre-training algorithms in medical datasets. The experiments were made using the MIMIC-CXR dataset [Joh+19] for the contrastive pre-training.

Based on the intuition that patients with similar conditions exhibit similar medical screening results, retrieval-based methods have also proven useful in medical image classification. In 2017, the k-NN [FH89] algorithm was used by Valavanis and Stathopoulos [VS17] in order to compete in the ImageCLEFmed 2017 concept detection task [Eic+17] where they acquired the 2nd position. However, they did not use a neural retrieval approach as they experimented with methods such as bag of colors (BoC) and bag of visual words (BoVW) to represent the images. In 2019, Kougia et al. [KPA19b; KPA20] acquired the 3rd position at the ImageCLEFmed concept detection task of the corresponding year [Ion+19] by utilizing a neural retrieval method. They used a DenseNet-121 [Hua+17] encoder pre-trained on ImageNet [Den+09] and fine-tuned it on the competition's data. Then, they performed a k-NN search to fetch the k most similar training images for each test image using scaled-dot product similarity on the image embeddings and then a majority voting scheme for the final decision. Sonker et. al [Son+20] acquired the 2nd position in the 2020 version of the task [Ion+20] using k-NN on ResNet-101 [He+16] embeddings and experimented with several schemes for the final label assignment. In 2021, AUEB's NLP group managed to acquire the 1st position at the 2021 ImageCLEFmed concept detection task [Pel+21] using an ensemble (with majority voting) of 1-NN neural retrieval models [Cha+21] that also utilized several CNN image encoders like EfficientNet [TL19], DenseNet [Hua+17] and ResNet [He+16].

Additionally, several works, albeit mainly in the NLP field, use neural retrieval methods in order to 'augment' the classification process. They use the representation of the neighbors that were retrieved, usually combining them with the query (test) instance before the final classification step [SWD22; Zhe+21; Sar+22].

3 Implemented systems

In this section, we will describe the methods that were implemented for the work of this thesis. They all used image encoders based on deep neural networks. We experimented with several architectures as well as different methods that aim to address the challenges of multi-label image classification. In the end, the methods were combined with either a linear classifier or a retrieval component.

3.1 Deep Learning-based classifiers

We will first present the systems we developed using image encoders combined with a classification head network.

3.1.1 Baseline: Image encoder + FFNN

This system follows our previous work [KPA20; Kar+20; Cha+21; KPA19b] and acts as the base for most of our experiments. In its simplest form, it uses an image encoder and an FFNN on top whose last layer is comprised of C neurons with sigmoid activations and acts as the classifier. C is the number of all the possible labels in the dataset. We experimented with several network architectures ranging from CNNs [TL19; He+16; Hua+17] to ViTs [Dos+21; Liu+21] as well as hybrids of the two [Dai+21]. All the encoders were pre-trained on ImageNet [Den+09] and we fine-tuned them on the target medical dataset.

We extracted image embeddings (feature vectors) using the last convolutional layer of the image encoder, followed by a Generalized-Mean (GeM) global pooling layer [RTC19]. In essence, given the input image, the output of the last convolutional layer L_f of the encoder is a 3D tensor $\boldsymbol{X}$ of size $W \times H \times K$, where K is the number of feature maps (channels) of L_f. Let $\boldsymbol{X}_k$ be the k-th feature map of dimensions $W \times H$ with $k \in \{1, 2, \ldots, K\}$. The GeM (global) pooling layer takes X as an input and returns a vector $v \in \mathbb{R}^{1 \times K}$ as output:

$$\boldsymbol{v} = [\tilde{v}_1 \ldots \tilde{v}_k \ldots \tilde{v}_K]^T, \; \tilde{v}_k = \left(\frac{1}{|\boldsymbol{X}_k|} \sum_{x \in \boldsymbol{X}_k} x^{p_k} \right)^{\frac{1}{p_k}} \tag{3.1}$$

where x is the value of the corresponding pixel of the k-th feature map X_k. v is the produced image embedding. Parameter p_k can be trained along with the network or be

manually set. There is a different parameter p_k per feature map so we train K parameters but it is also possible to have a single parameter p for all feature maps (where $p_k = p, \forall k \in \{1, 2, \ldots, K\}$). GeM is in fact a generalization of (global) max and average pooling which are defined as:

$$\text{MAX: } \boldsymbol{v} = [\tilde{v}_1 \ldots \tilde{v}_k \ldots \tilde{v}_K]^T, \ \tilde{v}_k = \max_{x \in \boldsymbol{X}_k} x \tag{3.2}$$

$$\text{AVG: } \boldsymbol{v} = [\tilde{v}_1 \ldots \tilde{v}_k \ldots \tilde{v}_K]^T, \ \tilde{v}_k = \frac{1}{|\boldsymbol{X}_k|} \sum_{x \in \boldsymbol{X}_k} x \tag{3.3}$$

The acquired image embeddings were then passed through the FFNN to produce the output logits and then the probabilities through the sigmoid activation (one probability per label).

In a fully-supervised setting, we trained a model (comprised of the backbone encoder and the FFNN) using labeled images that were rescaled to 224×224 and normalized with the mean and standard deviation of ImageNet's images in order to match the pre-training settings of the encoder. At inference, we predicted the labels for each image using the probabilities, as estimated by the classifier. We assigned a label to the test image if its output probability exceeded a threshold t. The final predicted label set consisted of the labels for which we had: $p_{\text{label}} \geq t$. Additionally, instead of using the default value $t = 0.5$ that is used in binary decision problems, we tuned t using the validation data.

3.1.2 Loss functions

In scenarios where large class imbalance is encountered, BCE which is the simplest loss that can be used for multi-label tasks, is usually 'overwhelmed'. When there are labels that appear rarely in the dataset (i.e., the gold truth binary vector y of most examples contains a lot of zeros), the loss tends to make the classifier classify most labels as negatives (even if they are not) in order to be decreased. There are a couple of losses that we experimented with that try to mitigate this issue. The first one is the *Focal (FL)* loss [Lin+17] which (for a label c) can be defined as:

$$\mathcal{L}_{FL_c} = -\left(y_c \alpha (1 - p_c)^\gamma \log(p_c) + (1 - y_c)(1 - \alpha) p_c^\gamma \log(1 - p_c)\right) \tag{3.4}$$

where p_c is the prediction of the model and α, γ are (tunable) hyper-parameters that control the behavior of the output loss value. $\alpha \in [0, 1]$ is a weighting factor which balances the importance of positive and negative examples. $\gamma \in [0, \infty)$ differentiates between easy and hard examples by down-weighting easy examples and thus focusing the training process

on hard negatives. The higher the value of γ, the lower the overall loss for well-classified examples, so the model focuses towards hard-to-classify examples.

The second loss that we experimented with is the *asymmetric (ASL)* loss [Bar+21]. ASL is a variant of FL with two basic modifications. Firstly, it decouples the focusing level of positive and negative samples by using two different focusing hyper-parameters, namely γ_+ and γ_-. This variation aims to avoid the elimination of the contribution of positive examples (a phenomenon encountered when setting $\gamma \gg 0$ in Eq. 3.4). Secondly, probability shifting (Eq. 3.5) is applied to the negative loss part in order to discard negative samples when their probability is very low.

$$p_{c_s} = \max(p_c - m, 0) \tag{3.5}$$

where probability $m \geq 0$ is a hyper-parameter. With probability shifting, the negative loss part gives a value of 0 when $p < m$. The ASL loss can be defined as:

$$\mathcal{L}_{ASL_c} = -\left(y_c(1-p_c)^{\gamma_+}\log(p_c) + (1-y_c)p_{c_s}^{\gamma_-}\log(1-p_{c_s})\right) \tag{3.6}$$

The final loss that we experimented with followed a different direction and aimed to directly optimize F_1, which was also the evaluation metric that was used in our experiments [Ion+22]. In order to make it differentiable, we modified the calculation of F_1 by computing true positives (TP), false positives (FP), and false negatives (FN) using the output probabilities of the model, without applying any threshold to round them to binary decisions [KNS20]. This (differentiable) version is referred to as the 'soft F_1' score (sF_1).[1] In practice, $1 - sF_1$ is used for minimization. The loss can be defined as:

$$\mathcal{L}_{F_1} = 1 - sF_1 = 1 - 2 \cdot \frac{s\text{Precision} \cdot s\text{Recall}}{s\text{Precision} + s\text{Recall}} \tag{3.7}$$

where

$$s\text{Precision} = \frac{sTP}{sTP + sFP} = \frac{\sum_c y_c \cdot \hat{y}_c}{\sum_c y_c \cdot \hat{y}_c + \sum_c (1-y_c) \cdot \hat{y}_c} \tag{3.8}$$

and

[1] Also see: https://towardsdatascience.com/the-unknown-benefits-of-using-a-soft-f1-loss-in-classification-systems-753902c0105d

$$s\text{Recall} = \frac{sTP}{sTP + sFN} = \frac{\sum_c y_c \cdot \hat{y}_c}{\sum_c y_c \cdot \hat{y}_c + \sum_c y_c \cdot (1 - \hat{y}_c)} \tag{3.9}$$

Additionally, due to the fact that the values of $\mathcal{L}_{F_1}$ lie in $[0, 1]$, we combined $\mathcal{L}_{F_1}$ with $\mathcal{L}_{BCE}$ in order to avoid constraining the search space. The combination can be either summation or multiplication of the two loss functions.

3.1.3 Optimization

In order to minimize the loss and optimize the parameters of the model, we used a variant of SGD (Eq. 2.4) called Adam [KB15] in most of our experiments. Adam belongs to a family of algorithms [DHS11; KB15] that modify the learning rate during the training process in order to allow faster convergence by taking smaller and smaller steps as the training continues.[2] Loosely speaking, the modification of the learning rate is done by scaling it using the values of the gradients and different hyper-parameters. In the case of Adam, we get the following weights' update formula:

$$\boldsymbol{w}_{t+1} \coloneqq \boldsymbol{w}_t - \frac{\eta \cdot \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.10}$$

where $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$, $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$ with $m_t = (1-\beta_1)g_t + \beta_1 m_{t-1}$ and $v_t = (1-\beta_2)g_t^2 + \beta_2 v_{t-1}$. β_1, β_2 are hyper-parameters, ϵ is a small term preventing division by zero and t denotes the current time-step of training. g is the value of the gradients of the mini-batch in time-step t and is defined as:[3]

$$g = \frac{1}{M} \sum_{i=1}^{M} \nabla \mathcal{L}_i(\boldsymbol{w}_t) \tag{3.11}$$

Additionally, we experimented with another optimization algorithm called Sharpness-Aware Minimization (SAM) [For+21] aiming to achieve better generalization performance. SAM is a two-step variation of SGD which tries to find parameter values whose neighborhoods have uniformly low training loss, rather than seeking parameter values that have a low training loss individually. The algorithm seeks neighborhoods having both low loss and low curvature, taking the sharpness of the loss landscape into consideration. Formally, the loss that SAM aims to minimize (omitting the loss type, time-step and possible regularizers for simplicity) is defined as:

[2] There is also RMSprop which was introduced by Geoffrey Hinton in: https://www.cs.toronto.edu/ tijmen/csc321/slides/lecture_slides_lec6.pdf

[3] Also see Eq. 2.4

$$\mathcal{L}^{SAM}(\boldsymbol{w}) = \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \mathcal{L}(\boldsymbol{w} + \boldsymbol{\epsilon}) \tag{3.12}$$

where $\boldsymbol{w} + \boldsymbol{\epsilon}$ is a perturbation of the model's parameters (can be roughly considered as the neighborhood) and $\rho \geq 0$ is a hyper-parameter that denotes the size of the neighborhood. Furthermore, $\mathcal{L}^{SAM}(\boldsymbol{w}) - \mathcal{L}(\boldsymbol{w})$ captures the sharpness of $\mathcal{L}$ at $\boldsymbol{w}$ by measuring how quickly the training loss can be increased by moving from $\boldsymbol{w}$ to $\boldsymbol{w} + \boldsymbol{\epsilon}$.

In order to minimize $\mathcal{L}^{SAM}(\boldsymbol{w})$, we can derive an approximation of $\nabla\mathcal{L}^{SAM}(\boldsymbol{w})$ as:

$$\nabla\mathcal{L}^{SAM}(\boldsymbol{w}) \approx \nabla\mathcal{L}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})} \tag{3.13}$$

where

$$\hat{\boldsymbol{\epsilon}}(\boldsymbol{w}) = \rho \text{sign}(g) \frac{|g|^{q-1}}{\left(\|g\|_q^q\right)^{\frac{1}{p}}} \tag{3.14}$$

$\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})$ solves the first-order Taylor expansion of $\mathcal{L}(\boldsymbol{w} + \boldsymbol{\epsilon})$ w.r.t to $\boldsymbol{\epsilon}$ around 0.[4] Moreover, $p = q = 2$ is considered optimal. The parameter update is illustrated in Fig. 3.1. Note that the SAM update can be also combined with adaptive learning rate algorithms such as Adam.

Besides SAM, we also conducted experiments using Gradient Centralization (GC) [Yon+20] aiming to improve generalization. GC essentially centralizes the gradient vectors of weight matrices to have zero mean and acts as a regularization technique. Considering a weight vector $\boldsymbol{w}_i$ of a weight matrix $\boldsymbol{W}$ with gradient $\nabla_{\boldsymbol{w}_i}\mathcal{L}$, we can initially define GC as:

$$\boldsymbol{\Phi}_{GC}(\nabla_{\boldsymbol{w}_i}\mathcal{L}) = \nabla_{\boldsymbol{w}_i}\mathcal{L} - \frac{1}{K}\sum_{j=1}^{K} \nabla_{\boldsymbol{w}_{i,j}}\mathcal{L} \tag{3.15}$$

The above formula can be rewritten in matrix formulation as:

$$\boldsymbol{\Phi}_{GC}(\nabla_{\boldsymbol{W}}\mathcal{L}) = \boldsymbol{P}\nabla_{\boldsymbol{W}}\mathcal{L}, \quad \boldsymbol{P} = \boldsymbol{I} - \boldsymbol{e}\boldsymbol{e}^T \tag{3.16}$$

[4] Thorough details on the derivation of the approximation can be found at [For+21].

