UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Universidad Autónoma
de Madrid

Master in Deep Learning for
Audio and Video Signal Processing

MASTER THESIS

SELF-SUPERVISED CURRICULAR DEEP LEARNING FOR CHEST X-RAY IMAGE CLASSIFICATION

Iván de Andrés Tamé
Advisor: Pablo Carballeira López

June 2022

SELF-SUPERVISED CURRICULAR DEEP LEARNING FOR CHEST X-RAY IMAGE CLASSIFICATION

Iván de Andrés Tamé
Advisor: Pablo Carballeira López

Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
June 2022

Video Processing and Understanding Lab
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politéctnica Superior
Universidad Autónoma de Madrid
June 2021

Trabajo parcialmente financiado por la Consejería de Educación e Investigación de la Comunidad de Madrid bajo el proyecto SI1/PJI/2019-00414 (Aiding diagnosis by self-supervised deep learning from unlabeled medical imaging).

Resumen

Las técnicas de Machine Learning son muy implementadas para clasificar imágenes. Una de las principales desventajas de estas soluciones es que para realizar estas predicciones, es necesario tener un gran conjunto de datos etiquetados con el que entrenar el modelo. Sin embargo, hay campos en los que no hay muchas bases de datos etiquetados, o las etiquetas son de baja calidad, como por ejemplo con datos médicos. Un enfoque popular para trabajar con datos sin etiquetar es preentrenar el modelo usando aprendizaje autosupervisado (SSL). Este pre-entrenamiento funciona dando a los datos una pseudoetiqueta basada en una tarea que el modelo tiene que resolver.

En este trabajo, implementamos un pre-entrenamiento de SSL que combina secuencialmente diferentes tareas pretexto. Entrenaremos nuestro modelo sobre la base de datos SIIM-FISABIO-RSNA que contiene radiografías de pulmón. Con esto, queremos comprobar si combinar diferentes tareas funciona mejor que usar solo una, así como probar si esta solución puede adaptarse bien al campo médico. También representaremos los diferentes mapas de atención de cada modelo para que podamos comprobar dónde centra la atención cada tarea.

Palabras clave

Aprendizaje auto-supervisado, Inteligencia artificial, Aprendizaje profundo, Mapas de atención, Combinacion de tareas, Imágenes médicas

Abstract

Machine learning techniques are widely implemented to classify images. One of the main disadvantages of these solutions is that to perform these predictions, it is necessary to have an extensively annotated dataset with which to train the model. However, there are fields in which there are not many labeled databases, or the labeling has low quality, i.e. with medical data. A popular approach to working with unlabeled data is using Self Supervised Learning (SSL) pretraining. This pretraining is based on giving the data a pseudo label based on a pretrain task that the model has to solve.

In this work, we implement a SSL pretraining that combines sequentially different pretext tasks and train it over the SIIM-FISABIO-RSNA dataset of lung radiographs. With this, we want to check if combining different tasks performs better than using only one, as well as test if this solution can adapt well to the medical domain. We also represent the different attention maps of each model so we can check where every SSL task focuses the attention.

Keywords

Self supervised learning, Machine learning, Deep learning, Attention maps, Combined tasks, Medical Image

Acknoledgements

I would like to thank the VPULab, without whom this could not be possible. Thanks to the different professors that gave me support during the Master and taught me all I needed to do this work. I would like to thank specially to my tutor Pablo Carballeira López for giving me the chance to do this project under his supervision.

I would also like to thank my family, which has supported me during all my student years, especially my parents and my grandmother.

Last but not least, doing this work would have been much harder without my friends, with whom I have worked side by side helping each other. Special thanks to Pablo and Julio who have helped me by staying with me all these years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The medical field is constantly evolving. It must be at the forefront of new techniques that can help both medical professionals and patients. In an increasingly technological world, these new technologies should be exploited by Medicine. Nowadays, this technological transformation is making that all the data that previously was recorded by hand, is transformed into digital mediums. Typical radiographs images that were always recorded over a polyester film, nowadays can be stored in multiple formats in a computer, making that the amount of data that we have in this field, has never been so large.

Deep learning (DL) is one of the branches of Artificial Intelligence (AI) that has seen exponential growth in recent years and benefits a lot from having these larges amounts of data. Deep learning solutions yield close-to-human accuracy for challenging vision tasks, like classifying or detecting an object. These solutions rely on large hand-annotated datasets to train the models. However, the labeling of these datasets is a heavy burden and scarce and expensive expertise is required for high-quality annotation of some domains such as medical imaging. Self-supervised learning (SSL) has proven to be a successful strategy to cope with this problem, as with SSL, the model can learn from unlabeled data.

Even though the amount of medical data has never been so large, traditional DL solutions require databases with various thousands or even millions of labeled data images to learn. Also, due to privacy regulations, medical data is difficult to obtain and treat. With this in mind, the main objective of this work was to apply state-of-the-art techniques based on SSL to a set of medical images. With this, we want to test the feasibility of using these techniques over this specific data, and to search for the best configuration of the used model for this problem.

This work has been possible thanks to the collaboration of the VPULab (Video Processing and Understanding Laboratory), which provided us access to a GPU to perform the experiments.

1.2 Objectives

This master's thesis focuses on implementing an SSL model that combines multiple pretext tasks in a sequential order to create a more complex and accurate visual representation of the data. After this pretraining, a classification step is performed with some labeled data, and we test its performance over medical images, specifically images from lungs. Upon literature reviews, and careful observations and discussions, the following points are set to achieve within the scope of this master's thesis.

- Collect a dataset of lung x-ray images for the objective of this master thesis.
- Perform semantic segmentation over it, to have both the full image and the segmented one only containing the lung regions.
- Set up and run different self-supervised learning pretext tasks and check their performance individually.
- Implement an SSL model that combines multiple pretext tasks in sequential order and test its performance over the full and segmented data.
- Obtain results from the training of this model and compare between tasks and type of images.
- Visualize where the models are focusing their attention to perform a quality analysis.
- Search for an optimum sorting of the pretext tasks for the targeted task.

1.3 Report structure

This report has the following chapters

- Chapter 1 Introduction: Motivation and thesis' objectives. In this chapter, we give an overview of the thesis.
- Chapter 2 Related work: Basic concepts about medical imaging, its problems, deep learning, and how it may benefit the medical field. In this chapter, we talk about current trends in Deep learning that we used, as well as, talk about the proposed database.
- Chapter 3 Design and development: In this chapter, we explain how we implemented the proposed solution for this project.
- Chapter 4 Experiments and results: In this chapter, we explain the different experiments that we performed for this project as well as the obtained results.
- Chapter 5 Conclusions and future work.

Chapter 2

Related Work

State-of-the-art deep learning techniques often require training over large and varied datasets. If not, the trained model may not be able to learn quality visual representations. Therefore, not having enough data for your specific task is the main point of not using deep learning techniques for solving the problem.

Acquiring this data usually is an expensive process in which multiple steps need to be performed before is ready to be used by a model. One of the most extensive and tedious steps is labeling the data. This process is even harder when the data is from a field in which is necessary an expert to perform the labeling. Therefore, obtaining a big and varied database of medical images is expensive and complicated.

To surpass this problem, most of the models are pretrained in general-purpose datasets with lots of labeled data, like the Imagenet dataset, which is an image database containing more than fourteen million annotated images with more than twenty thousand categories [8], and then fine-tuned with the target data, thus achieving that the model has already some information about how to interpret the images and making our training faster and more efficient.

In this chapter we explore more in-depth how to obtain medical images as well as why is useful to analyze them (Section 2.1). After that, we explain how deep learning solutions are used for different tasks like image classification (Section 2.2) and how these methods can be applied to medical images along with the limitations that this field presents (Section 2.3). Then, we present the state-of-the-art approaches that are used to solve the mentioned problems (Section 2.4). Finally, we present the evaluation frameworks that we used to validate this project, being these the contemplated databases and the evaluation metrics (Section 2.5).