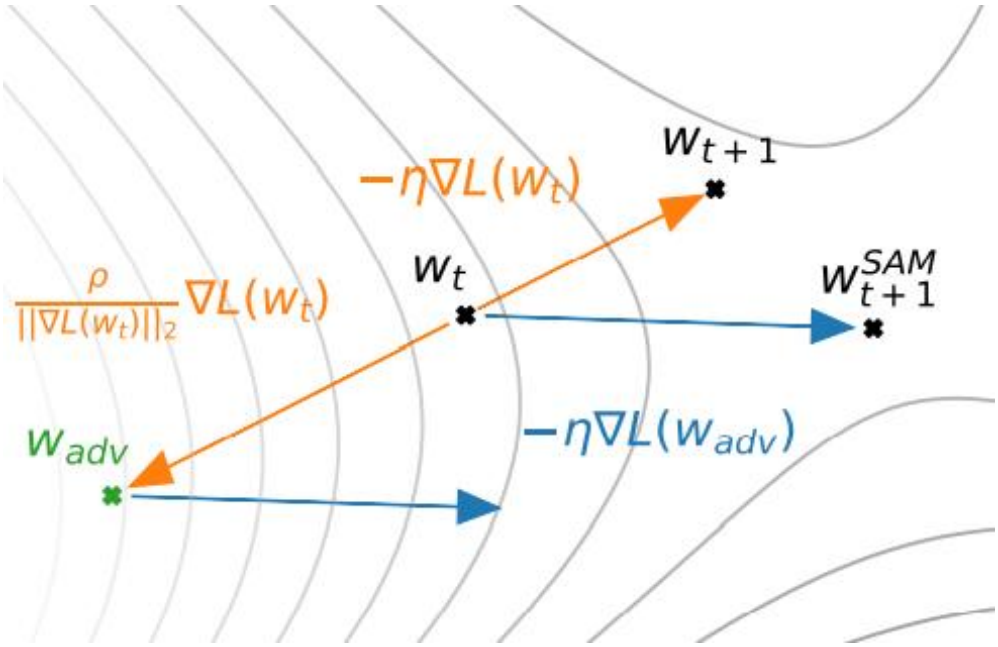

Fig. 3.1: Illustration of the SAM parameter update. Vanilla SGD update is denoted by $\boldsymbol{w}_{t+1}$ while final SAM update is denoted by $\boldsymbol{w}_{t+1}^{SAM}$. Figure taken from [For+21].

where $\boldsymbol{I} \in \mathbb{R}^{K\times K}$ is the identity matrix and $\boldsymbol{e} = \frac{1}{\sqrt{K}}\mathbb{1} \in \mathbb{R}^K$ with $\mathbb{1}$ being the unit vector. $\boldsymbol{P}$ is in fact an orthogonal projection ($\boldsymbol{P}^2 = \boldsymbol{P} = \boldsymbol{P}^T$) and $\boldsymbol{P}\nabla_{\boldsymbol{W}}\mathcal{L}$ is the gradient projected on the hyperplane $\boldsymbol{e}^T(\boldsymbol{w} - \boldsymbol{w}') = 0$ and is used to update the weights (see Fig. 3.2). In fact, the regularization effect results from $\boldsymbol{e}^T(\boldsymbol{w} - \boldsymbol{w}') = 0 \Rightarrow \boldsymbol{e}^T\boldsymbol{w} = \boldsymbol{e}^T\boldsymbol{w}'$ which eventually gives us that $\boldsymbol{e}^T\boldsymbol{w}$ is constant. This provides us with a constrained objective function (w.r.t to one weight vector $\boldsymbol{w}$) which regularizes the search space of $\boldsymbol{w}$:[5]

$$\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}), \quad \boldsymbol{e}^T(\boldsymbol{w} - \boldsymbol{w}^0) = 0 \tag{3.17}$$

Finally, we also experimented with Bayesian Optimization [SLA12] in order to perform hyper-parameter tuning. More particularly, instead of tuning a single decision threshold value (the same for all labels - see Section 3.1.1), we tried to tune a different threshold value for each label. Due to the large size of the label set and the large search space for each threshold, we employed Bayesian Optimization instead of a full parameter sweep. Bayesian Optimization is an iterative method that attempts to find the global optimum of an (expensive) objective function $\boldsymbol{f}$ in a minimum number of steps. It constructs a posterior distribution of functions (a *Gaussian Process* - GP) that describes $\boldsymbol{f}$. As the number of samples grows with each iteration, the posterior distribution improves, and the algorithm becomes more certain of which regions in parameter space are worth exploring and which are not. At each iteration, the GP is fitted to the known samples, and the

[5] More details as well as proofs for the mathematical formulae can be found at [Yon+20].

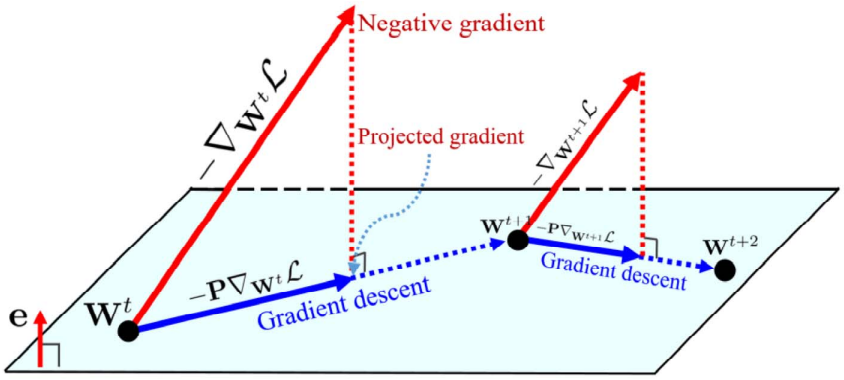

Fig. 3.2: Geometrical interpretation of GC. The projected gradient is used to update the weights. Figure taken from [Yon+20].

posterior distribution, combined with an exploration strategy, also called an acquisition function (e.g., Expected Improvement - EI), are used to determine the next point that should be explored. The acquisition function is used to determine the next sampling point by optimizing it over the GP. EI is defined as:

$$EI(x) = \mathbb{E}\max(\boldsymbol{f}(x) - \boldsymbol{f}(x^+), 0) \tag{3.18}$$

and can evaluated analytically under the GP model.[6] $\boldsymbol{f}(x^+)$ is the value of the best sample and x^+ is the 'location' of that sample. For our experimentations, we used our F_1 evaluation metric as $\boldsymbol{f}$, aiming at maximize it.

3.1.4 Ensembles

In an effort to improve the performance of our models, we experimented with ensembling methods in order to combine the predictions of separate instances. We experimented with three different methods. Following [Kar+20; Cha+21], the first one regarded set

[6] Also see: http://krasserm.github.io/2018/03/21/bayesian-optimization/

operations where we combined the k instances using the UNION and the INTERSECTION of their predicted label sets.

$$\text{UNION: } \hat{\mathcal{P}} = \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_k \tag{3.19}$$

$$\text{INTERSECTION: } \hat{\mathcal{P}} = \mathcal{P}_1 \cap \ldots \cap \mathcal{P}_k \tag{3.20}$$

The second method followed a simple majority voting rule where a label was assigned to a test image if it was predicted by the majority of the systems. The third (and more elaborate) method was the creation of '*model soups*' [Wor+22]. '*model soups*' use an average of the weights of the model instances. Similar to [Wor+22], we employed '*greedy soup*' algorithm where the ensemble is constructed by sequentially adding each instance in the 'soup', and only keeping it if the performance on the held out validation set improves. The algorithm is defined as:[7]

Algorithm 1 GreedySoup

Input: Instances list $\boldsymbol{\theta} = \{\theta_1, \ldots \theta_k\}$ (optionally sorted ValAcc(θ_i) $\geq$ ValAcc(θ_{i+1}))

if $\boldsymbol{\theta}$ is sorted **then**
    soup $\leftarrow \theta_1$
    $j^* \leftarrow 1$
else
    $j^* \leftarrow \arg\max\{\text{ValAcc}(\boldsymbol{\theta})\}$
    soup $\leftarrow \theta_{j^*}$
for $i \in \{1, \ldots, j^* - 1, j^* + 1, \ldots, k\}$ **do**
    if ValAcc(average(soup $\cup\{\theta_i\}$)) $\geq$ ValAcc(average(soup)) **then**
        soup $\leftarrow$ soup $\cup\{\theta_i\}$
return soup

3.1.5 Self-supervision

We also experimented with self-supervised models in order to compare them with the models trained in a fully-supervised setting. We tested three different methods, each with a different pre-training objective. Namely, we used CLIP [Rad+21], a Vector Quantised-Variational AutoEncoder (VQ-VAE) [OVK17] and a masked AutoEncoder (MAE) [He+22].

CLIP is a model that "efficiently learns visual concepts from natural language supervision" [Rad+21]. It consists of a text encoder and an image encoder that are trained on image-text pairs. Image and text embeddings are mapped to a joint space. Two types of image-text pairs are formed: matching pairs, in which an image is paired up with its corresponding caption, and mismatching pairs, in which an image is paired up with another random caption. The encoders are trained to map the matching pairs to nearby points on this joint space, and mismatching pairs to distant points. Thus, CLIP uses a bidirectional contrastive objective (see Eq. 2.19) between image and text embeddings. We used the pre-trained ViT

[7] Adapted from [Wor+22].

version of CLIP and extracted image embeddings using the last hidden state of the encoder and GeM global pooling.[8] We then used the CLIP embeddings in the training of our model where we concatenated them with the embeddings extracted from the base image encoder, acquiring $\tilde{\boldsymbol{v}}$:

$$\tilde{\boldsymbol{v}} = [\boldsymbol{v}; \boldsymbol{v}_{CLIP}] \tag{3.21}$$

VQ-VAEs (or discrete Variational AutoEncoders - dVAEs) follow the encoder-decoder architecture of traditional AutoEncoders but unlike VAEs, whose latent space is continuous and is sampled from (usually) a Gaussian distribution, they operate on a discrete latent space (where posterior and prior distributions are categorical) and maintain a trainable codebook of size K, $e \in \mathbb{R}^{K \times D}$. The codebook stores continuous embeddings and is used as a look-up table between these embeddings and the encoder outputs. The output of the encoder $z_e(x)$ is passed through the discretization bottleneck followed by mapping onto the nearest element of e:

$$z_q(x) = e_k, \quad k = \arg\max_j \|z_e(x) - e_j\|_2 \tag{3.22}$$

The corresponding embeddings of the codebook are then fed to the decoder, which is trained to generate reconstructed samples (see Fig. 3.3).

We essentially want to maximize the log-likelihood of the data distribution (also called evidence) $\log p_\theta(\boldsymbol{x})$ but this distribution is defined by an intractable integral:

$$p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}, \boldsymbol{z})\, dz = \int p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_\theta(\boldsymbol{z})\, dz \tag{3.23}$$

Because of this intractability, we aim to *approximate* $p_\theta(\boldsymbol{x})$ using a 'simple' distribution q_ϕ. This can be done by using the Kullback–Leibler divergence [KL51] (which tries to make p_θ and q_ϕ approach each other) as the training objective:

$$D_{KL} = (P\|Q) = \sum_i \log\left(\frac{P(i)}{Q(i)}\right) \tag{3.24}$$

It has been proven that, using mathematical operations, we can derive:

[8] https://huggingface.co/openai/clip-vit-base-patch32

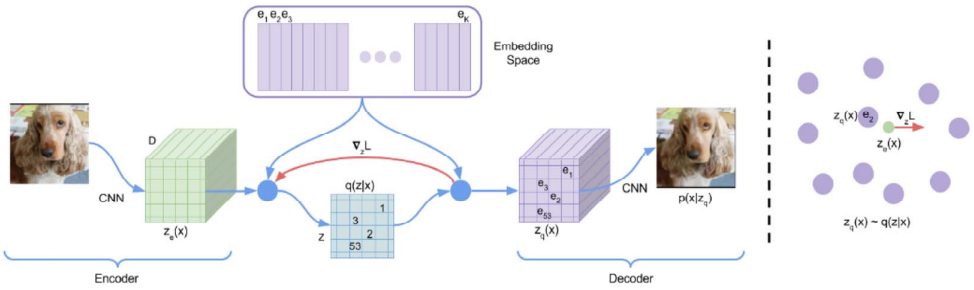

Fig. 3.3: Left: Illustration of the VQ-VAE. Right: Visualisation of the embedding space. Since the quantization process is not differentiable, a straight-through estimator in between the decoder and the encoder is applied, so that the decoder gradients are directly propagated to the encoder. Other techniques such as gumbel softmax relaxation [JGP17] can be used. Figure taken from [OVK17].

$$D_{KL} = \log p_\theta(\boldsymbol{x}) - \text{ELBO} \Rightarrow \tag{3.25}$$
$$\text{ELBO} = \log p_\theta(\boldsymbol{x}) - D_{KL}$$

where ELBO is the evidence lower bound [KW14]:

$$\log p_\theta(\boldsymbol{x}) = \log \int p_\theta(\boldsymbol{x}, \boldsymbol{z})\, dz = \ldots \geq \mathbb{E}_{\boldsymbol{Z}\sim q}\left[\log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{Z})}{q(\boldsymbol{Z})}\right] \tag{3.26}$$

In VAEs, the distributions p_θ, q_ϕ are modeled by neural networks. VAEs (and VQ-VAEs) can be trained using the following objective (derived from the previous equations) [KW14]:

$$\log p_\theta(\boldsymbol{x}) \geq \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}(\log p_\theta(\boldsymbol{x}|\boldsymbol{z})) - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \| p_\theta(\boldsymbol{z})) \tag{3.27}$$

The right hand side of the inequality is the objective. The first term denotes the reconstruction log-likelihood and the second term measures the distance between the prior distribution p_θ and the approximate posterior q_ϕ.