2.1 Automatic Medical Image Analysis

Most medical issues occur inside the body, so making a diagnosis can be challenging. Being able to take an image from inside the body has become easier over the last century. Medical imaging is the technique of producing visual representations of areas inside the human body to diagnose medical problems and monitor treatment. Some examples of different types of medical images are radiography, magnetic resonance

Figure 2.1: Radiography from a thorax. We can see white in the hardest tissues (bones) and black in the lungs.

imaging (MRI), ultrasound, and nuclear medicine. For this work, the method that we analyzed is the radiographs.

Radiography uses electromagnetic radiation to take images from the inside of the body. The most common form of radiography is the x-ray. For this procedure, an x-ray machine beams high-energy waves onto the body. The soft tissues, such as skin and organs, do not absorb these waves, whereas the hard ones like bones do absorb the waves. The machine transfers the results of the x-ray onto a film, showing the bones in white and leaving the unabsorbed materials in black. We can see an example of radiography in Figure 2.1.

Thanks to being able to capture these images, doctors are capable to detect some diseases faster than if they had to rely upon older approaches. For example in the case of suffering pneumonia, there are cases in which we can detect at first sight that there is something atypical (Figure 2.2). To search for these oddities, typically an expert doctor needed to analyze them, to know the source of the disease. After it, the doctor had to consider whether it was the right decision without supporting information. Even though there still needs to be doctors to perform the final diagnosis, every day there are more support tools to help the doctors do their diagnosis, speeding up the whole process. Some of this solutions are explained more in-depth in the Deep learning solutions for medical images Section (2.3).

2.2 Deep Learning and Image Processing

Artificial intelligence (AI) refers to the ability of machines to learn how to perform a task, without specific indications provided by their designer. Machine learning (ML), is a subset of AI in which the algorithm that gives intelligence to the machine try to learn data patterns as it is exposed to more data over time. Finally, deep learning (DL) is a subset of ML in which a multilayered neural network learns from a vast amount of data.

Figure 2.2: Radiography from three different thoraces, in which we can see white opacities inside the lungs that are not present in the healthy patient.

In recent years, deep learning has become more and more popular in research and has been incorporated into a large number of applications, including image classification, video recommendation, social network analysis, natural language processing, and so forth. [9, 10, 11]

One of the most popular and used architectures of deep neural networks are the convolutional neural networks (CNN) [12, 13, 14]. A CNN can learn convolutional filters with which it can obtain efficient visual representations from the input data. The CNN convolves the learned features with the input data employing 2D convolutional layers, which makes this architecture suitable for processing 2D data, such as images. These networks' real strength is the ability to perform feature extraction from images without any need to explicitly program them to extract them. We can see an example of how CNN works in Figure 2.3 for a classification task. As we increase the number of layers that conform to the network, this gains complexity. The first layers can recognize simpler patterns as geometrical shapes, colors, or borders, while the deeper ones recognize more complex forms. As the complexity is increased, the capacity to obtain visual representations and the time the network requires to learn also increase.

We can increase the complexity of a model by changing the architecture of the model itself (i.e. AlexNet [8], VGG [15], ResNet [16], Inception [17], etc) or building a deeper model [15]. When building these more complex models, the number of parameters tends to increase along with the complexity of the model, which may be better for some complex problems but worse if we need our model to train fast. We can see this effect in Figures 2.4 and 2.5, and it increases, even more, when the depth of the net raises.

Alternatively, another way to improve the accuracy is by building a hybrid model by either combining different architectures or by using the same network trained multiple times with different initialization [18, 19].

Convolutional neural networks' performance has been proven in different scenarios, in which they exceed or are proximate to human performance [20, 21]. Current state-of-the-art on Imagenet [8] classification is over 90% as we can see in Figure 2.6 while a trained human annotator has an error rate of the 5,1% [22]. Even though humans can perform better in this task, another potential of CNN is the amount of time they need to do their prediction. Human annotators take 6 seconds on average, while CNN

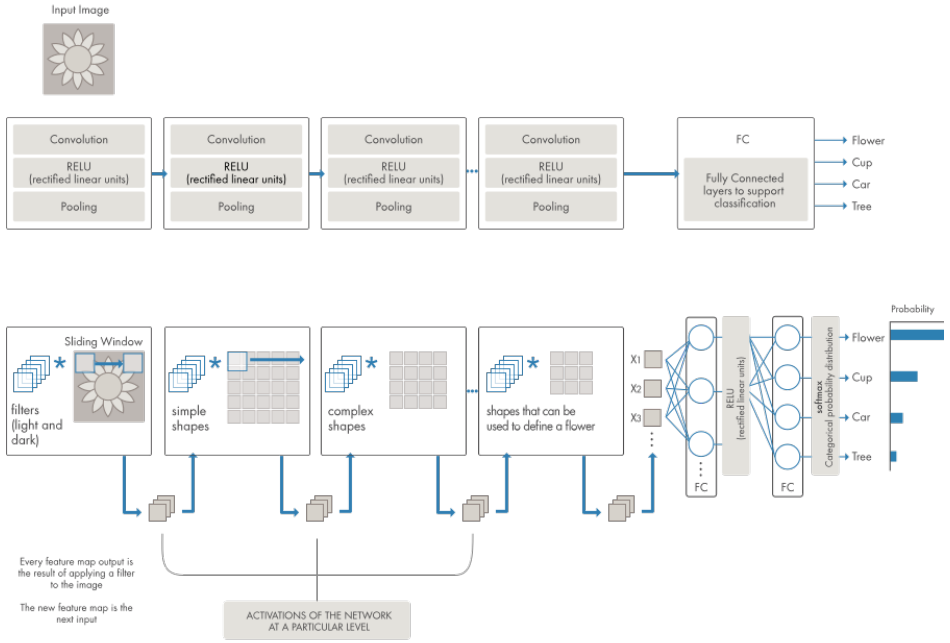


Figure 2.3: Example of a network with many convolutional layers. Filters are applied to each training image with different resolutions, and the output of each convolutional image is used as input for the next layer.

Comparison					
Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B

Figure 2.4: Comparison between different architectures over Imagenet [1]

takes a couple of milliseconds. This gives deep learning models the capacity to be implemented in applications that require above human performance in time.

2.3 Deep Learning for Medical Images

Research in deep learning solutions for medical images bears many promises to improve patients' health. However, some challenges are slowing down the progress of the field, such as the limitations of the data, the biases, or the difficulties to share patients' information. As this work uses X-rays, in this section we explain various kinds of radiologic applications.

2.3.1 Deep Learning Challenges

While deep learning has seemingly limitless potential to gather new insights from medical images, implementing the technology in clinical settings comes with some major barriers. The two main problems that medical imaging faces are the lack of quality

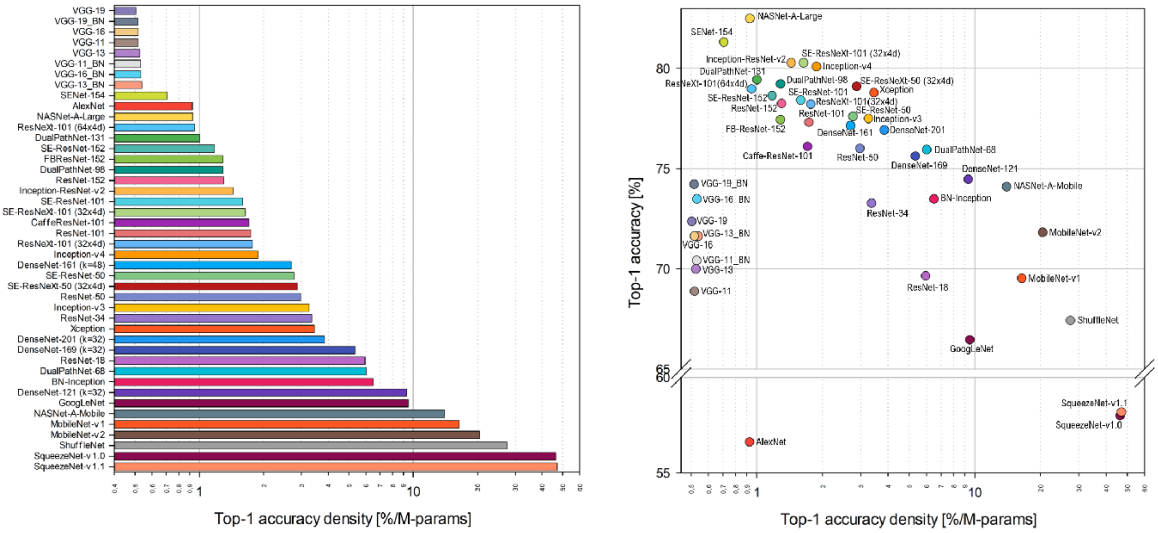


Figure 2.5: Top-1 models accuracy vs. computational complexity (# of parameters). [2]