VQ-VAEs lie behind the success of OpenAI's DALL-E [Ram+21] where the latent visual tokens (concatenated with text tokens) are used as the input to an autoregressive Transformer decoder. We experimented with OpenAI's pre-trained dVAE and ViLMedic's [Del+22] dVAE pre-trained on MIMIC [Joh+19] in order to compute visual tokens using the images of our dataset.[9] During the training of our model, we concatenated the tokens with the embeddings extracted from the base image encoder, acquiring $\tilde{\boldsymbol{v}}$:

$$\tilde{\boldsymbol{v}} = [\boldsymbol{v}; \boldsymbol{v}_d] \tag{3.28}$$

As for MAE, it is a simple AutoEncoder system that masks random patches of the input image and aims at reconstructing the missing pixels. The encoder and decoder adopt the ViT architecture which by default requires the input image to be divided into patches. We then mask a portion of the input patches. During pre-training, the encoder is applied only to the visible (non-masked) patches while the full set of encoded and masked tokens is processed by the decoder that reconstructs the original image. The loss objective computes the MSE (see Eq. 2.20) between the reconstructed and original images in the pixel space [He+22]. After pre-training, the encoder can be applied to uncorrupted images for downstream tasks. We used MAE pre-trained on ImageNet and experimented with supervised fine-tuning and further pre-training using our dataset.[10]

3.1.6 Aggregating information from the labels

This family of experiments aimed at exploiting any possible information that the label space could provide. We created a graph $\mathcal{G}$ based on the statistical co-occurrence of the labels and experimented with various methods consisting of GNNs in order to process $\mathcal{G}$ and acquire features which we combined with the image features that stemmed from the image encoder.

The first approach follows the work of [Zha+18] and their DXML (Deep embedding for XML classification) system. Given our dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ where $\boldsymbol{x}_i \in \mathbb{R}^D$ is a computed image embedding and $\boldsymbol{y}_i \in \{0,1\}^{|C|}$ is the multi-hot gold truth vector of image i, we first created the label graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node in $\mathcal{V}$ denotes a label (i.e., $\mathcal{V} = C$) and there is an edge between two labels if they co-appear on at least one training example. Then, an unsupervised node representation method (e.g., NODE2VEC [GL16], DEEP GRAPH

[9] OpenAI's implementation of dVAE: https://github.com/openai/DALL-E
[10] https://huggingface.co/facebook/vit-mae-base

INFOMAX [Vel+19] - see Sec. 2.2) was used to obtain a ℓ-dimensional representation for each node in $\mathcal{G}$, providing us with the label matrix $\boldsymbol{M} \in \mathbb{R}^{\ell \times |C|}$. The feature vector of each $\boldsymbol{y}_i$ was computed as:

$$f_{\boldsymbol{y}_i} = \frac{\boldsymbol{M}\boldsymbol{y}_i}{nnz(\boldsymbol{y}_i)} \tag{3.29}$$

where $nnz(\boldsymbol{y}_i)$ denotes the number of non-zero elements in $\boldsymbol{y}_i$. Then, a non-linear projection network $F_{\boldsymbol{\theta}}$ was trained in order to obtain the mapping from the original feature vector set $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ to embedded feature vector set $f_{\boldsymbol{X}} = F_{\boldsymbol{\theta}}(\boldsymbol{X}) \in \mathbb{R}^{\ell \times N}$ by minimizing the distance between $f_{\boldsymbol{X}}$ and $f_{\boldsymbol{Y}}$. A distance loss was used as an objective. Following [Zha+18], we experimented Huber loss [Hub64]:

$$\mathcal{L}_H = \sum_{i=1}^{N} d(f_{\boldsymbol{x}_i}, f_{\boldsymbol{y}_i}) \tag{3.30}$$

where

$$d(f_{\boldsymbol{x}_i}, f_{\boldsymbol{y}_i}) = \sum_{j}^{\ell} h((f_{\boldsymbol{x}_i})_j, (f_{\boldsymbol{y}_i})_j) \tag{3.31}$$

and

$$h(a,b) = \begin{cases} \frac{1}{2}(a-b)^2, & |a-b| \leq \delta \\ \delta \cdot (a-b| - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \tag{3.32}$$

δ is the point where the loss function changes from a quadratic to linear and is usually set to 1. MSE loss (see Eq. 2.20) can also be used. The trained $F_{\boldsymbol{\theta}}$ can then be used for downstream tasks [Zha+18].

The second approach was based on [CWG19]'s ML-GCN. The overall method is somewhat similar to DXML with differences in the creation of the adjacency matrix $\boldsymbol{A}$ of $\mathcal{G}$ and the node representation method. We initially create a correlation matrix $\boldsymbol{M} \in \mathbb{R}^{|C| \times |C|}$ where $\boldsymbol{M}_{ij}$ denotes the co-occurrences of labels l_i and l_j. We then acquire the conditional probability matrix $\boldsymbol{P}$:

$$\boldsymbol{P}_i = \frac{\boldsymbol{M}_i}{N_i} \tag{3.33}$$

by normalizing $\boldsymbol{M}$ using the occurrences of l_i, N_i, in the training set. $\boldsymbol{P}_{ij}$ equals the probability $P(l_j|l_i)$ of label l_j appearing when label l_i appears. $\boldsymbol{P}$ is then binarized by the rule:

$$\boldsymbol{A}_{ij} = \begin{cases} 0, & \boldsymbol{P}_{ij} < \tau \\ 1, & \boldsymbol{P}_{ij} \geq \tau \end{cases} \tag{3.34}$$

where τ is a threshold value. The binarization is done in order to avoid over-fitting the correlation matrix on the training set.

A GCN [KW17] (with initial node features and an adjacency matrix as inputs) was used in order to process $\mathcal{G}$ and update the nodes' features for every node u (label) in the graph. The initial node features were initialized with contextual embeddings based on the text of each label l_i [Mik+13; Dev+19] (but they can also be randomly initialized). Additionally, in order to avoid the over-smoothing of the produced node embeddings, a re-weighted correlation matrix $\boldsymbol{A'}$ was used:

$$\boldsymbol{A}'_{ij} = \begin{cases} (p/\sum_{\substack{i=1 \\ i \neq j}}^{C} \boldsymbol{A}_{ij}) \times \boldsymbol{A}_{ij}, & i \neq j \\ 1-p, & i = j \end{cases} \tag{3.35}$$

p determines the weights assigned to a node itself and its neighbors. When updating the node feature, there will be a fixed weight for the node itself and the weights for its neighbors will be determined by the neighborhood distribution. When $p \to 1$, the feature of a node itself will not be considered and when $p \to 0$, neighboring information tends to be ignored [CWG19]. The final node embeddings matrix $\boldsymbol{H} \in \mathbb{R}^{|C| \times D}$ was applied to the image representation $\boldsymbol{v} \in \mathbb{R}^D$ (extracted from the input image via the image encoder) using the dot-product operation in order to obtain the prediction probabilities:

$$\hat{\boldsymbol{y}} = \sigma(\boldsymbol{H}\boldsymbol{v}) \tag{3.36}$$

where σ denotes the sigmoid function. The image encoder and the GCN are jointly trained in a supervised setting using the images' labels and an appropriate loss function (e.g., $\mathcal{L}_{BCE}$, $\mathcal{L}_{FL}$, etc.).

The third graph-based approach, called LDGN (Label-specific Dual Graph Neural network), was introduced in [Ma+21] for text classification and constitutes a two-step extension of ML-GCN. In the first step, a label guided attention mechanism is used in order to learn label-specific image representations. That is, given the 3D (dims: $W \times H \times K$) output tensor of the last convolutional layer of the image encoder, we reshape it into a 2D tensor consisting of $wh = W \times H$ image 'patches' (dims: $wh \times K$). Using this tensor and an attention MLP $F_{\boldsymbol{\theta}} \in \mathbb{R}^{D \times |C|}$, the label-specific visual representations matrix $\boldsymbol{V} \in \mathbb{R}^{|C| \times D}$ were calculated:

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_c \\ \vdots \\ \boldsymbol{v}_C \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_{1,1} & \cdots & \boldsymbol{v}_{1,D} \\ \cdots & \cdots & \cdots \\ \boldsymbol{v}_{c,1} & \cdots & \boldsymbol{v}_{c,D} \\ \cdots & \cdots & \cdots \\ \boldsymbol{v}_{C,1} & \cdots & \boldsymbol{v}_{C,D} \end{bmatrix} \tag{3.37}$$

where

$$\boldsymbol{v}_c = \sum_{j=1}^{wh} \alpha_{cj} x_j \tag{3.38}$$

and

$$\alpha_{cj} = softmax(F_{\boldsymbol{\theta}}(x_j)_c; F_{\boldsymbol{\theta}}(x_1)_c, \ldots, F_{\boldsymbol{\theta}}(x_{wh})_c) \tag{3.39}$$

indicates how informative the j-th patch feature vector is for the c-th label. In the second step, a GCN was utilized to learn deep relationships between the label-specific visual representations guided by the statistical label correlations. The conditional probability matrix $\boldsymbol{P}$ as the adjacency matrix $\boldsymbol{A}^s$ and $\boldsymbol{V}$ was used as the inital node representations $\boldsymbol{H}^0$. In this system, instead of binarizing and re-weighting $\boldsymbol{P}$, the resulting node representations $\boldsymbol{H}' \in \mathbb{R}^{|C| \times D'}$ were used to re-learn the interactions among the label-specific representations guided by the information of the reconstructed adjacency matrix $\boldsymbol{A}^R$ [Ma+21]. More specifically, $\boldsymbol{H}'$ was projected using two convolution layers:

$$\boldsymbol{A}^R = \sigma\left(\left(F_{\boldsymbol{W}_1}\left(\boldsymbol{H}'\right)\right)^T \left(F_{\boldsymbol{W}_2}\left(\boldsymbol{H}'\right)\right)\right) \tag{3.40}$$

where $F_{\boldsymbol{W}_1}$, $F_{\boldsymbol{W}_2}$ denote the convolution layers and σ is the sigmoid function. A new GCN was then used to process the reconstructed graph. The network takes $\boldsymbol{H}'$ as inputs, and outputs the final label-specific representations $\boldsymbol{H}'' \in \mathbb{R}^{|C| \times D'}$. The two types of label-specific feature vectors were concatenated and fed to a liner classifier to obtain the prediction probabilities:

$$\hat{\boldsymbol{y}} = \sigma\left(F_{\boldsymbol{\theta}}\left(\left[\boldsymbol{H}'; \boldsymbol{H}''\right]\right)\right) \tag{3.41}$$

Like ML-GCN, LDGN is trained in a supervised manner.

3.1.7 Self-training with Teacher-Student

In order to provide additional examples for under-represented labels, we employed an unlabeled dataset (see Sec. 4.3.6 below) which we silver-labeled and utilized an iterative self-training approach in order to train a Student model using the extra unlabeled data. We followed a variation of the Noisy Student algorithm [Xie+20; FMi+21] where we considered adding the most confident unlabeled samples only if the silver labels contained certain labels that we chose. The modified algorithm can be described as:[11]

Algorithm 2 Self-training with Teacher-Student (for specific labels)

Input: Labeled data: L, Unlabeled data: U, Teacher: F^T, Student: F^S, Number of pseudo-labeled data in an iteration: k, Labels to focus on: Q
Output: A trained Student model F^S

Initialize F^T and train F^T on L
while F^S not good enough & $U \neq \emptyset$ **do**
  Initialize F^S, $L' \leftarrow PriorityList()$
  for $x \in U$ **do**
    Compute prediction probabilities $\hat{y}_x = F^T(x) \in [0,1]^{|C|}$
  for $c \in Q$ **do**
    $L'.insert(\{\hat{y}_x[c], x\}) \quad \forall x \in U)$
    $L' \leftarrow L'.top(k)$ ▷ based on $\hat{y}_x[c]$
  $L \leftarrow L \cup L', U \leftarrow U \backslash L'$
  Train F^S ▷ noise can be added with Dropout and Augmentation
  $F^T \leftarrow F^S$

3.2 (Neural) Retrieval-based approaches

We will now present the systems we developed using image encoders combined with a non-parametric retrieval component.

[11] Adapted from [FMi+21].

3.2.1 k-NN

This system follows the work of [KPA19a; KPA19b; Kar+20] and consists of an image encoder followed by a k-NN retrieval module that uses majority voting to assign labels during inference. We experimented with various image encoders (described in Sec. 3.1). For each training image, we computed its image embedding (encoding) $\boldsymbol{v}$ using the (trained) encoder and stored it in an offline index. Given a test image x_q, we obtained its embedding and then retrieved the k most similar training images $\tilde{\boldsymbol{X}} = \{(x_{t_i}, y_i)\}_{i=1}^{k}$ according to the highest cosine similarity $\langle x_q, x_{t_i} \rangle = \frac{x_q^T x_{t_i}}{||x_q|| ||x_{t_i}||}$ with the test image (calculated on image embeddings) and assigned the r most frequent concepts present in them to the test image (see Fig. 3.4). We tuned the two hyper-parameters k, r using the validation set. For k, we experimented with values between 1 and 200. For r, we used values between 1 and 10 and also considered the average number of labels present in the k retrieved images: $r = \frac{1}{k} \sum_{i=1}^{k} |\mathcal{T}_i|$

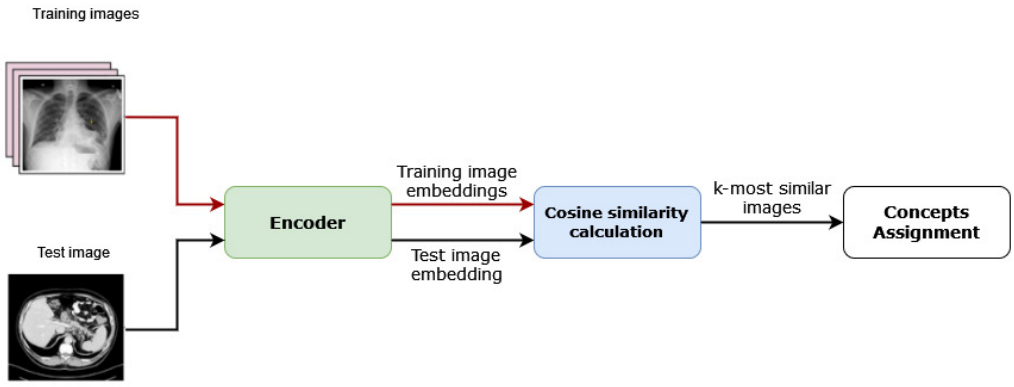

Fig. 3.4: Illustration of how the retrieval-based k-NN system works at inference time. The training image embeddings are computed offline. Figure adapted from [KPA19b].

3.2.2 Weigthed k-NN

This approach is similar to the previous k-NN system but also considers the ranking of the neighbors by using a weight vector $\boldsymbol{w} = [w_1, \ldots, w_k]$. The values of the vector correspond to each one of the k neighbors and we have that $w_1 > \ldots > w_k$, denoting that closer neighbors have larger weights assigned to them.