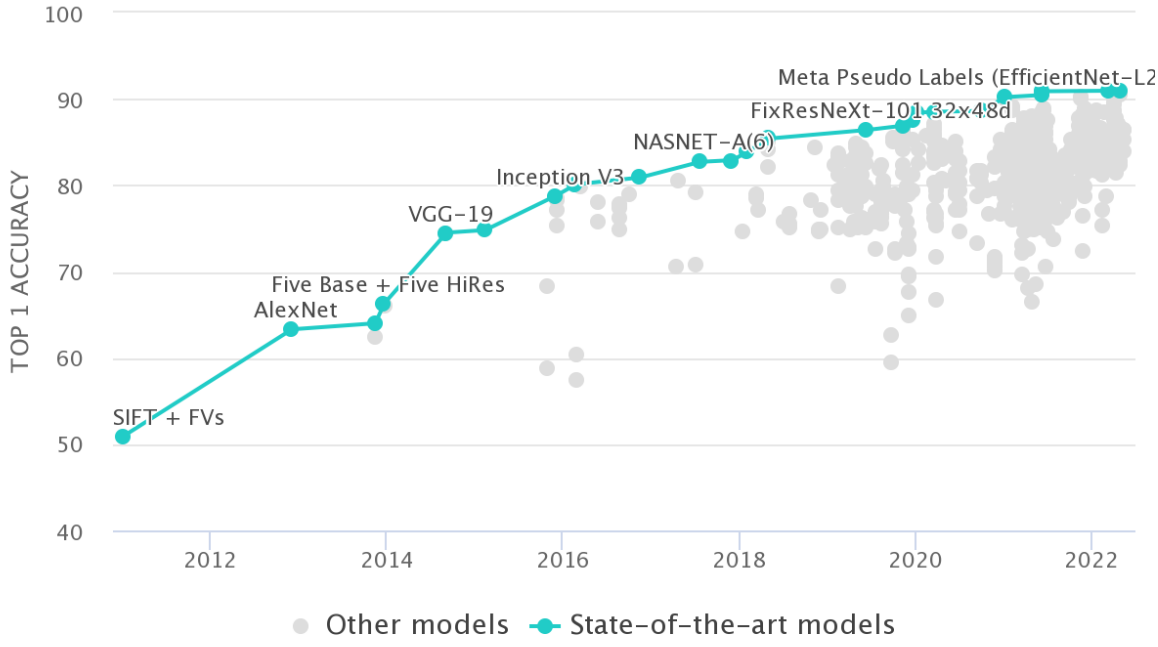


Figure 2.6: Imagenet benchmark

Name	Number of images
CheXpert	224,316
ChestX-Det-Dataset	3,578
NIH Chest X-rays	112,000
Segmentation in Chest Radiographs	247
Imagenet	14,000,000

Table 2.1: Comparison between the number of training images between medical datasets and Imagenet

databases, with a big enough training dataset, and the difficulty to share this data because of the privacy regulations [23]. A comparison between medical and other fields databases can be found in Table 2.1.

Structured, text electronic health record data are comparatively easy to aggregate and de-identify (prevent someone's identity from being revealed). Such data are commonly shared by health organizations with service providers or partner facilities for quality assessment or analytics. On the other hand, the distribution of medical images requires explicit patient consent.

2.3.2 Image Classification

Image classification is useful to radiologists for a wide range of applications from determining the presence or absence of disease to identifying the type of malignancy. Some works prove the efficiency of several CNN helping in the diagnosis of several diseases like retinopathy and skin cancers [24, 25]. One problem that deep learning models face while training with images is the complex patterns that these present. To learn faster these patterns, state-of-the-art solutions perform a pretraining over the network (explained in Section 2.4). Some works use the features obtained by the CNN to replace the radiomics from current models [26, 27]. Radiomics is a method that extracts a large number of features from medical images using data-characterization algorithms [28]. These features, termed radiomic features, are used to predict different complex diseases.

2.3.3 Object Detection

Object detection is identifying and locating objects in an image. In biomedical images, a detection technique is also performed to identify the areas where the patient's lesions are located as box coordinates. Object detection is split into two types. One is region-based, which consists of selecting different patches from the image and detecting the object inside these [29]. The other consists of using a one-stage network that performs directly the finding and detecting bounding box and probability from the image [30]. These solutions are generic for every object detection problem, but there are multiple solutions performed for specific medical image-related topics.

One of the main challenges in microscopic image analysis comes from the need of analyzing all individual cells for accurate diagnosis because the differentiation of most

disease grades highly depends on the cell-level information. Cireşan et al. [31] used deep CNN to detect mitosis in breast histology images. Su et al. [32] used sparse auto encoders and sparse representation to detect and segment cells from microscopic images. Xu et al. [33] also used sparse auto encoders to detect cells on breast cancer tissue images.

2.3.4 Image Generation

As stated at the beginning of the chapter, one of the main problems that using deep learning solutions with medical images, is the lack of labeled data as well as privacy issues. To overcome this issue, several studies exploit GANs (Generative Adversarial Networks) [34] to make realistic synthetic images of whole X-rays or regions of interest of specific lesions, such as liver or lung cancer [35, 36].

2.4 Training Trends With Limited Data

In this section, we explain different improvements that were applied to classic models to overcome the stated problems in Section 2.3. One of the most used techniques for the training of big models if there is not enough data available is transfer learning. This aims at improving the performance of target learners on target domains by transferring the knowledge from similar fields and it is often used when data is scarce or full-scale training is too costly [37, 38, 39]. These pretrained models are usually trained in big databases such as Imagenet. After you have a pretrained model, you adapt it to your dataset by training again over your target data. This whole process is called fine-tuning. However, one of the biggest limitations to transfer learning is the problem of negative transfer. Transfer learning only works if the initial and target problems are similar enough. If the problems are too different, the model may perform worse than if it had never been trained at all. For this, is usually a good idea to train over a big and varied database. Another problem transfer learning carries is that the bias from the initial target is also learned by the model. It has been proven models trained at Imagenet tend to focus their attention in the center of the image, as most of the regions of interest in this database are placed there (see Figure 2.7), or in the object texture more than its shape [40, 41].

2.4.1 Self-Supervised Learning Tasks

A popular approach to dealing with insufficient data is Self-Supervised Learning (SSL) pretraining. Self-supervised learning is a machine learning technique that relies on pretext tasks, which are problems that can be formulated using only unsupervised data. A pretext task is designed in a way that solving it requires learning a useful image representation [42, 11], therefore making it easier for the model to be trained over data that is similar to the one used to solve the SSL task. In this subsection, we explain different SSL pretext tasks.