We experimented with two voting schemes. The first one was introduced in [CLL12] and is based on calculating a score for each label in the label set. Given a test image x_q, for each label l_i, we calculate the weighted sum of k scores, from each of the k neighbors retrieved from the index when x_q was the query:

$$f_i(x_q) = \frac{\sum_{j=1}^{k} w_j \cdot y_i(N_k(x_q, j))}{\sum_{j=1}^{k} w_j} \tag{3.42}$$

where $y_i(N_k(x_q, j))$ denotes the presence of the i-th label in the multi-hot vector of the j-th nearest neighbor of x_q, and w_j is the weight assigned to the j-th nearest neighbor. The weight vector $\boldsymbol{w} = [w_1, \ldots, w_k]$ can be either learned or be manually set with monotonically decreasing weights, e.g., the top-ranked neighbor can be given weight k and the lowest-ranked neighbor weight 1. We used the latter simple linear assignment method in our experiments. Having calculated f_i for every label l_i, we assigned l_i to x_q by the rule:

$$h_i(x_q) = \begin{cases} 1, & f_i(x_q) \geq \tau \\ 0, & f_i(x_q) < \tau \end{cases} \tag{3.43}$$

yielding the predicted label set $H(x_q) = \{l_i | h_i(x_q) = 1\} = \{l_i | f_i(x_q) \geq \tau\}$. We tuned the value of k in the range from $[5, 150]$ with a step of 5 using the validation set. $\tau \in [0, 1]$ is a threshold hyper-parameter.

In the second scheme [Kum+16], the weight of each neighbor is a function of its similarity with test image x_q. In particular, the weight of j-th most similar neighbor is given by:

$$w_j = \frac{d_1 + \boldsymbol{\epsilon}}{d_j + \boldsymbol{\epsilon}} \tag{3.44}$$

where $d_j = 1 - \langle x_q, x_{t_j} \rangle$ denotes the cosine distance and $\boldsymbol{\epsilon}$ is used to avoid divisions by zero. Finally, we again obtain a weighted vote for every label l_i:

$$v_i = \sum_{j=1}^{k} y_i(N_k(x_q, j)) w_j \tag{3.45}$$

Having calculated $\boldsymbol{v} = [v_1, \ldots, v_C]$, we assigned the r labels with the highest v_i to x_q. We tuned the value of k in the range from $[5, 100]$ with a step of 10 using the validation set.

3.2.3 Retrieval-augmented classification

We also made experiments where we used a retrieval component in order to 'augment' the predictions of our neural network classifier. In essence, we combined the labels predicted by the classifier with the labels that were assigned from the images retrieved by the retrieval component. We experimented with two different schemes for combining the labels.

In the first scheme, every test image was passed through classifier and the predicted labels were added (set UNION) to the labels predicted by a k-NN system. For the k-NN predictions, we retrieved the top-k closest training images (w.r.t. cosine similarity computed on the image embeddings) of the test image and returned the r concepts that were most frequently

assigned to the k images (also see Sec. 3.2.1 for detailed description of the k-NN system). For k and r, we used the values produced by the tuning experiments of the standalone k-NN system.

The second scheme follows the work of [SWD22]. The classifier's predictions are interpolated with the neighbors' predictions. More specifically, given a test image $\tilde{x}$ and the classifier's prediction vector $\hat{y}_m \in [0,1]^{|C|}$, $\tilde{x}$ is also used to query the index of training image embeddings and obtain the k nearest neighbors $\tilde{\boldsymbol{X}} = \{(x_{t_i}, y_i)\}_{i=1}^{k}$. Then the k-NN prediction is calculated by:

$$\hat{y}_k = \sum_{i=1}^{k} \alpha_i y_i, \quad \alpha_i = \frac{e^{-d(x_{t_i},\tilde{x})/\tau}}{\sum_j e^{-d(x_{t_j},\tilde{x})/\tau}} \tag{3.46}$$

where $d(\cdot,\cdot)$ denotes a distance metric (e.g., the cosine distance $1 - \langle x_{t_i}, \tilde{x} \rangle$), τ is a scaling parameter and α_i denotes the weight of the i-th neighbor. The final prediction is calculated as the interpolation of the two predictions:

$$\hat{y} = \lambda \hat{y}_k + (1-\lambda)\hat{y}_m, \quad \lambda \in [0,1] \tag{3.47}$$

Additionally, in order to improve the quality of retrieved neighbors, [SWD22] proposed to train the model with a contrastive learning objective that utilizes a dynamic coefficient for each instance pair based on the label similarity. This encourages the model to generate closer representations for instance pairs with more shared labels and push away those pairs that have completely different labels. In that way, the k-NN mechanism retrieves more relevant instances. In practice, for a specific instance i in the current batch of size b, $g(i) = \{j | j \in \{1,2,\ldots,b\}, j \neq i\}$ denotes all the other instances in the batch. The objective for each pair (i,j) can be computed as:

$$\mathcal{L}_{con}^{ij} = -\beta_{ij} \log \frac{e^{-d(z_i,z_j)/\tau'}}{\sum_{k \in g(i)} e^{-d(z_i,z_k)/\tau'}} \tag{3.48}$$

where

$$\beta_{ij} = \frac{C_{ij}}{\sum_{k \in g(i)} C_{ik}}, \quad C_{ij} = y_i^T y_j \tag{3.49}$$

$d(\cdot, \cdot)$ again denotes a distance metric, τ' is a scaling parameter and z_i denotes the image embedding of image i. The total loss is the aggregation over all pairs:

$$\mathcal{L}_{con} = \sum_{i} \sum_{j \in g(i)} \mathcal{L}_{con}^{ij} \tag{3.50}$$

For a pair of instances (i, j), a large value of label similarity C_{ij} will bring larger coefficient β_{ij}, thereby increasing the value of their loss term $\mathcal{L}_{con}^{ij}$. As a result, their distance $d(z_i, z_j)$ will be optimized to be closer. On the other hand, if there are no shared labels $(\beta_{ij} = C_{ij} = 0)$, then the value of $\mathcal{L}_{con}^{ij}$ is also 0. The classifier can be trained using:

$$\mathcal{L} = \mathcal{L}_{BCE} + \gamma \mathcal{L}_{con} \tag{3.51}$$

4 Experiments and results

In this section, we will present the medical dataset that was used in our experiments as well as the evaluation results of these experiments.

4.1 Data

There are several medical image datasets that are available in the literature. They cover a wide range of medical examinations (X-rays, PET/CT scans etc.) as well as body parts. Most commonly, they are used for tasks like classification, detection and segmentation [Lit+17]. The most common medical modality is chest X-rays and several recent works [Joh+19; Raj+18; Irv+19; Wan+17] have introduced large, labeled, chest X-ray datasets (suitable for image classification). Additionally, there are datasets like MIMIC-CXR [Joh+19] which also include textual reports that can be used to train a report generation system or as an extra information modality for multi-modal models. Moreover, there are datasets that are collected and distributed as part of competition tasks. These competitions aim to encourage the implementation of new systems in order to develop automated methods to assist the diagnostic process and therefore help clinicians in their daily workflow. An example of such competitions, the ImageCLEFmedical Caption task [Ion+19; Ion+20; Pel+21; Ion+22], which is held annually, provides a dataset comprised of annotated radiology images.

Medical image datasets can be comprised of patient cases which usually include more than one images but only one report (or just a single set of labels) is associated with the whole case. Using each image of the same case as a different instance (i.e., an image-based approach) may prove insufficient. Intuitively, the diagnosis was conducted by the clinician by looking at all the case's images and on top of that, the medical findings may not be present in every image of the case. To address these limitations, a case-based approach can be followed. There are different types of case-based approaches. The simplest one is to obtain the model's probability distributions over the target labels for each image separately and then average the probabilities and obtain the final decision for the case. However, there is a risk that a label detected in only one image may not be chosen due to the averaging operation [Raj+17]. Another solution would be to to choose the labels with the largest probabilities per image [Irv+19]. Additionally, a more elaborate take is to extract one image embedding for every image of the case and then combine (concatenate,

add or average) them in order to obtain a single image embedding that will be the case representation [Li+18].

In the following section, we will briefly describe the ImageCLEFmedical Caption dataset for the 2022 task which was used in the experiments of this thesis. It has to be noted that the dataset did not have the notion of patient cases so an image-based approach was used.

4.1.1 ImageCLEFmedical Caption 2022

ImageCLEF [Ion+22] is an evaluation campaign held annually since 2003 as part of CLEF, revolving around image, and more generally, multimedia analysis and retrieval tasks.[1] ImageCLEFmedical, which was one of the four main ImageCLEF 2022 tasks, consists of a series of challenges associated with the study and processing of medical images. The ImageCLEFmedical Caption Task [Rüc+22] includes the Concept Detection sub-task, where the goal was to perform multi-label classification of medical images by automatically assigning medical terms (called concepts) to them, and the Caption Prediction sub-task which aims to automatically generate draft diagnostic captions for the images. The work of this thesis is centered around the Concept Detection sub-task. The ImageCLEFmedical Caption 2022 dataset is a subset of the Radiology Objects in Context (ROCO) [Pel+18] dataset, which contains medical images extracted from open access biomedical journal articles of PubMed Central.[2] There are images of several different (radiology) modalities in the dataset (e.g., X-rays, CT-Scans), although no information regarding the modality of each image is present. The concepts that had to be predicted for each image, were extracted based on the medical report of the image and were manually curated. Each concept is represented by a Concept Unique Identifier (CUI) which stems from the Unified Medical Language System (UMLS®) [Bod04]. A mapping between CUIs and actual medical terms was also provided.

[1] http://www.clef-initiative.eu/
[2] https://www.ncbi.nlm.nih.gov/pmc/

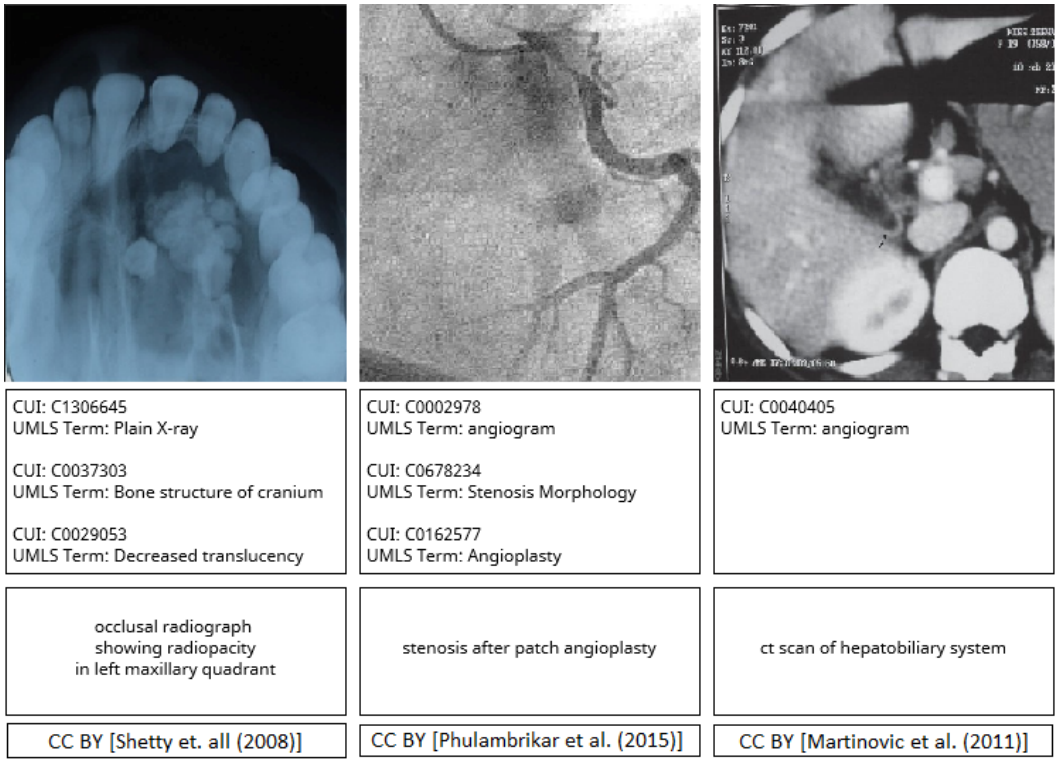

Fig. 4.1: Three images of the dataset [SSS08; MG11; Phu+15] (1st row) with their corresponding ground truth concepts (2nd row) and captions (3rd row).

CUI	UMLS term	Images
C0040405	X-RAY COMPUTED TOMOGRAPHY	28,885
C1306645	PLAIN X-RAY	26,412
C0024485	MAGNETIC RESONANCE IMAGING	15,693
C0041618	ULTRASONOGRAPHY	12,236
C0817096	CHEST	8,030
C0002978	ANGIOGRAM	6,464
C0000726	ABDOMEN	6,243
C0037303	BONE STRUCTURE OF CRANIUM	5,175
C0221198	LESION	4,094
C0205131	AXIAL	3,528

Tab. 4.1: The 10 most frequent concepts (CUIs and corresponding UMLS terms) in the dataset of ImageCLEFmedical Caption 2022 and how many images they are assigned to.

Each image in the dataset is associated with several concepts which were the ground truth labels of the images in the Concept Detection sub-task. An image can be associated with multiple CUIs (see Fig. 4.1). Additionally, Table 4.1 shows the 10 most frequent ground truth concepts of the whole dataset, their corresponding UMLS terms, and how many images they are assigned to. As it can be seen, the concepts used do not only correspond

to medical conditions but also other medical-related terms such as body parts, organs, types of medical exams etc. Moreover, the dataset is highly imbalanced. For example, the concept X-RAY COMPUTED TOMOGRAPHY occurs 28,885 times in the whole dataset (Table 4.1), while there are 1,149 other concepts that appear only 3 times each. A depiction of the imbalance can be seen in Fig. 4.2 where the long tail distribution is visible. We can also see that approximately half of the concepts are present in the data less than 10 times. In addition, there is a large number of unique concepts (number of available classes) that stops at 8,374. The average number of concepts assigned to each image is 4.74, while the minimum number is 1. There are 4,316 images with only 1 assigned concept, and only 1 image with 50 concepts (the maximum number of assigned concepts). An official training set of 83,275 images and an official validation set of 7,645 images were provided. There was also an official test set comprised of 7,601 images but its gold truth concepts were hidden. For our experiments, we merged the provided (official) training and validation sets, and used 15% of the merged data as our validation set, and another 20% of the merged data as our development set; the former was used for hyper-parameter tuning, whereas the latter was used to evaluate the performance of our models. The remaining 65% served as our training set. The results presented in Section 4.3, are based on our development set which included gold truth concepts.

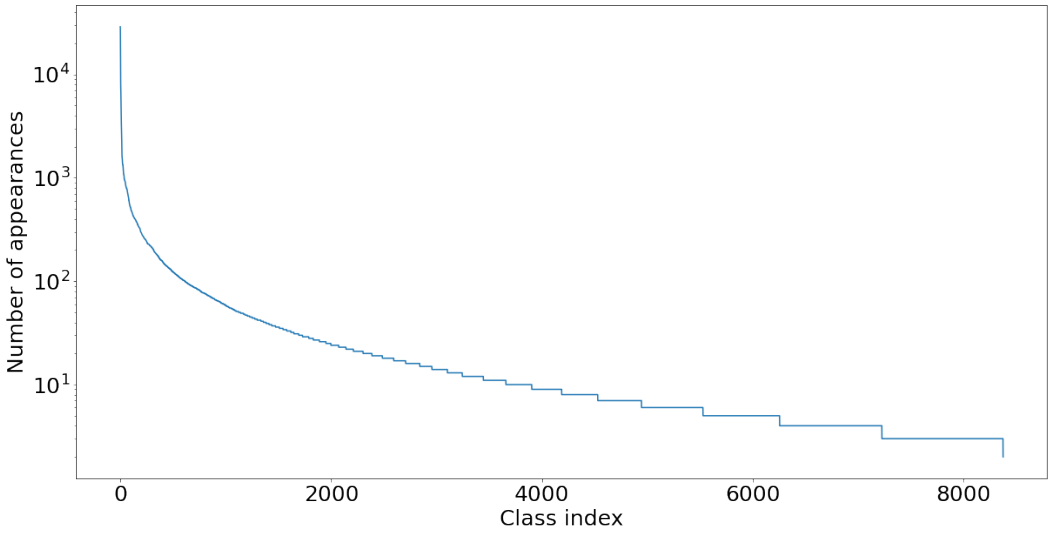

Fig. 4.2: Number of images (vertical axis) tagged with each concept in the training data. To save space, the horizontal axis shows the index of each concept (class index), instead of its CUI.

4.2 Evaluation measures

The evaluation of the models is conducted using the F_1 score between predicted and ground truth concepts. The F_1 score is calculated per test instance i by comparing the binary multi-hot ground truth label vector y^i and the predicted binary multi-hot label vector $\hat{y}^i$. Then the total F_1 score is computed by macro-averaging over all the test instances:

$$F_1^* = \frac{\sum_{i=1}^{n} F_1(y^i, \hat{y}^i)}{n}, \quad n = \text{number of test instances} \tag{4.1}$$

This averaging is known as 'samples' averaging.[3] This was the official measure of the ImageCLEFmedical competition and we also utilized it for our additional experiments.

4.3 Experimental Results

In this section we present the evaluation of the methods proposed in Section 3.

4.3.1 Visual encoders

Our first set of experiments consisted of a search for the architecture of the classifier. We experimented with several architectures for the visual encoder backbone. We considered various convolutional architectures as well as Transformer-based networks and Transformer-convolutional hybrids. Additionally, we experimented with various classification heads by tuning the hyper-parameters (i.e., number of hidden layers) of the final classification MLP. Tables 4.2 and 4.3 contain the results of the corresponding trials in terms of F_1 score. Table 4.2 displays the results of an EfficientNetV2-B0 encoder [TL21] when combined with a linear classifier and a 2-layer non-linear MLP respectively. Table 4.3 shows a subset of the experiments with several image encoders. Due to the fact that a more complex classification head did not provide any notable improvement, all experiments in Table 4.3 were made using a linear classification head.

Approach	$\mathbf{F_1^*}$ Score
EfficientNetV2-B0[TL21] + MLP	45.05
EfficientNetV2-B0[TL21] + LR	**45.25**

Tab. 4.2: Scores comparing a non-linear vs. a linear classifcation head.

Looking at Table 4.3, we can firstly deduce that convolutional models outperformed Transformer-based models by a large margin. This may be explained by the fact that Vision Transformers are data-hungry models and work well only with very large sets of data as they lack the inductive biases that exist in CNNs. Secondly, we can see that using more complex and parameter-heavier models does not necessarily result in better performance. Thus, during our early experiments, we decided to stick with the EfficientNetV2-B0 [TL21] backbone CNN and conduct the following experiments using this network.

[3] Also see: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

Encoder	# Parameters	$\mathbf{F_1^*}$ Score
EfficientNetV2-B0[TL21]	7.2M	**45.25**
DenseNet-121[Hua+17]	8.1M	44.08
ResNet-50[He+16]	25.6M	43.02
EfficientNet-B0[TL19]	5.3M	44.93
ViT-B[Dos+21]	86M	39.86
Swin-ViT-B[Liu+21]	88M	21.99
ConvNeXt-B[Liu+22]	89M	44.89
CoAtNet-0[Dai+21]	25M	43.26
NFNet-F1[Bro+21]	132.6M	45.18

Tab. 4.3: Scores comparing backbone visual encoders.

4.3.2 Loss functions and optimization

As mentioned in Sec. 3.1.2, standard BCE loss usually makes the model to output a low probability value for rare classes. Given that our dataset has high imbalance, we experimented with the Focal [Lin+17] and ASL [Bar+21] losses (described in Sec. 3.1.2) in order to try and make the model focus on rare classes during training. Results can be seen in Table 4.4.

Loss function	$\mathbf{F_1^*}$ Score
$\mathcal{L}_{FL}\ (\gamma = 2, \alpha = 0.25)$	45.14
$\mathcal{L}_{FL}\ (\gamma = 2, \alpha = 0.4)$	44.87
$\mathcal{L}_{FL}\ (\gamma = 4, \alpha = 0.25)$	44.46
$\mathcal{L}_{FL}\ (\gamma = 4, \alpha = 0)$	44.15
$\mathcal{L}_{FL}\ (\gamma = 2, \alpha = 0)$	44.57
$\mathcal{L}_{ASL}\ (\gamma_- = 4, \gamma_+ = 0, m = 0.05)$	45.09
$\mathcal{L}_{ASL}\ (\gamma_- = 2, \gamma_+ = 1, m = 0)$	45.00
$\mathcal{L}_{ASL}\ (\gamma_- = 4, \gamma_+ = 1, m = 0.5)$	43.01
$\mathcal{L}_{ASL}\ (\gamma_- = 10, \gamma_+ = 0, m = 0.05)$	43.98
$\mathcal{L}_{ASL}\ (\gamma_- = 2, \gamma_+ = 0, m = 0.05)$	45.45
$\mathcal{L}_{F_1} * \mathcal{L}_{BCE}$	**45.85**

Tab. 4.4: Scores comparing different loss functions.

We can see that both $\mathcal{L}_{FL}$ and $\mathcal{L}_{ASL}$ can be competitive, given certain values of hyper-parameters. We observed that the output probabilities given to each class were higher than that given from $\mathcal{L}_{BCE}$. However, in our experiments, both $\mathcal{L}_{FL}$ and $\mathcal{L}_{ASL}$ did not manage to surpass $\mathcal{L}_{BCE}$ (see Table 4.3) in most cases in terms of the total (average) performance. A reason for this may be that despite the higher output values given in under-represented classes, the average performance depends on the threshold value τ, which apparently had a higher value than the output probabilities and did not allow these classes to be selected in the final prediction. Additionally, another reason could be the limited tuning that we

conducted. A more extensive search could possibly provide more suitable hyper-parameter values for this specific dataset and result in a better performance.

On the contrary, $\mathcal{L}_{F_1} * \mathcal{L}_{BCE}$ provided an improvement in the performance which is not strange considering that it indirectly optimizes our evaluation function. Using just $\mathcal{L}_{F_1}$ resulted in no convergence in our experiments, probably due to the restricted range of the loss value.

As for the optimization algorithms, we experimented with SAM [For+21] and GC [Yon+20] (both described in Sec. 3.1.3). Using GC, we did not manage to surpass the previous performance of the model. On the contrary, SAM manages to generalize slightly better but when combined with GC the performance drops. In general, the use of GC seems to lead to a degradation of performance, albeit not a statistical significant one. Additionally, using a separate threshold value t_i for each concept c_i, acquired via Bayesian Optimization [SLA12], did not add any improvement to the performance. However, due to the large amount of concepts ($|C| = 8{,}374$) and the fact that each threshold can be given any value in $[0, 1]$, we performed a limited search, given our time and resource restrictions. It is probable that a more diverse search (or even a full value swipe), could provide an improved performance. Overall, having a unique threshold value for each concept is a better idea than having a single same value for all concepts.

Lastly, as an ablation test, we decided to optimize a subset of the model's parameters, following BitFit [ZGR22] and BNFit [FSM21]. In BitFit, we allow only the bias terms of each layer to be updated during fine-tuning by keeping the weight matrices frozen. Similarly, in BNFit, we fine-tune only the Batch Normalization [IS15] layers. These experiments provided a (slightly) faster training time but worse performance. But, as [ZGR22; FSM21] suggest, both solutions can be a competitive faster alternative in a low-resource regime.

Approach	$\mathbf{F_1^*}$ Score
$\mathcal{L}_{F_1}$ + SAM	**45.94**
$\mathcal{L}_{F_1}$ + GC	45.58
$\mathcal{L}_{F_1}$ + SAM + GC	45.75
$\mathcal{L}_{F_1}$ + BAYES OPT.	45.68
$\mathcal{L}_{F_1}$ + BitFit [ZGR22]	44.71
$\mathcal{L}_{F_1}$ + BNFit [FSM21]	44.95

Tab. 4.5: Scores comparing different optimization methods.

Despite the fact that SAM provided a better performance score, we did not use it for the following experiments as its two-step optimization doubled the time needed for each epoch to end.

4.3.3 Ensembles

We created four ensemble models using the UNION set operation, four ensemble models using INTERSECTION and one using MODEL SOUPS [Wor+22]. For the UNION and INTERSECTION ensembles, we created four different variation using two, three, four and five instances of the base classifier, resulting in eight models in total (four per set operation).

Approach	F_1^* Score
UNION - 2 instances	**45.91**
UNION - 3 instances	45.86
UNION - 4 instances	45.76
UNION - 5 instances	45.57
INTERSECTION - 2 instances	45.63
INTERSECTION - 3 instances	45.44
INTERSECTION - 4 instances	45.28
INTERSECTION - 5 instances	45.22
MODEL SOUPS	45.51

Tab. 4.6: Scores comparing different ensemble models.

The instances used for the ensembles were created using Monte Carlo cross-validation [XL01]. Each instance was initialized with different weight values, yielding a different optimization space per instance (thus, a different optimal minimum point). We can see that with more instances, the score diminishes. However, the data split plays an important role (see Sec. 4.3.10)

4.3.4 Self-supervision

For MAE [He+22], we initially experimented with the ImageNet pre-trained model. We added a linear classifier on top of the ViT and ran two experiments by freezing and unfreezing the pre-trained encoder. Next, we further pre-trained the model using our dataset. We then conducted the same two experiments using the further pre-trained encoder. Results can be seen in Table 4.7. We can conclude that the ImageNet pre-trained MAE had much better results compared to its supervised ImageNet pre-trained counterpart (see Table 4.3). The performance is further improved with the extra pre-training step. These results show the the usefulness of MAE's pre-training when it comes to the ViT architecture.

For CLIP [Rad+21], we conducted a few experiments where we concatenated its produced embeddings with the embeddings of the base image encoder. We experimented with variations which included freezing and unfreezing CLIP during the training process as well as adding Dropout [Sri+14] to one or both signals.

The same experiments were made using the dVAE. We used two different models. The first one was the dVAE provided by OpenAI (which was used to train DALL-E [Ram+21]) and the second was the dVAE provided in the ViLMedic package [Del+22] which was pre-trained on MIMIC [Joh+19]. The visual tokens were extracted via the codebook [OVK17] of the architecture.

Approach	F_1^* Score
MAE (Linear probing)	41.25
MAE (Fine-tuning)	43.58
MAE (pre-training + Linear probing)	43.27
MAE (pre-training + Fine-tuning)	44.49
CLIP (freezed embeddings)	45.77
CLIP (trainable)	45.68
CLIP + Dropout (base encoder signal)	45.42
CLIP + Dropout (both signals)	45.69
dVAE (OpenAI)	45.74
dVAE (OpenAI + projection)	45.71
dVAE (OpenAI + Dropout (base encoder signal))	25.99
dVAE (OpenAI + Dropout (both signals))	**45.85**
dVAE (ViLMedic)	45.60
dVAE (ViLMedic + projection)	45.70
dVAE (ViLMedic + projection + Dropout (base encoder signal))	33.00
dVAE (ViLMedic + projection + Dropout (both signals))	45.62

Tab. 4.7: Scores comparing experiments with self-supervised methods.

Table 4.7 contains the results of our experiments. In the experiments with CLIP and dVAE, when Dropout is applied to the base image encoder signal, it is applied to the whole output embedding (not just a random subset of the neurons) in order to make the model pay attention only to the CLIP (or dVAE) embedding. On the contrary, when Dropout is applied to both signals, it is applied on random features of the corresponding output embeddings (standard Dropout [Sri+14]).

In the case of the dVAE, the experiment with Dropout applied to the base image encoder signal produced a very low score, indicating that the classifier relied mainly on the signal of the CNN encoder in order to produce a competitive score. The same pattern emerged for ViLMedic's model.

4.3.5 Information from the labels

Here, we focused on treating the relations between the concepts as a graph and tried to extract any information that this graph could provide us using GNNs.

For DXML [Zha+18] that required an unsupervised graph representation algorithm, we experimented with NODE2VEC [GL16], GRAPHSAGE [HYL17b] and DEEP GRAPH INFOMAX (DGI) [Vel+19]. After training the projection, we used linear probing in order to evaluate the learned representations.