We can split the SSL tasks into two main groups: contrastive and non-contrastive methods. Contrastive [43] methods work by contrasting samples against each other to learn attributes that are common between data classes and that set apart a data

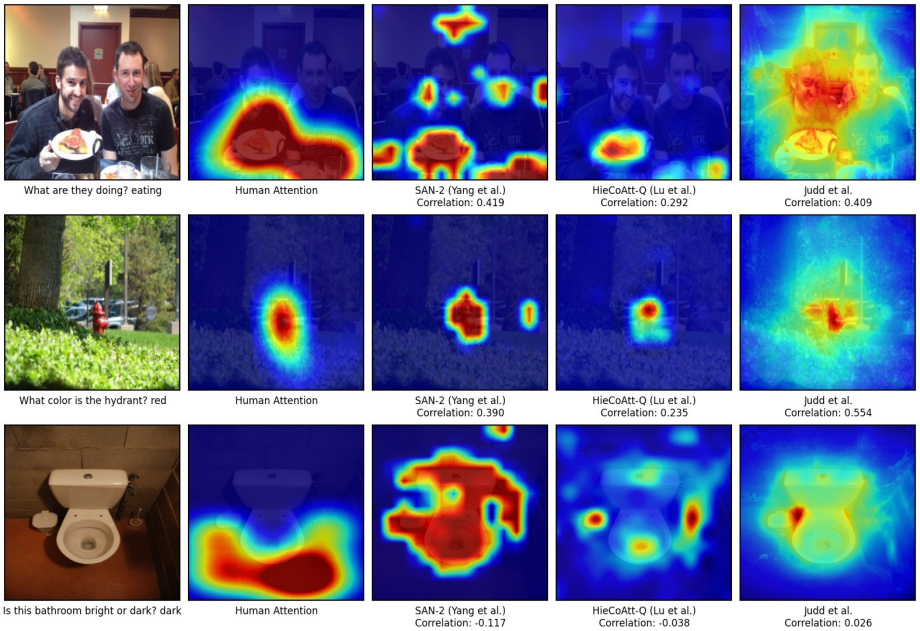
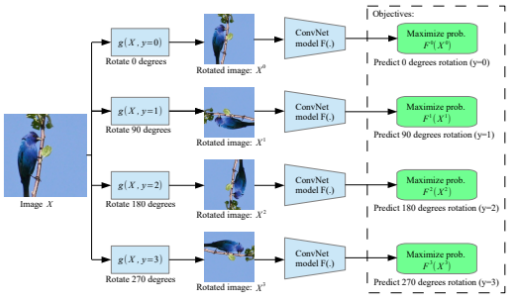


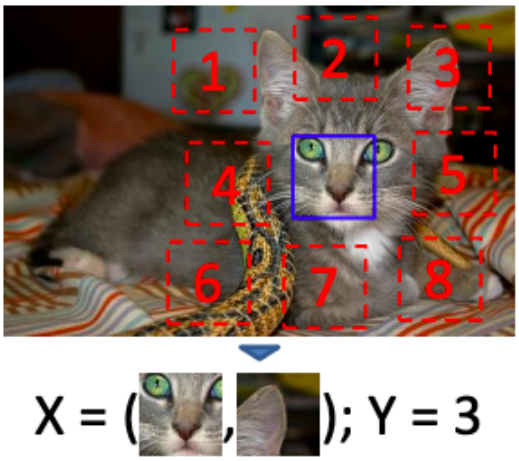
Figure 2.7: Random samples of human attention (column 2) vs machine-generated attention (columns 3-5). [3]

class from another. Chen et al. [44] proposed a contrastive method, which is based in the Momentum Contrast (MoCo) method proposed by He et al. [45]. This method works by assigning to every image a key, like in a dictionary. The key generated is the output of that image from a convolutional neural network (a vector). To check the similarity between two images and let the model learn, an image would be sent as a vector and would be assigned to the nearest key (it would be necessary then to have a way to calculate distances (see Figure 2.8c). Caron et al. [46] propose another contrastive method named SwAV (Swapping Assignment between Views). This method has the advantage over other contrastive methods because it does not have to compute comparisons between pairs. The method consists of taking an image, performing augmentations, and clustering them in a way that they are next to each other (see Figure 2.8d).

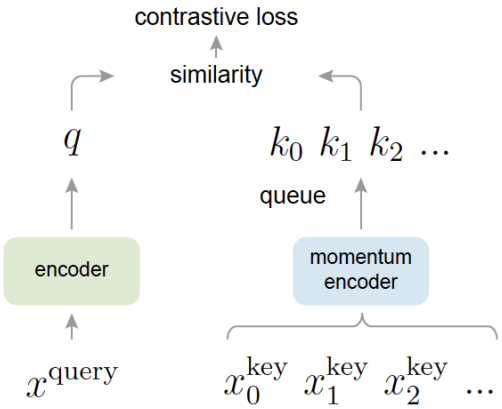
There are also non-contrastive methods, which do not try to compare pairs of images and search for solving a task that can be formulated with only one image. Doersch et al. proposed a method that trains a CNN model using the spatial information from the image. This task works by predicting the relative location of two randomly sampled non-overlapping image patches [47] (see Figure 2.8b). They claim that solving well this task requires that the model learns to recognize objects and their parts. Gidaris et al. suggested another method that does the training by rotating an image a certain amount of degrees and then using the transformed image as input. The model task is to predict the degrees that an image had rotated [48] (see Figure 2.8a). We can see in Table 2.2 a comparison of how some of these SSL tasks perform over imagenet.



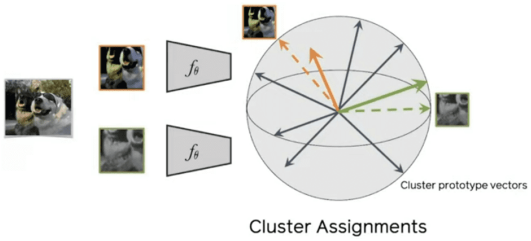
(a) Rotation prediction pretext task



(b) Relative location pretext task



(c) MoCoV2 pretext task



(d) SwAV pretext task

Figure 2.8: Different SSL pretext tasks

Training method	Accuracy over Imagenet
Supervised	77.15%
MoCoV2	83.2%
Swav	75.3%
Rotation prediction	70.87%

Table 2.2: Different SSL tasks evaluated over imagenet classification problem

2.4.2 Curriculum Learning

One popular method to deal with data that our model is not able to learn effectively is relying on curriculum learning. The idea of curriculum learning comes from the fact that concepts are easier to learn when seen in an orderly manner. This was implemented for machine learning algorithms by Elman et al. [49] and Bengio et al. [50], making the main idea to learn easier aspects of the problem earlier while gradually increasing the difficulty. There are also more general methods that do not require labeling the training data as harder or easier, e.g. supervising the learning by a teacher network (teacher-student) [51] or taking into consideration the learning progress of the model (self-paced learning) [52, 53]. This is different from the idea proposed by Bengio et al. [50] in the way samples are evaluated, as in this method the difficulty is measured during training.

Curriculum learning can be implemented along with SSL or transfer learning, as it only considers how the data is presented. Several works join these learning methods [54, 55, 56] and prove that the performance is increased with respect only using SSL or transfer learning.

2.4.3 Attention Maps

Visual attention refers to our capacity to focus our attention on an explicit region of our vision. When we are training a neural network, we need a way for it to explicitly incorporate the concept of visual attention. It needs to be able to learn where is the most important place to look to make distinctions between objects. Several works implement different methods to explicitly add attention to a model [57, 58, 59]. However, there is also implicit attention in every model, and there is no need to implement new modules to visualize it. The main idea behind being able to know where the model is focusing its attention is to use the intermediate layers (usually next to the final layers of our model), which contain visual representations relevant to performing the final classification and being able to relate them to the original image.

There are multiple ways of visualizing the model's attention [60, 61, 62]. However, one of the more popular methods is obtaining the class activation map (CAM) [63]. These maps are generated from the final convolutional layer of CNN. They can highlight discriminative object regions for the class of interest.

2.4.4 Multiple SSL Task Training

In this subsection, we explain different approaches to solving the problem of classifying medical images. As we stated in Subsection 2.4.1, one of the current trends to solve problems in which there is a lack of labeled data is using self-supervised learning. Creating a model pretrained using this technique has been explored in several papers [64, 11, 42]. However, to add a layer of complexity to this solution some works propose to combine different pretext tasks [4, 65, 66, 67]. Different pretext tasks focus their attention on different areas of the image [68]. The intuition behind proposing a combination of pretext tasks is that the different information that each pretext task contributes may give a better pretraining than the information from only one.