For ML-GCN [CWG19], we conducted various experiments using GCNs with different depth as well as different initialization strategies for the node embeddings. In particular, we tried using randomly initialized node embeddings as well as node embeddings initialized with the corresponding values of the concept's word embeddings. To derive these word embeddings, we experimented with fastText [Boj+17] and sentence [PGJ18] embeddings pre-trained on biomedical corpora [Zha+19; CPL19] as well contextual embeddings derived from a BERT model [Dev+19] pre-trained on medical data [Bas+20].

We also performed an ablation test where we removed the GCN and just used the DOT PRODUCT between the concept's word embeddings and the image embedding.

For LDGN [Ma+21], limited experiments were conducted due to time and resource restrictions. The large size of the adjacency matrix resulted at a time-consuming reconstruction step. For this reason, small-sized GCNs and a small-sized attention MLP were used.

Approach	F_1^* Score
DXML (NODE2VEC)	45.40
DXML (GRAPHSAGE)	45.44
DXML (GRAPHSAGE + MSE)	45.57
DXML (DGI)	43.65
ML-GCN (2-layer GCN - RANDOM INIT.)	45.56
ML-GCN (1-layer GCN - RANDOM INIT.)	**45.68**
ML-GCN (1-layer GCN - FASTTEXT INIT.)	45.02
ML-GCN (1-layer GCN - SENT2VEC INIT.)	45.35
ML-GCN (1-layer GCN - BERT INIT.)	44.78
ML-GCN (1-layer GCN - BERT INIT. (w/ projection))	45.45
Word Embeddings (FASTTEXT INIT.)	43.13
Word Embeddings (SENT2VEC INIT.)	44.88
Word Embeddings (BERT INIT.)	44.97
LDGN	45.47

Tab. 4.8: Scores comparing different label-space aware experiments.

DXML provided competitive scores but it did not manage to outperform the baseline. GRAPHSAGE (combined with the MSE loss) scored the highest. What is interesting about DXML, is that these competitive scores were obtained by predicting a very small amount of possible concepts (~ 30). This could be due to the information provided by the structure of the graph, which was incorporated in the training of the model and allowed it to focus on the more prominent concepts.

The same observation holds for the ML-GCN model. A 1-layer GCN combined with randomly initialized node embeddings performed the best amongst the experiments with ML-GCN while in the ablation with the GCN removed, the use of BERT's contextual embeddings provided a better score.

LDGN was out-performed by ML-GCN for a small margin but this can be due to the small-sized networks that were used. Additionally, it managed to predict a larger number of concepts (~ 100) compared to the previous two graph-based approaches, indicating that the adjacency matrix reconstruction pays attention to not so common labels.

4.3.6 Preliminary experiments with Teacher-Student self-training

As mentioned in Sec. 3.1.7, we used a modified version of the Noisy Student algorithm [Xie+20] in which we obtained confident samples from the unlabeled dataset based on certain concepts (in order to carry out just a preliminary exploration). For the unlabeled images, we used the ROCO dataset [Pel+18] which constitutes a superset of the competition's dataset.[4] We randomly picked 15 concepts, 5 from each 'size band' of the dataset. In essence, we picked 5 concepts that appeared frequently in our test set, 5 that appeared few times and 5 that appeared very few times. We ran the self-training algorithm for a fixed number of only 2 iterations. Results for each concept can be seen in Table 4.9. The comparison was made based on the **concept's** F_1 score.

Concept	**Occurences**	**Per-class F_1 Score**	
		Before	**After**
C0817096	1599	66.02	**68.31**
C0002978	1275	94.94	94.79
C0000726	1087	49.85	**50.26**
C0037303	848	86.38	**86.59**
C0221198	51	0.23	0.23
C0205126	79	0.0	**10.71**
C0046056	69	45.28	**48.74**
C3827002	56	14.00	**20.45**
C0449468	75	0.0	0.0
C0042149	18	0.0	**4.87**
C0025526	18	18.18	10.0
C0223683	16	11.11	10.0
C0225358	8	0.0	**9.09**
C0001209	10	0.0	0.0
C1629036	6	0.0	**50.0**

Tab. 4.9: Comparison of concepts before and after self-training.

[4] https://github.com/razorx89/roco-dataset

We can see that the F_1 score improved for the majority of the selected concepts (even those with few occurrences). However, this specific experiment was very limited as there were not many confident samples for certain labels. A non-constrained (on specific labels) Teacher-Student approach with more iterations could provide additional performance benefits and also improve the predictions for the tail labels.

4.3.7 k-NN retrieval with majority voting

For this system, we conducted a single experiment using the supervised-trained image encoder. The result can be seen in Table 4.10. We tuned the values of k and r using our validation set, which resulted in the values seen in Table 4.10.

Encoder	Hyper-parameters	$\mathbf{F_1^*}$ Score
EfficientNetV2B0[TL21]-supervised	$k = 73, r = 2$	43.78

Tab. 4.10: Scores comparing different encoders combined with k-NN.

4.3.8 Adjusting the weight of each neighbor

Trying to improve the performance of k-NN, the same experiment was conducted for the weighted version of the algorithm. For each one of the voting schemes, we used the supervised encoder, acquiring the following results:

Approach	Hyper-parameters	$\mathbf{F_1^*}$ Score
Linear weighting (EfficientNetV2B0[TL21]-supervised)	$k = 10$	**45.27**
Distance weighting (EfficientNetV2B0[TL21]-supervised)	$k = 60, r = 2$	43.97

Tab. 4.11: Scores comparing different encoder combined with k-NN.

We can see that attributing weights to the neighbors availed the k-NN classifier and resulted in better performance, especially in the linear weighting scheme. For τ, we used a value of 0.4.

4.3.9 Combining retrieval and classifier predictions

These experiments aimed at making the predictions of the models more diverse. We observed that, the number of the predicted concepts of our models was very low compared to the total number of possible concepts (i.e., only a few hundreds vs. thousands). In general, our models tended to predict concepts that appeared frequently in the dataset (i.e., not concepts from the tail - see Fig. 4.2) with a high performance in terms of precision. However, they yielded low recall scores due to the fact that the 'retrieved' concepts were few. In order to alleviate this problem, we tried to augment the predictions of the base classifier

with concepts that belonged to nearest neighbors (see Sec. 3.2.3 for the methodology). Results can be seen in Table 4.12.

Approach	Hyper-parameters	F_1^* Score
UNION	$k = 10, r = 2$	34.66
UNION	$k = 10, r = 1$	39.02
UNION	$k = 15, r = 1$	40.36
UNION	$k = 73, r = 1$	40.21
UNION (RARE)	$k = 15, r = 1$	37.81
INTERPOLATION	$k = 10, \lambda = 0.5, \tau = 1$	42.44
INTERPOLATION	$k = 10, \lambda = 0.25, \tau = 1$	45.68
INTERPOLATION	$k = 5, \lambda = 0.25, \tau = 1$	45.77
INTERPOLATION	$k = 70, \lambda = 0.2, \tau = 1$	45.74
INTERPOLATION - INVERTED COS SIM.	$k = 5, \lambda = 0.25, \tau = 1$	**45.89**
INTERPOLATION + CL [SWD22]	$k = 5, \lambda = 0.25, \tau = 1, \gamma = 0.005, \tau' = 10$	45.74

Tab. 4.12: Scores comparing retrieval-augmented classification experiments.

We observe that aggregating extra concepts in the classifier using the UNION method resulted in a big performance decrease. The extra concept(s) slightly improved the recall performance (due to the 'retrieval' of diverse concept(s)) but resulted in a much lower score precision-wise leading to worse F_1. UNION (RARE) is a variation where only concepts from the tail of the distribution (i.e., concepts with very few occurrences - in our experiments we picked the value of 20), are retrieved from the nearest neighbors.

As for the INTERPOLATION method, it provided competitive performance, especially when the inverted cosine similarity $\frac{1}{\langle x,y \rangle}$ was used as the distance metric. In this scheme, the predictions were not diversified (i.e., recall scores did not increase), probably due to the thresholding process which was applied after the interpolation of the probabilities and was the same as the one in the baseline. Moreover, the contrastive objective did not offer any benefits in the performance of this particular task.

4.3.10 ImageCLEFmedical Caption 2022 submissions

We managed to rank 1st in the Concept Detection sub-task of the ImageCLEFmedical Caption task using the UNION ensemble of 2 instances of our CNN+FFNN classifier. We submitted 5 more systems that are described in Table 4.13. All the additional methods and experiments that are described in Sections 3 and 4, were conducted as follow-up work for the fulfilment of this thesis.

Apart from the primary measure F_1^*, this year's task included a secondary F_1 score that was calculated using a subset of manually validated concepts (only anatomy and image modality ones).

ID	*Run ID*	Approach	Primary F_1^*		Secondary F_1^*	Rank
			Development	Test		
cd1	182358	2xCNN[TL21]+FFNN@U	**47.01**	**45.11**	79.07	**1**
cd2	182356	2xCNN[TL21]+FFNN@I	46.32	44.27	**84.42**	12
cd3	182359	2xCNN[TL21]+FFNN/wk-NN	–	44.63	84.30	5
cd4	182340	CNN[TL21]+FFNN	45.71	44.39	81.20	9
cd5	182354	CNN[TL21]+FFNN(SAM+GC+$\mathcal{L}_{F_1}$)	46.06	45.02	82.14	3
cd6	182333	CNN[TL21]+wk-NN	45.25	43.05	84.40	28

Tab. 4.13: Scores and rankings of our submitted systems on our final development and the official test sets. cd3's development score is not available, as its predictions were created using the output files of the ensemble's members.

2xCNN+FFNN@U and 2xCNN+FFNN@I denote the UNION and INTERSECTION ensembles respectively while 2xCNN+FFNN/wk-NN denotes an ensemble of 2 instances of the CNN+FFNN system (see Sec. 3.1.1) and an instance of the CNN+wk-NN which is the system of Sec. 3.2.2. This last ensemble used a majority voting rule in order to assign concepts to the test images (i.e., assign a concept to a test image iff it is predicted by at least 2 of the 3 systems).

To generate the predictions for the official test set, we merged our training, validation and development data. We used a held-out set (15% of the merged data) as our final development set for hyper-parameter tuning and trained the models using the rest of the data. This is why the Development scores of Table 4.13 are different and cannot be compared with the scores presented in the rest of the tables.

5 Conclusions and Future Work

5.1 Conclusions

This thesis addressed the task of medical image tagging, a form of multi-label medical image classification. The main goal of the task is to assist clinicians and the diagnostic process when they examine medical images by automatically assigning appropriate tags that correspond to the findings in an image. Furthermore, it can potentially constitute a first step in the automatic generation of diagnostic reports. Following the recent advancements in medical image analysis, the systems implemented in this thesis were based on Deep Learning. We benchmarked several image encoder architectures, tested self-supervised models and experimented with loss functions and GNNs in order to tackle the class imbalance of the dataset. Furthermore, we conducted preliminary experiments using Teacher-Student self-training in order to exploit additional unlabeled data for under-represented categories. Our approaches also included neural retrieval-based methods which, once more, proved very competitive. We managed to acquire the 1st position at the 2022 ImageCLEFmedical Caption's Concept Detection sub-task using an ensemble of simple classifiers.

5.2 Future Work

A direction for future work could be the further experimentation with the Teacher-Student self-training in order to further improve performance when using imbalanced datasets. Additionally, integrating a re-ranking step in retrieval-based methods could provide performance improvements by bringing more relevant neighbors higher in the neighbors list and thus giving them a bigger weight value. Moreover, integrating the retrieval augmentation in the training process [Sar+22] could constitute a topic for future research in the medical image classification task. Finally, given the collection of sufficient medical unlabeled data, self-supervised pre-training could provide more meaningful representations that will be more suitable for solving downstream tasks than representations learned from pre-training on (out-of-domain) ImageNet [Den+09].

Bibliography

[Azi+21] S. Azizi, B. Mustafa, F. Ryan, et al. "Big Self-Supervised Models Advance Medical Image Classification". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Online, 2021, pp. 3478–3488.

[Bao+22] H. Bao, L. Dong, S. Piao, and F. Wei. "BEiT: BERT Pre-Training of Image Transformers". In: *International Conference on Learning Representations*. Online, 2022.

[Bar+21] E. Ben Baruch, T. Ridnik, N. Zamir, et al. "Asymmetric Loss For Multi-Label Classification". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 82–91.

[Bas+20] M. Basaldella, F. Liu, E. Shareghi, and N. Collier. "COMETA: A Corpus for Medical Entity Linking in the Social Media". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online, 2020, pp. 3122–3137.

[Bel+19] I. Bello, B. Zoph, Q. Le, A. Vaswani, and J. Shlens. "Attention Augmented Convolutional Networks". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea, 2019, pp. 3285–3294.

[BKH16] J. L. Ba, J. R. Kiros, and G. E. Hinton. *Layer Normalization*. 2016.

[Bod04] O. Bodenreider. "The Unified Medical Language System (UMLS): Integrating Biomedical Terminology". In: *Nucleic acids research* 32 (2004), p. 4.

[Boj+17] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146.

[Bro+21] A. Brock, S. De, S. L. Smith, and K. Simonyan. "High-Performance Large-Scale Image Recognition Without Normalization". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. 2021, pp. 1059–1071.

[Car+18] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. "Deep Clustering for Unsupervised Learning of Visual Features". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Munich, Germany, 2018.

[Car+20] M. Caron, I. Misra, J. Mairal, et al. “Unsupervised Learning of Visual Features by Contrasting Cluster Assignments”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Online, 2020, pp. 9912–9924.

[Car+21] M. Caron, H. Touvron, I. Misra, et al. “Emerging Properties in Self-Supervised Vision Transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Online, 2021, pp. 9650–9660.

[Cau47] A. Cauchy. “Méthode générale pour la résolution des systémes d’équations simultanées”. In: *Comptes rendus de l’Académie des Sciences* 25 (1847), pp. 536–538.

[CH21] X. Chen and K. He. “Exploring Simple Siamese Representation Learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 15750–15758.

[Cha+21] F. Charalampakos, V. Karatzas, V. Kougia, J. Pavlopoulos, and I. Androutsopoulos. “AUEB NLP Group at ImageCLEFmed Caption Tasks 2021”. In: *CLEF2021 Working Notes*. CEUR Workshop Proceedings. Bucharest, Romania: CEUR-WS.org, 2021, pp. 1184–1200.

[Che+20a] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. “A Simple Framework for Contrastive Learning of Visual Representations”. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. 2020, pp. 1597–1607.