Pre-training	ImageNet top1		ImageNet top5	PASCAL		NYU
	Prev.	Ours	Ours	Prev.	Ours	Ours
Relative Position	31.7	36.21	59.21	61.7	66.75	80.54
Color	32.6	39.62	62.48	46.9	65.47	76.79
Exemplar	-	31.51	53.08	-	60.94	69.57
Motion Segmentation	-	27.62	48.29	52.2	61.13	74.24
INet Labels	51.0	66.82	85.10	69.9	74.17	80.06

Table 2.3: Comparison of Doersch and Zisserman implementation with previous results on the evaluation tasks: ImageNet with frozen features (left), and PASCAL VOC 2007 mAP with fine-tuning (middle), and NYU depth (right).[4]

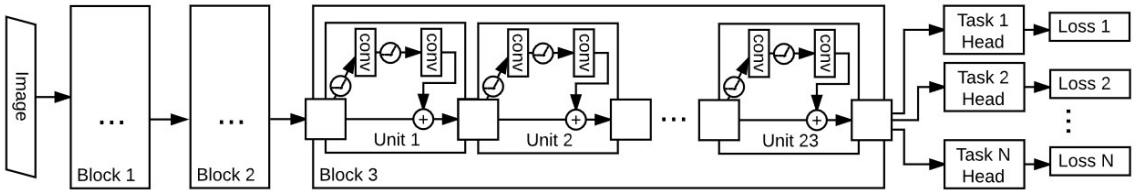


Figure 2.9: The structure of the multi-task network from [4]. It is based on ResNet-101 and each SSL task head is attached to the output of the network.

Doersch and Zisserman [4] after comparing four different self-supervised tasks individually, they combined them and trained a network with them joined. The method they proposed to do this combined training was to train the models in the different tasks and do a combined backpropagation to update the parameters. This process was done at batch size, i.e. each time a batch of training data is processed by every SSL task, they updated the network with a combination of the gradients. We can see in Figure 2.9 a diagram of the network. In contrast with our work, they trained their model over three different large datasets, Imagenet [8], PASCAL VOC [69], and NYU depth prediction [70] which have different objectives. We can see how the individual SSL task performed on each of these datasets in Table 2.3, while in Table 2.4 we can observe the performance of combining different SSL tasks. They begin all their combinations with the relative position task as the base task, as it is the one with a better performance individually. The results seen in these tables indicate that combining SSL tasks improves the performance regarding using only one.

Another method for pretraining a model with multiple SSL tasks was proposed by Kirill et al. [65]. In this work, they propose to train the same backbone by changing the SSL heads. The main difference is that in this work, the moment in which the SSL task is updated is when the model has been trained with the whole dataset. They tested this concept over the ISIC-2019 dataset [71, 72], which contains images of skins lessons from 8 different categories. In Table 2.5 we can see that with their method, they were able to beat the winner of the ISIC-2019 challenge, and as we can see with the column δ, most of the SSL tasks improve when combined.

Our work follows the steps performed in Kirill et al. paper, as the data target from their work is in the medical field as well as ours (Section 2.5). With this work we want

Pre-training	ImageNet	PASCAL	NYU
RP	59.21	66.75	**80.54**
RP+Col	66.64	68.75	79.87
RP+Ex	65.24	69.44	78.70
RP+MS	63.73	68.81	78.72
RP+Col+Ex	68.65	69.48	80.17
RP+Col+Ex+MS	**69.30**	**70.53**	79.25
INet Labels	85.10	74.17	80.06

Table 2.4: Comparison of various combinations of self-supervised tasks. Abbreviations: RP: Relative Position; Col: Colorization; Ex: Exemplar Nets; MS: Motion Segmentation. Metrics: ImageNet: Recall@5; PASCAL: mAP; NYU: % Pixels below 1.25. [4]

	1st task	2nd task	3rd task	Balanced accuracy (%)	δ (%)	Better than ImageNet
-	Rel. loc.			69.52	-	No
(AC)	Rel. loc.	ODC		70.68	1.16	No
(MC)	Rel. loc.	ODC	MoCo v2	75.00	5.49	**Yes**
(C)	Rel. loc.	MoCo v2		74.10	4.58	**Yes**
(MC)	Rel. loc.	MoCo v2	ODC	74.38	4.86	**Yes**
-	MoCo v2			72.74	-	No
(AC)	MoCo v2	ODC		72.72	-0.02	No
(MC)	MoCo v2	ODC	Rel. loc.	67.00	-5.74	No
(AC)	MoCo v2	Rel. loc.		66.72	-6.02	No
(AC)	MoCo v2	Rel. loc.	ODC	69.80	-2.95	No
-	ODC			63.52	-	No
(C)	ODC	Rel. loc.		68.23	4.71	No
(C)	ODC	Rel. loc.	MoCo v2	73.36	9.84	No
(C)	ODC	MoCo v2		75.44	11.92	**Yes**
(MC)	ODC	MoCo v2	Rel. loc.	65.73	2.21	No
	ISIC-2019 challenge winner [73]			72.5 $\pm$ 1.7	-	-
	Supervised ImageNet			73.76	-	-
	No pretraining			49.27	-	-

Table 2.5: Top accuracies for the Sirotkin et al. work. The column "δ" indicates how the performance of a combination of pretext tasks differs from an individual pretext task. The left-most column shows whether a combination follows Curriculum (C), Anti-Curriculum (AC) or Mixed Curriculum (MC) ordering. [65]

to test if this approach can be used over X-Ray images from lungs, and have a good performance, comparing it with state-of-the-art models and techniques. We searched for the optimum hyper-parameters for this classification task and tested different SSL task positions to achieve better performance. We also implement a segmentation of the lung area, as the data that we used may be biased towards regions outside of the lungs. This is explained more in detail in further sections (Section 2.5). Finally, to qualitatively measure the performance of our model we implemented attention maps so we can check if the model attention is inside the lungs area.

2.5 Evaluation Frameworks

In this section, we expose different datasets of lung X-Ray images, the dataset in which we performed our experiments, and some works that have been realized over it.

As explained in Section 2.3 current medical databases are difficult to work with because they present some explicit challenges in this field. Regarding the accessibility to these databases, most of them ask to indicate what are our purposes with the database and do not allow commercial use from it. Regarding the lack of quality datasets, as we mentioned in Table 2.1 in Section 2.3, the amount of data most of the datasets have is not enough for Deep Learning solutions.

The images come all from patients that tested positive for COVID-19. The data was obtained from the SIIM-FISABIO-RSNA COVID-19 Detection dataset [74]. This is provided by the Society for Imaging Informatics in Medicine (SIIM) and consists of identifying and locating pneumonia abnormalities on chest radiographs from COVID-19 patients, and classifying them into four groups: negative for pneumonia, typical, indeterminate, or atypical. However, in this work we are not solving a detection task, so we changed the dataset labels for it to be only a classification problem. We can see a description of each label in Table 2.6. so The SIIM-FISABIO-RSNA dataset training data consists of 6,334 chest scans in DICOM format (Digital Imaging and Communication In Medicine) with 79% COVID-19 positives and 21% COVID-19 negatives and is made up of the other two datasets, the BIMCV-COVID19+ [75], and the MIDRC-RICORD [76]. All of the training dataset images were de-identified to protect patient privacy. A panel of expert radiologists labeled the images. For the creation of this dataset, the images from the BIMCV and MIDRC-RICORD were preprocessed, deleting images that were taken from the lateral of the chest, as well as the images that do not have a DICOM image associated information. The two bigger classes negative and typical account for about 75% of the total number of samples. Indeterminate and atypical samples account for the remaining 25% of samples. Only 600 of these 6,334 images are from the MIDRC-RICORD dataset.

The BIMCV dataset, where most of the images from SIIM-FISABIO-RSNA come, is one of the biggest COVID-19 related datasets, and has a multitude of works around it that try to predict the diseases in the lungs [77, 78, 79]. However, all this work detects the same flaw in the dataset, and as DeGrave et al. [80] represents by performing different experiments (Figure 2.10), it is biased toward the external lungs area. These zones may add information to the model, that could be able to learn about the age or gender of the patient, and do its prediction based on this information more

Label	Medical Meaning
Negative for pneumonia	No findings of pneumonia. However, chest radiographic findings can be absent early in the course of COVID-19 pneumonia.
Typical appearance	Findings typical of COVID-19 pneumonia are present. However, these can overlap with other infections, drug reactions, and other causes of acute lung injury.
Indeterminate appearance	Findings indeterminate for COVID-19 pneumonia and which can occur with a variety of infections and noninfectious conditions.
Atypical appearance	Findings atypical or uncommonly reported for COVID-19 pneumonia. Consider alternative diagnoses.

Table 2.6: Labels from the SIIM-FISABIO-RSNA COVID-19 dataset and their meanings

Experiment	F1 Score	Acc. (%)
Baseline: DenseNet121	0.4205 ± 0.0149	57.34 ± 2.79
MoCo-V2	0.4583 ± 0.0168	58.60 ± 1.69
L/R targeted inpainting	**0.4794 ± 0.0327**	**61.77 ± 2.28**

Table 2.7: Results from the Ridzuan et al. [7] methods

than focusing on the area of interest for these works, the lungs. As most of the SIIM-FISABIO-RSNA data come from the BIMCV dataset, this bias towards the outside of the lungs is transferred to the SIIM-FISABIO-RSNA dataset. This bias can be caused because of the information that is inside of the image, like the gender of the patient or their age, or it could be caused by the machine that was used to take the images, that may add information to the pixel values that the model may be taking advantage of.

As the dataset we are using is labeled as an object detection problem, there are not many works that focus on the classification task. However, Ridzuan et al. [7] perform a classification using also a SSL approach, along with other methods. They used a Densenet-121 [81] as a backbone model and implemented different SSL tasks, such as MoCo-V2 or the inpainting method [82]. They also implemented various methods of data augmentation, such as mirroring the lungs or replacing them with the lungs from another image. However, these augmentations tend to worsen the performance (in Figure 2.11 we can see some examples of bad augmentations). Finally, they trained their model over the CheXpert dataset [83], which has much more training data than the SIIM-FISABIO-RSNA dataset. We can see some of the results they obtained in Table 2.7.

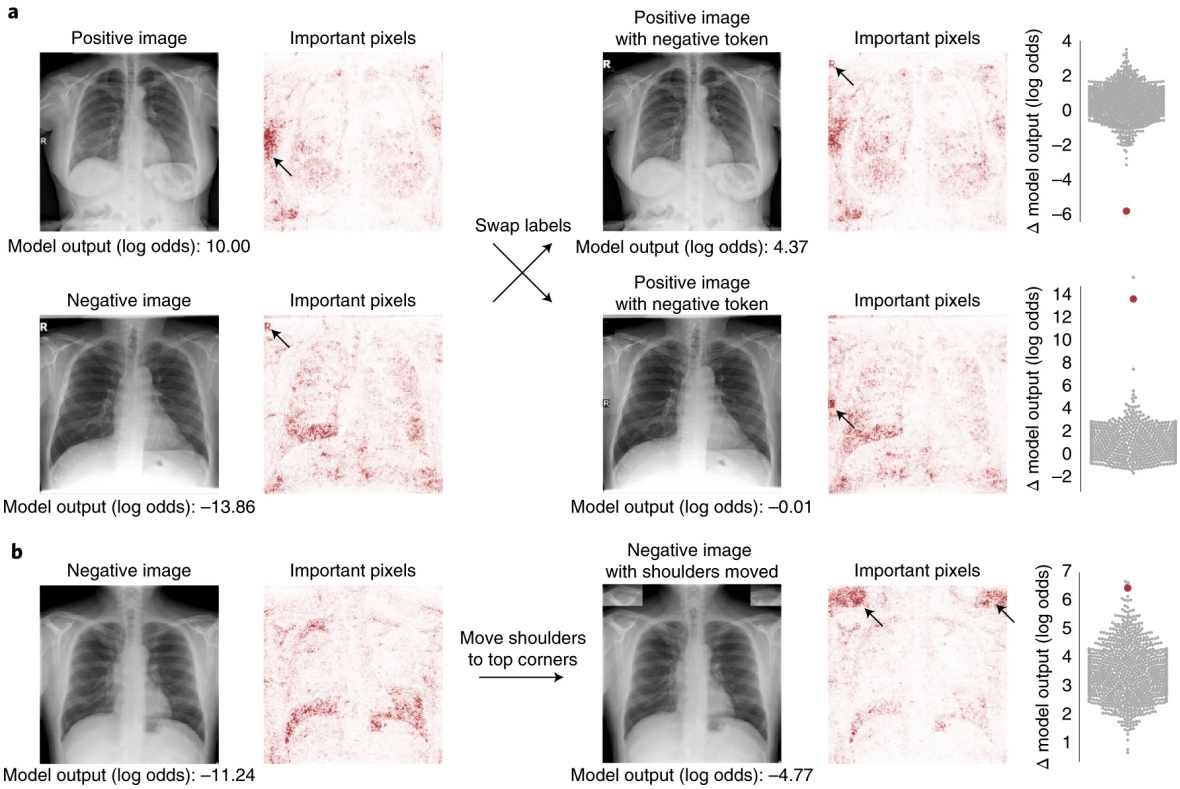


Figure 2.10: Model's attention over the chest image to perform the classification. We can see in some examples that it focuses a lot outside of the lungs. Figure obtained from the original report [80]

Figure 2.11: Failures in (a) mirrored lung augmentation and (b) lung replacement augmentation.

Chapter 3

Design and development

In this chapter, we explain the solution that we implemented along with all the decisions made for this project. This chapter is divided into three main sections. In Section 3.1 we explain the different parts our model has and how they were implemented. In Section 3.2 we explain the followed process to preprocess the data. Finally, in Section 3.3 we explain the methods that we used to evaluate and compare our models. In Figure 3.1 we have an overview of all the modules that conform the complete implementation. All the development for this project was performed in the programming language `python3` [84].