[Che+20b] Q. Cheng, J. Kwon, M. Glick, et al. “Silicon Photonics Codesign for Deep Learning”. In: *Proceedings of the IEEE* PP (2020), pp. 1–22.

[CLJ20] J. Cordonnier, A. Loukas, and M. Jaggi. “On the Relationship between Self-Attention and Convolutional Layers”. In: *8th International Conference on Learning Representations, ICLR 2020*. Addis Ababa, Ethiopia, 2020.

[CLL12] T. Chiang, H. Lo, and S. Lin. “A Ranking-based KNN Approach for Multi-Label Classification”. In: *Proceedings of the Asian Conference on Machine Learning*. Vol. 25. Singapore Management University, Singapore, 2012, pp. 81–96.

[CPL19] Q. Chen, Y. Peng, and Z. Lu. “BioSentVec: creating sentence embeddings for biomedical texts”. In: *2019 IEEE International Conference on Healthcare Informatics (ICHI)*. Los Alamitos, CA, USA, 2019, pp. 1–5.

[CWG19] Z. Chen, X. Wei, and P. Wangand Y. Guo. “Multi-Label Image Recognition With Graph Convolutional Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[CXH21] X. Chen, S. Xie, and K. He. “An Empirical Study of Training Self-Supervised Vision Transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Online, 2021, pp. 9640–9649.

[Dai+21] Z. Dai, H. Liu, Q. V. Le, and M. Tan. “CoAtNet: Marrying Convolution and Attention for All Data Sizes”. In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021, pp. 3965–3977.

[dAs+21] S. d'Ascoli, H. Touvron, M. L. Leavitt, et al. "ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 2286–2296.

[Dea20] J. Dean. "The Deep Learning Revolution and Its Implications for Computer Architecture and Chip Design". In: *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. San Fransisco, CA, USA, 2020, pp. 8–14.

[Del+22] J. Delbrouck, K. Saab, M. Varma, et al. "ViLMedic: a framework for research at the intersection of vision and language in medical AI". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Dublin, Ireland, 2022, pp. 23–34.

[Den+09] J. Deng, W. Dong, R. Socher, et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Miami, FL, USA, 2009, pp. 248–255.

[Dev+19] J. Devlin, M. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota, USA, 2019, pp. 4171–4186.

[DGE15] C. Doersch, A. Gupta, and A. A. Efros. "Unsupervised Visual Representation Learning by Context Prediction". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile, 2015.

[DHS11] J. Duchi, E. Hazan, and Y. Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159.

[Din+22] M. Ding, B. Xiao, N. Codella, et al. "DaViT: Dual Attention Vision Transformer". In: (2022). eprint: arXiv:2204.03645.

[Dos+21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *9th International Conference on Learning Representations, ICLR*. Vienna, Austria, 2021.

[Dwi+21] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. "With a Little Help From My Friends: Nearest-Neighbor Contrastive Learning of Visual Representations". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Online, 2021, pp. 9588–9597.

[Eic+17] C. Eickhoff, I. Schwall, A. García Seco de Herrera, and H. Müller. "Overview of ImageCLEFcaption 2017 – Image Caption Prediction and Concept Detection for Biomedical Images". In: *CLEF2017 Working Notes*. CEUR Workshop Proceedings. Dublin, Ireland: CEUR-WS.org, 2017.

[EKN17] A. Esteva, B. Kuprel, and R. A. Novoa. “Dermatologist-level classification of skin cancer with deep neural networks”. In: *Nature* 542 (2017), pp. 115–118.

[Erm+21] A. Ermolov, A. Siarohin, E. Sangineto, and N. Sebe. “Whitening for Self-Supervised Representation Learning”. In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. Online, 2021, pp. 3015–3024.

[FH89] E. Fix and J. L. Hodges. “Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties”. In: *International Statistical Review / Revue Internationale de Statistique* 57.3 (1989), pp. 238–247.

[FMi+21] F.Mi, W. Zhou, L. Kong, et al. “Self-training Improves Pre-training for Few-shot Learning in Task-oriented Dialog Systems”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic, 2021, pp. 1887–1898.

[For+21] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. “Sharpness-aware Minimization for Efficiently Improving Generalization”. In: *International Conference on Learning Representations*. Online, 2021.

[FSM21] J. Frankle, D. J. Schwab, and A. S. Morcos. “Training BatchNorm and Only BatchNorm: On the Expressive Power of Random Features in {CNN}s”. In: *9th International Conference on Learning Representations, ICLR 2021*. Vienna, Austria, 2021.

[Fuk75] K. Fukushima. “Cognitron: A self-organizing multilayered neural network”. In: *Biological Cybernetics* 20.3 (1975), pp. 121–136.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[GF18] P. Goyal and E. Ferrara. “Graph embedding techniques, applications, and performance: A survey”. In: *Knowledge Based Systems* 151 (2018), pp. 78–94.

[GL16] A. Grover and J. Leskovec. “node2vec: Scalable Feature Learning for Networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and Data Mining*. San Francisco, CA, USA, 2016, pp. 855–864.

[Gri+20] J. Grill, F. Strub, F. Altché, et al. In: *Advances in Neural Information Processing Systems*. Vol. 33. Online, 2020, pp. 21271–21284.

[He+16] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA, 2016, pp. 770–778.

[He+20] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick. “Momentum Contrast for Unsupervised Visual Representation Learning”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 9726–9735.

[He+22] K. He, X. Chen, S. Xie, et al. “Masked Autoencoders Are Scalable Vision Learners”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, LA, USA, 2022, pp. 16000–16009.

[Hes+19] M. H. Hesamian, W. Jia, X. He, and P. Kennedy. “Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges”. In: *Journal of Digital Imaging* 32 (2019).

[Hua+17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. “Densely Connected Convolutional Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA, 2017, pp. 2261–2269.

[Hub64] P. J. Huber. “Robust Estimation of a Location Parameter”. In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. “Distilling the Knowledge in a Neural Network”. In: *NIPS Deep Learning and Representation Learning Workshop*. Montréal, Canada, 2015.

[HYL17a] W. Hamilton, Z. Ying, and J. Leskovec. “Inductive Representation Learning on Large Graphs”. In: *Advances in Neural Information Processing Systems*. Long Beach, CA, USA, 2017.

[HYL17b] W. Hamilton, Z. Ying, and J. Leskovec. “Inductive Representation Learning on Large Graphs”. In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[Ion+19] B. Ionescu, H. Müller, R. Péteri, et al. “ImageCLEF 2019: Multimedia Retrieval in Medicine, Lifelogging, Security and Nature”. In: *Experimental IR Meets Multilinguality, Multimodality, and Interaction*. Proceedings of the Tenth International Conference of the CLEF Association (CLEF 2019). Lugano, Switzerland, 2019, pp. 358–386.

[Ion+20] B. Ionescu, H. Müller, R. Peteri, et al. “ImageCLEF 2020: Multimedia Retrieval in Lifelogging, Medical, Nature, and Internet Applications”. In: 2020.

[Ion+22] B. Ionescu, H. Müller, R. Péteri, et al. “Overview of the ImageCLEF 2022: Multimedia Retrieval in Medical, Social Media and Nature Applications”. In: *Experimental IR Meets Multilinguality, Multimodality, and Interaction*. Proceedings of the 13th International Conference of the CLEF Association (CLEF 2022). Bologna, Italy: LNCS Lecture Notes in Computer Science, Springer, Sept. 2022.

[Irv+19] J. A. Irvin, P. Rajpurkar, M. Ko, et al. “CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison”. In: *AAAI*. 2019, pp. 590–597.

[IS15] S. Ioffe and C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning*. Lille, France, 2015, pp. 448–456.

[JGP17] E. Jang, S. Gu, and B. Poole. “Categorical Reparameterization with Gumbel-Softmax”. In: *5th International Conference on Learning Representations, ICLR*. Toulon, France, 2017.

[Joh+19] A. E. W. Johnson, T. Pollard, S. J. Berkowitz, et al. "MIMIC-CXR, a de-identified publicly available database of chest radiographs with free-text reports". In: *Scientific Data* 6 (2019), p. 317.

[Kar+20] B. Karatzas, V. Kougia, J. Pavlopoulos, and I. Androutsopoulos. "AUEB NLP Group at ImageCLEFmed Caption 2020". In: *CLEF2020 Working Notes*. CEUR Workshop Proceedings. Thessaloniki, Greece: CEUR-WS.org, 2020.

[KB15] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations (ICLR)*. San Diego, CA, USA, 2015.

[KC+21] K. KC, Z. Yin, D. Li, and Z. Wu. "Impacts of Background Removal on Convolutional Neural Networks for Plant Disease Classification In-Situ". In: *Agriculture* 11.9 (2021).

[Kij+19] R. Kijowski, F. Liu, F. Caliva, and V. Pedoia. "Deep Learning for Lesion Detection, Progression, and Prediction of Musculoskeletal Disease". In: *Journal of Magnetic Resonance Imaging* 52 (2019).

[KL51] S. Kullback and R. A. Leibler. "On information and sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[KNS20] M. Kalimuthu, F. Nunnari, and D. Sonntag. "A Competitive Deep Neural Network Approach for the ImageCLEFmed Caption 2020 Task". In: *CLEF2020 Working Notes*. CEUR Workshop Proceedings. Thessaloniki, Greece: CEUR-WS.org, 2020.

[Kot20] S. Kotitsas. "Neural Graph Representations and their Application to Link Prediction". MSc thesis, Department of Informatics, Athens University of Economics and Business. 2020.

[KPA19a] V. Kougia, J. Pavlopoulos, and I. Androutsopoulos. "A Survey on Biomedical Image Captioning". In: *Workshop on Shortcomings in Vision and Language of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Minneapolis, MN, USA, 2019, pp. 26–36.

[KPA19b] V. Kougia, J. Pavlopoulos, and I. Androutsopoulos. "AUEB NLP Group at ImageCLEFmed Caption 2019". In: *CLEF2019 Working Notes*. CEUR Workshop Proceedings. Lugano, Switzerland: CEUR-WS.org, 2019.

[KPA20] V. Kougia, J. Pavlopoulos, and I. Androutsopoulos. "Medical Image Tagging by Deep Learning and Retrieval". In: *Experimental IR Meets Multilinguality, Multimodality, and Interaction Proceedings of the Eleventh International Conference of the CLEF Association (CLEF 2020)*. Thessaloniki, Greece, 2020.

[Kum+16] A. Kumar, S. Dyer, J. Kim, et al. "Adapting content-based image retrieval techniques for the semantic annotation of medical images". In: *Computerized Medical Imaging and Graphics* 49 (2016), pp. 37–45.

[KVB88] N. Kanopoulos, N. Vasanthavada, and R.L Baker. "Design of an image edge detection filter using the Sobel operator". In: *IEEE Journal of Solid-State Circuits* 23.2 (1988), pp. 358–367.

[KW14] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations ICLR*. Banff, AB, Canada, 2014.

[KW17] T. N. Kipf and M. Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. Toulon, France, 2017.

[KW52] J. Kiefer and J. Wolfowitz. "Stochastic Estimation of the Maximum of a Regression Function". In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466.

[LeC+89] Y. LeCun, B. Boser, J. S. Denker, et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551.

[Lem12] C. Lemaréchal. "Cauchy and the Gradient Method". In: *Documenta Mathematica* Extra (2012), pp. 251–254.

[Li+18] Y. Li, X. Liang, Z. Hu, and E. P. Xing. "Hybrid Retrieval-Generation Reinforced Agent for Medical Image Report Generation". In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018.

[Li+22a] C. Li, J. Yang, P. Zhang, et al. "Efficient Self-supervised Vision Transformers for Representation Learning". In: *International Conference on Learning Representations*. Online, 2022.

[Li+22b] Y. Li, F. Liang, L. Zhao, et al. "Supervision Exists Everywhere: A Data Efficient Contrastive Language-Image Pre-training Paradigm". In: *International Conference on Learning Representations*. Online, 2022.

[Li+22c] Y. Li, C. Wu, H. Fan, et al. "MViTv2: Improved Multiscale Vision Transformers for Classification and Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, LA, USA, 2022, pp. 4804–4814.

[Lin+17] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. "Focal Loss for Dense Object Detection". In: *IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2999–3007.

[Lit+17] G. Litjens, T. Kooi, B. E. Bejnordi, et al. "A survey on Deep Learning in Medical Image Analysis". In: *Medical Image Analysis* 42 (2017), pp. 60–88.

[Liu+21] Z. Liu, Y. Lin, Y. Cao, et al. "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Online, 2021, pp. 10012–10022.

[Liu+22] Z. Liu, H. Mao, C. Wu, et al. "A ConvNet for the 2020s". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 11976–11986.

[Lu+19] J. Lu, D. Batra, D. Parikh, and S. Lee. "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks". In: *Advances in Neural Information Processing Systems*. Vol. 32. Vancouver, Canada, 2019.

[Ma+21] Q. Ma, C. Yuan, W. Zhou, and S. Hu. "Label-Specific Dual Graph Neural Network for Multi-Label Text Classification". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online, 2021, pp. 3855–3864.

[MG11] I. Martinovic and H. Greve. "Surgical reconstruction of the left main coronary artery with patch-angioplasty". In: *J Cardiothorac Surg* 6 (2011), p. 24.

[Mik+13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. 2013.

[NF16] M. Noroozi and P. Favaro. "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles". In: *Computer Vision – ECCV 2016*. Amsterdam, Netherlands, 2016, pp. 69–84.

[OLV18] A. van den Oord, Y. Li, and O. Vinyals. *Representation Learning with Contrastive Predictive Coding*. 2018. eprint: `arXiv:1807.03748`.

[OVK17] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. "Neural Discrete Representation Learning". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[Pel+18] O. Pelka, S. Koitka, J. Rückert, F. Nensa, and C. M. Friedrich. "Radiology Objects in COntext (ROCO): A Multimodal Image Dataset". In: vol. 11043. Lecture Notes in Computer Science (LNCS), 2018, pp. 180–189.

[Pel+21] O. Pelka, A. Ben Abacha, A. García Seco de Herrera, et al. "Overview of the ImageCLEFmed 2021 Concept and Caption Prediction Task". In: *CLEF2021 Working Notes*. CEUR Workshop Proceedings. Bucharest, Romania: CEUR-WS.org, 2021.