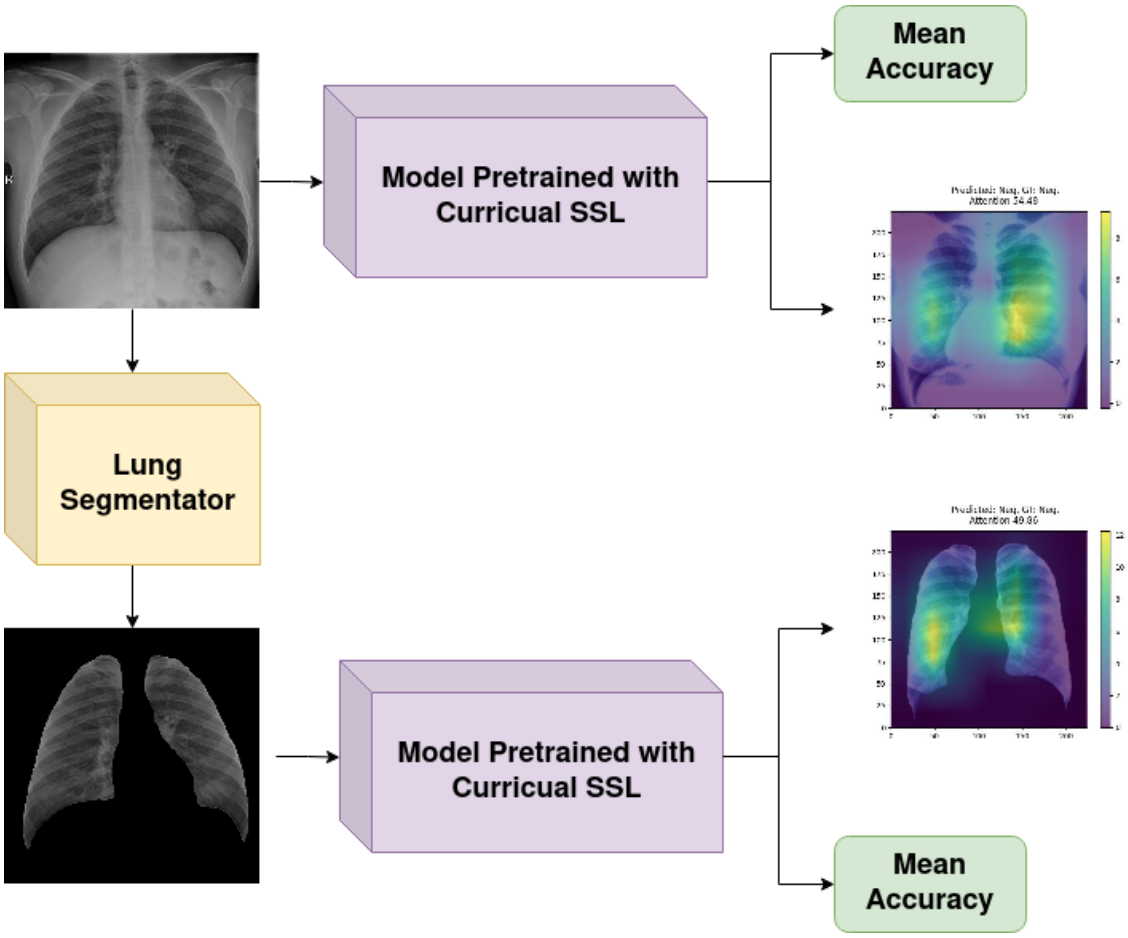


Figure 3.1: Diagram of the complete implementation conformed by a Segmentator, the model pretrained over different SSL tasks and finally the output.

3.1 Training Scheme for Curricular Self-Supervised Learning

In this work, we want to build and train a model that performs a classification using lung radiographs. As we explained in Section 2.4, we have followed the idea proposed by Kirill et al. [65] to check the feasibility of their method over a different dataset. We can see in Figure 3.2 the training scheme that they proposed. The main characteristic of this training scheme is that it keeps the backbone architecture and changes the last layers to change the SSL task the model is training to, and we can add as many SSL tasks as we want. We call these last layers heads. Each head serves a different purpose, as they correspond to the different SSL tasks. The weights of the model are kept when changing heads, so each consecutive task updates the weights from the previous ones. As explained in Section 2.4, the intuition behind this is that the information of the different tasks complements each other. After the model has been pretrained with the tasks that were configured, there is always a classification head at the end, which is used to train the model to classify the target data. For both, the pretraining and the final training we use the complete dataset.

Figure 3.2: Training scheme proposed by Kirill et al. [65] which uses different SSL task to pretrain the model to perform a final classification.

3.1.1 Model's Implementation Using Python

To perform all the steps in this work we created a model that trains over N different pretext tasks and finally performs a classification. We implemented this model using the programming language `python`.

As the implementation of the different pretext tasks is out of the scope of this work, we decided to use a framework that can be configured to perform different training with multiple pretext tasks. The chosen framework is the MMSelfSup [85] which is an open-source self-supervised learning toolbox based on PyTorch [86]. This framework is based on OpenSelfSup and is part of the OpenMMLab project. With this framework, we can design the model's backbone architecture and pretrain it over different SSL pretext tasks.

This framework works with configuration files written in `python`. In these configuration files, we can specify the different hyper-parameters that our model uses, like the batch size, Learning Rate (LR), epochs, etc. We can also define the different data augmentation that we want our model to perform over the data. Using these configuration files we automated the search of the LR for every step of the training. Previous to training the model using a complete number of epochs we train it with a reduced number with different LRs. The number of epochs that the model has to train to select

	Resnet-50	DenseNet-121
Trainable parameters	23 M	9 M
Accuracy Imagenet	77.15%	74.98%

Table 3.1: Number of parameters and accuracies over Imagenet for the two models.

the optimum LR has to be enough to decide between the different LRs. To search for this LR we train the model with the complete dataset. The number of epochs, for the search and the training, and the LR ranges that we used are detailed in Chapter 4.

3.1.2 Model's Backbone

As we explained in Section 2.3, there are different architectures for building a model and selecting one that molds to the problem is one of the most important decisions to make before tackling the problem. In this section, we explain the decisions made regarding the network as well as the different modules it is conformed by.

We consider two different architectures to act as the backbone of our model: DenseNet-121 [81] and Resnet-50 [16]. We thought of these two architectures because DenseNet-121 is used for medical image classification in plenty of works [87, 88, 89, 90], and Resnet-50 is one of the more popular architectures for all kinds of DL solutions [91, 92, 93]. As we can see in Table 3.1 DenseNet-121 has less trainable parameters than Resnet-50. However we decided to use Resnet-50 as DenseNet requires a significant amount of GPU memory, and the available GPU is limited by this. Each DenseNet layer is connected to all previous layers. Because features accumulate, the final classification layer has access to a large and diverse feature representation. As well, as these features accumulate we need to save them in memory and this makes that the amount of GPU memory needed grows exponentially [94]. As we intend to do several trainings for searching hyperparameters and training the model, having such a heavy backbone is not feasible.

3.2 Data Preparation

Many factors affect the success of Artificial Intelligence on a given task. The representation and quality of the instance data is first and foremost. This also applies to Deep Learning tasks, in which training with noisy images or having different ranges for pixel values may affect the model's learning. In this section, we explain the preprocessing we performed over our initial data to improve the learning of our model.

3.2.1 Lung Segmentation

As mentioned in Section 2.5 the dataset that we used is biased towards the outside of the lungs, making the model use the information that is not inside of the lungs region to make the final prediction. To avoid this we created a version of the data that extracts

the background from the image and uses only the lungs to train. We performed all the experiments with the original version of the data (the complete images) and with the segmented version. Using both data we are able to check how much the region outside the lungs affects the final prediction. This is important because we want to avoid all the information that may not be replicated in other experiments, i.e. we can not assure that all the capture systems take the information from outside of the lungs.

To perform this segmentation, we configure the solution proposed by Selvan et al. [5] in which they use a U-net type model [95] with an encoder-decoder architecture and a variational encoder for data imputation, as shown in Figure 3.4. To execute this we used the model's weight that they hand in their github page (https://github.com/raghavian/lungVAE/tree/master/saved_models). After preprocessing we get the mask that the model predicts, and one post-procesed which has less errors, like for example out-layers. We can see an example of this in Figure 3.3.

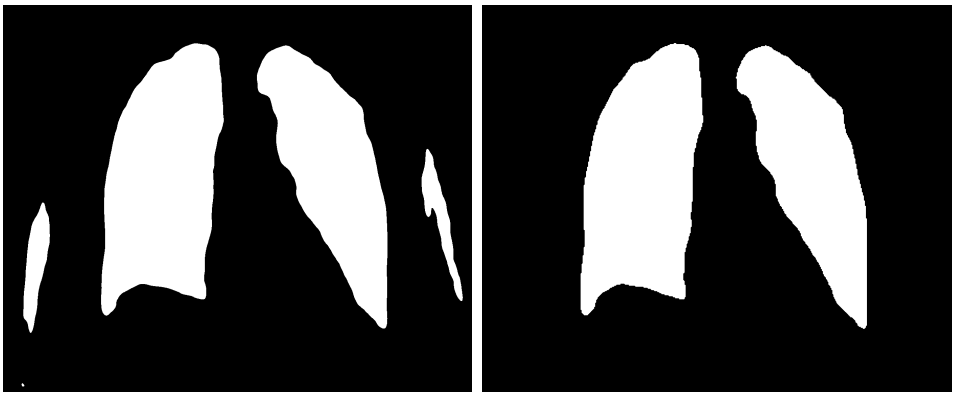


Figure 3.3: Differences between the original mask and the post-processed one.