[PGJ18] M. Pagliardini, P. Gupta, and M. Jaggi. "Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, LA, USA, 2018, pp. 528–540.

[Pha+21] H. Pham, Z. Dai, Q. Xie, M. Luong, and Q. V. Le. "Meta Pseudo Labels". In: *IEEE Conference on Computer Vision and Pattern Recognition*. Online, 2021, pp. 11557–11568.

[Phu+15] T. Phulambrikar, S. Vilas Kant, M. Kode, and S. Magar. "Cone Beam Computed Tomography Findings in Calcifying Cystic Odontogenic Tumor Associated with Odontome: A Case Report". In: *J Dent (Shiraz)* 16.4 (2015), pp. 374–379.

[PRS14] B. Perozzi, A. Rami, and S. Skiena. "DeepWalk: Online Learning of Social Representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. New York, NY, USA, 2014.

[Rad+21] A. Radford, J. W. Kim, C. Hallacy, et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. Online, 2021, pp. 8748–8763.

[Raj+17] P. Rajpurkar, J. Irvin, K. Zhu, et al. *CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning*. 2017. eprint: arXiv:1711.05225.

[Raj+18] P. Rajpurkar, J. A. Irvin, A. Bagul, et al. "MURA Dataset: Towards Radiologist-Level Abnormality Detection in Musculoskeletal Radiographs". In: *Medical Imaging with Deep Learning*. Amsterdam, Netherlands, 2018.

[Ram+21] A. Ramesh, M. Pavlov, G. Goh, et al. "Zero-Shot Text-to-Image Generation". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. 2021, pp. 8821–8831.

[Rey19] Anh H. Reynolds. *Convolutional Neural Networks (CNNs)*. 2019. URL: https://anhreynolds.com/blogs/cnn.html (visited on Oct. 10, 2022).

[RHW86] D. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536.

[RM51] H. Robbins and S. Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407.

[Ros57] F. Rosenblatt. *The Perceptron, a Perceiving and Recognizing Automaton*. Cornell Aeronautical Laboratory, 1957.

[Ros63] F. Rosenblatt. "Principles of Neurodynamics. Perceptrons and the theory of brain mechanisms." In: *American Journal of Psychology* 76 (1963), p. 705.

[RTC19] F. Radenović, G. Tolias, and O. Chum. "Fine-Tuning CNN Image Retrieval with No Human Annotation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41 (2019), pp. 1655–1668.

[Rüc+22] Johannes Rückert, Asma Ben Abacha, Alba García Seco de Herrera, et al. "Overview of ImageCLEFmedical 2022 – Caption Prediction and Concept Detection". In: *CLEF2022 Working Notes*. CEUR Workshop Proceedings. Bologna, Italy, 2022.

[Sar+22] S. M. Sarwar, D. Zlatkova, M. Hardalov, et al. "A Neighborhood Framework for Resource-Lean Content Flagging". In: *Transactions of the Association for Computational Linguistics* 10 (2022), pp. 484–502.

[Sha+21] A. K. Sharma, A. Nandal, A. Dhaka, and R. Dixit. "Medical Image Classification Techniques and Analysis Using Deep Learning Networks: A Review". In: *Health Informatics: A Computational Perspective in Healthcare*. Singapore: Springer Singapore, 2021, pp. 233–258.

[SLA12] J. Snoek, H. Larochelle, and R. P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems*. Vol. 25. Lake Tahoe, NV, USA, 2012.

[Soh16] K. Sohn. "Improved Deep Metric Learning with Multi-class N-pair Loss Objective". In: *Advances in Neural Information Processing Systems*. Vol. 29. 2016.

[Son+20] R. Sonker, A. Mishra, P. Bansal, and A. Pattnaik. "Techniques for Medical Concept Detection from Multi-Modal Images". In: *CLEF2020 Working Notes*. CEUR Workshop Proceedings. Thessaloniki, Greece: CEUR-WS.org, 2020.

[Sow+21] H. Sowrirajan, J. Yang, A. Y. Ng, and P. Rajpurkar. "MoCo Pretraining Improves Representation and Transferability of Chest X-ray Models". In: *Medical Imaging with Deep Learning*. Lübeck, Germany, 2021.

[Sri+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1929–1958.

[SSS08] B. Shetty, P. K. Shetty, and P. Sharma. "Ascariasis cholecystitis: An unusual cause". In: *J Minim Access Surg* 4.4 (Oct. 2008), pp. 108–110.

[SWD22] X. Su, R. Wang, and X. Dai. "Contrastive Learning-Enhanced Nearest Neighbor Mechanism for Multi-Label Text Classification". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Dublin, Ireland, 2022, pp. 672–679.

[TL19] M. Tan and Q. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Long Beach, CA, USA, 2019, pp. 6105–6114.

[TL21] M. Tan and Q. Le. "EfficientNetV2: Smaller Models and Faster Training". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. 2021, pp. 10096–10106.

[Tou+21a] H. Touvron, M. Cord, M. Douze, et al. "Training data-efficient image Transformers & Distillation Through Attention". In: *International Conference on Machine Learning*. Vol. 139. Online, 2021, pp. 10347–10357.

[Tou+21b] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou. "Going Deeper With Image Transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Online, 2021, pp. 32–42.

[Tu+22] Z. Tu, H. Talebi, H. Zhang, et al. "MaxViT: Multi-Axis Vision Transformer". In: (2022). eprint: arXiv:2204.01697.

[Vas+17] A. Vaswani, N. Shazeer, N. Parmar, et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. Long Beach, CA, USA, 2017, pp. 5998–6008.

[Vel+19] P. Velickovic, W. Fedus, W. L. Hamilton, et al. "Deep Graph Infomax". In: *7th International Conference on Learning Representations, ICLR*. New Orleans, LA, USA, 2019.

[VS17] L. Valavanis and S. Stathopoulos. "IPL at ImageCLEF 2017 Concept Detection Task". In: *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, 2017*. Vol. 1866. Dublin, Ireland, 2017.

[Wan+17] X. Wang, Y. Peng, Le. Lu, et al. "ChestX-ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA, 2017, pp. 3462–3471.

[WH18] Y. Wu and K. He. "Group Normalization". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[Wil+12] D. Wilken, X. Baur, L. Barbinova, et al. "What are the benefits of medical screening and surveillance?" In: *European Respiratory Review* 21.124 (2012), pp. 105–111.

[Wor+22] M. Wortsman, G. Ilharco, S. Y. Gadre, et al. "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. 2022, pp. 23965–23998.

[Wu+18] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. "Unsupervised Feature Learning via Non-Parametric Instance Discrimination". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Salt Lake City, UT, USA, 2018.

[Wu+21] H. Wu, B. Xiao, N. Codella, et al. "CvT: Introducing Convolutions to Vision Transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Online, 2021, pp. 22–31.

[Xie+20] Q. Xie, M. T. Luong, E. Hovy, and Q. V. Le. "Self-Training With Noisy Student Improves ImageNet Classification". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Online, 2020.

[Xie+22] Z. Xie, Z. Zhang, Y. Cao, et al. "SimMIM: A Simple Framework for Masked Image Modeling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, LA, USA, 2022, pp. 9653–9663.

[XL01] Q. Xu and Y. Z. Liang. "Monte Carlo Cross Validation". In: *Chemometrics and Intelligent Laboratory Systems* 56 (2001), pp. 1–11.

[Xu+21] Y. Xu, Q. Zhang, J. Zhang, and D. Tao. "ViTAE: Vision Transformer Advanced by Exploring Intrinsic Inductive Bias". In: *Advances in Neural Information Processing Systems*. Vol. 34. Online, 2021, pp. 28522–28535.

[Yao+22] L. Yao, R. Huang, L. Hou, et al. "FILIP: Fine-grained Interactive Language-Image Pre-Training". In: *International Conference on Learning Representations*. Online, 2022.

[Yon+20] H. Yong, J. Huang, X. Hua, and L. Zhang. "Gradient-Centralization: A New Optimization Technique for Deep Neural Networks". In: (2020), pp. 635–652.

[Yu+22] J. Yu, Z. Wang, V. Vasudevan, et al. *CoCa: Contrastive Captioners are Image-Text Foundation Models*. 2022. eprint: `arxiv.2205.01917`.

[Zbo+21] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. "Barlow Twins: Self-Supervised Learning via Redundancy Reduction". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. Online, 2021, pp. 12310–12320.

[ZGR22] E. Ben Zaken, Y. Goldberg, and S. Ravfogel. "BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Dublin, Ireland, 2022, pp. 1–9.

[Zha+18] W. Zhang, J. Yan, X. Wang, and H. Zha. "Deep Extreme Multi-Label Learning". In: *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 100–107.

[Zha+19] Y. Zhang, Q. Chen, Z. Yang, H. Lin, and Z. Lu. "BioWordVec, improving biomedical word embeddings with subword information and MeSH". In: *Scientific Data* 6.52 (2019).

[Zha+21] Y. Zhang, H. Jiang, Y. Miura, C. D. Manning, and C. Langlotz. *Contrastive Learning of Medical Visual Representations from Paired Images and Text*. 2021.

[Zha+22] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. "Scaling Vision Transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, LA, USA, June 2022, pp. 12104–12113.

[Zhe+21] X. Zheng, Z. Zhang, J. Guo, et al. "Adaptive Nearest Neighbor Machine Translation". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, 2021, pp. 368–374.

[Zon+21] F. Zong, H. Shenghua, R. Su, W. Xiaowei, and L. Hua. "Deep learning-based multi-class COVID-19 classification with X-ray images". In: 11598 (2021). Ed. by Cristian A. Linte and Jeffrey H. Siewerdsen, pp. 286–296.

List of Figures

2.1 An illustration of a biological neuron with annotations corresponding to the function of an artificial neuron. Figure taken from [KC+21]. 5

2.2 An illustration of how Gradient Descent tries to reach the minimum of the optimization space by iteratively updating the weight parameters. 6

2.3 An MLP for handwritten digit recognition. There are four layers (the input layer, two hidden layers and the final/output layer). The final layer receives the image representation vector (i.e., the output vector of the last hidden layer) as an input and outputs a probability distribution showing the probability of the input image belonging in each category. Figure taken from [Che+20b]. . 10

2.4 An illustration of the convolution operation. The kernel slides over the image and the dot product between the current patch and the kernel is calculated. In CNNs, the kernel is a weight matrix which is updated during training. The remaining values of the output tensor are calculated likewise. Figure taken from [Rey19]. 11

2.5 Max pooling operation. f denotes the window size and s the stride. Max pooling downsamples the input by keeping only the max value of the window. Figure taken from [Rey19]. 12

2.6 (Top) When $\vec{s}$ is formed by convolution with a kernel of width 3, only three inputs affect s_3. (Bottom) When $\vec{s}$ is formed by matrix multiplication, connectivity is no longer sparse, so all the inputs affect s_3. However, even though direct connections in a CNN are very sparse, units in the deeper layers can be indirectly connected to all or most of the input image. Figure taken from [GBC16]. 13

2.7 (Top) The black arrows indicate uses of the central element of a 3-element kernel in a convolutional model. Because of parameter sharing, this single parameter is used at all input locations. (Bottom) The single black arrow indicates the use of the central element of the weight matrix in an MLP. This model has no parameter sharing, so x_3 affects only s_3. Figure taken from [GBC16]. 14

2.8 (Left) The Scaled Dot-Product QKV Self Attention. Q, K, V are vectors. (Right) In practice, the operation is repeated many times and the results are concatenated (Multi-Head Attention). Figure taken from [Vas+17]. 15
2.9 The Transformer architecture. (Left) The encoder of the sequence. (Right) The decoder which predicts the next token (Transformers were initially introduced for machine translation). The decoder has two self-attention components. Masked self-attention which does not 'look' future tokens and a (non-masked) cross-attention mechanism in order to attend over the encoder outputs. Figure taken from [Vas+17]. 16
2.10 A high-level view of ViT. An image is divided in non-overlapping patches which are linearly projected and fed to the encoder in order to compute the feature vector. The encoder is a Transformer network as described in [Vas+17]. The class ([CLS]) embedding is fed to an MLP in order to be classified. Figure taken from [Dos+21]. 17

3.1 Illustration of the SAM parameter update. Vanilla SGD update is denoted by $\boldsymbol{w}_{t+1}$ while final SAM update is denoted by $\boldsymbol{w}_{t+1}^{SAM}$. Figure taken from [For+21]. 30
3.2 Geometrical interpretation of GC. The projected gradient is used to update the weights. Figure taken from [Yon+20]. 31
3.3 Left: Illustration of the VQ-VAE. Right: Visualisation of the embedding space. Since the quantization process is not differentiable, a straight-through estimator in between the decoder and the encoder is applied, so that the decoder gradients are directly propagated to the encoder. Other techniques such as gumbel softmax relaxation [JGP17] can be used. Figure taken from [OVK17]. 34
3.4 Illustration of how the retrieval-based k-NN system works at inference time. The training image embeddings are computed offline. Figure adapted from [KPA19b]. 40

4.1 Three images of the dataset [SSS08; MG11; Phu+15] (1st row) with their corresponding ground truth concepts (2nd row) and captions (3rd row). . . . 47
4.2 Number of images (vertical axis) tagged with each concept in the training data. To save space, the horizontal axis shows the index of each concept (class index), instead of its CUI. 48

List of Tables

4.1 The 10 most frequent concepts (CUIs and corresponding UMLS terms) in the dataset of ImageCLEFmedical Caption 2022 and how many images they are assigned to. 47
4.2 Scores comparing a non-linear vs. a linear classifcation head. 49
4.3 Scores comparing backbone visual encoders. 50
4.4 Scores comparing different loss functions. 50
4.5 Scores comparing different optimization methods. 51
4.6 Scores comparing different ensemble models. 52
4.7 Scores comparing experiments with self-supervised methods. 53
4.8 Scores comparing different label-space aware experiments. 54
4.9 Comparison of concepts before and after self-training. 55
4.10 Scores comparing different encoders combined with k-NN. 56
4.11 Scores comparing different encoder combined with k-NN. 56
4.12 Scores comparing retrieval-augmented classification experiments. 57
4.13 Scores and rankings of our submitted systems on our final development and the official test sets. cd3's development score is not available, as its predictions were created using the output files of the ensemble's members. . 58

List of Algorithms

1 GreedySoup 32
2 Self-training with Teacher-Student (for specific labels) 39