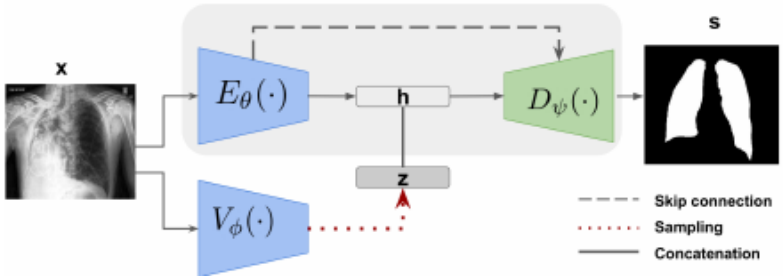


Figure 3.4: Overview of the proposed model for segmentating the lungs. Obtained from [5]

3.2.2 Format Details

As mentioned in chapter 2, the data provided in the SIIM-FISABIO-RSNA dataset is in DICOM (Digital Imaging and Communication In Medicine) format. DICOM [96, 97] is a standard used mainly for sharing medical images. Even though this format can collect plenty of information regarding medical imaging, for our model to be able to read the input images, it is necessary to transform them into a format in which the

images' pixels are represented as a number, like PNG (Portable Network Graphics) [98]. To do this we use the `pydicom` library [99], which allows us to work with the DICOM format in the python programming language. With this library, we transform the SIIM-FISABIO-RSNA database images into PNG.

After transforming the images from DICOM to PNG, we proceed to segment the images, separating the lungs region from the background. Finally, it is necessary to perform a normalization step. Normalization consists in project the pixel values from an image into a predefined range (i.e. usually [0, 1]). Not normalizing would imply that the weights from your network, are multiplied by pixel values of large magnitudes, and this would force the neuron to saturate. This is performed by subtracting the mean, μ, of each feature, x, and a division by the standard deviation, σ (Equation 3.1).

$$x := \frac{x - \mu}{\sigma} \tag{3.1}$$

3.3 Evaluation Metrics

In this work, we test the performance of the model in two ways. Quantitatively, by comparing the mean accuracy over the final classification task of the model with different pretraining. As the classes from the database are imbalanced, we need to see the mean accuracy of all the classes. Qualitatively by comparing the attention maps of the models that used the complete images, and the ones that used the segmented images.

3.3.1 Attention Maps

To check where the model is focusing its attention to make a prediction, we implemented a module that returns a class activation map (CAM) [63]. However, CAM has a drawback, as to be able to create a CAM the network architecture is restricted to have a global average pooling layer after the final convolutional layer, and then a linear (dense) layer. This makes that CAM can not be used on architectures that have multiple fully connected layers at the end of the network, such as AlexNet and VGG-19.

We can see an image of a lung with its activation map in figure 3.5. This activation map works by forwarding the image we want to check through our model. Then we get the signals from the prediction layer and map them to the last convolutional layer. Finally, we resize this last layer with the weights associated with the predicted class to match the input image and being able to see where the model's attention is.

To compare between models if they are focusing their attention on the lung region, we implemented a quantitative metric that checks the intensity of the Attention values that lay Inside the Lungs (AIL). To get the AIL we sum all the values from the extended CAM (Attention in the complete image) and the values from the extended CAM that are inside the lungs (*Attention inside the lungs*). We can see this same concept in Equation 3.4. As these values are larger where the model attention is focused if we divide the value obtained from inside the lungs by the total we get a percentage of how much attention is inside the lungs.

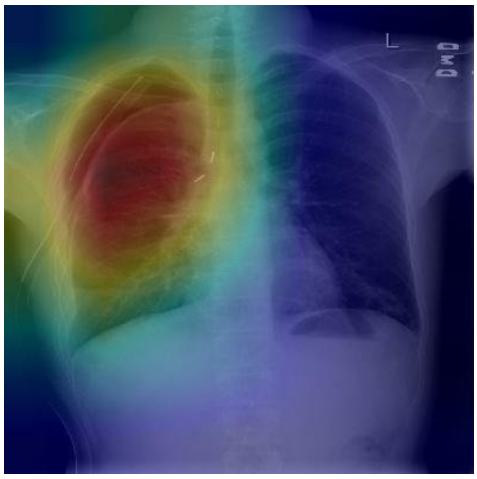


Figure 3.5: Class activation map over a lung radiography [6].

$$\text{Attention in the complete image} = \sum_i CAM_i \tag{3.2}$$

$$\text{Attention inside the lungs} = \sum_j CAM_j \tag{3.3}$$

being i all the values of the CAM, and j all the values that are inside the lung area in the CAM.

$$AIL = \frac{\text{Attention inside the lungs}}{\text{Attention in the complete image}} \tag{3.4}$$

Chapter 4

Experiments and results

This chapter is divided into four different sections. First, in Section 4.1 we complete the information that is in Section 3.2 about the changes that we did over the target dataset, specifying the values used. Then, in Section 4.2 we explain the different configurations that we did over the model's different heads for them to be able to apply the different SSL tasks to our data. In Section 4.3 we explain in detail how the training was performed, as well as the hyperparameter search. Finally, in Section 4.4, we summarize all the results and also we give quality results by displaying the model's attention maps.

4.1 Data Segmentation and Metadata Formating

For this work, we performed some transformations over the original SIIM-FISABIO-RSNA dataset, as it did not suit all of our requirements. First of all, the images were stored in sparse paths and labeled as an object detection problem, with the disease along with the bounding boxes. As we are only interested in the classification task, we iterated over the different metadata datasets (in *csv* format) to search for the information that we required and created a dataframe that summarizes all we needed. This dataframe has one column for the image name, and a second column for the disease, simplifying the recompilation of the data. We also group all the images in a single folder, as MMSelfSup requires.

As mentioned in Chapter 3, we used complete and segmented images. We decided to perform the experiments in three case scenarios: with the complete image, with the segmented image and black background, and with the segmented image and a gray image (Figure 4.1). This gray color corresponds to the mean value from the full images. We performed this last case so the mean of the complete image is not affected much because of the segmentation. Later on the results, we check if this step has any advantage over a classic segmentation with a black background. We can see the value of the mean and standard deviation of the different groups in Table 4.1. As we do not have the mean nor the standard deviation from the SIIM-FISABIO-RSNA dataset, we have to first do a preliminary data reading so we can calculate these two values.

Figure 4.1: Different type of images that are feed to the network. Complete image, segmented with a gray background and segmented with a black background (from left to right)

	Mean	**Standard Deviation**
Full image	0.5234	0.2415
Segmented Gray	0.5026	0.0859
Segmented Black	0.1028	0.2022

Table 4.1: Table with the means and standard deviation from all the datasets of images feeded to the networks

4.2 Data Augmentation and Hyperparameter Configuration

As explained in Section 3.1, we decided to use Resnet-50 over DenseNet-121 for this work. Using the configuration files that MMSelfSup provides us we can change the model's head configuration. This is particularly important when the model is about to perform a SSL task, as the data transformations performed over the data are core for the learning. In addition to these transformations, needed for the SSL task to work, we added some data augmentation to vary our data. We split these augmentations into three different groups depending on the task the head is going to solve, which can be seen in Table 4.2. We perform these augmentations because of the procedure explained in [100].

Head	**Transformations**	**SSL Tasks**
Contrastive learning	Random crop Horizontal flip Random rotation	MoCo-V2 SwAV
Non-Contrastive learning	Random crop Horizontal flip	Rotation Prediction Relative Location
Classification	Random crop Horizontal flip	Classification

Table 4.2: Different augmentations for each SSL Task

Finally, for the training process, some hyper-parameters had to be set. In this work, we set the optimizer configuration and the batch size, while we explore different values for the LR. In Table 4.3 we can see the values that we established for batch size and the optimizer for every task. The decision for the batch size is mainly influenced by the capacity of the GPU used for this work (RTX 2080). The information on this GPU can be found in Appendix A. We used SGD (Stochastic Gradient Descend) [100] for most of the tasks except with SwAV in which we used LARS (Layer-wise Adaptive Rate Scaling) [101] which was used because it is recommended optimization technique proposed in [46]. SwAV also uses a queue that is composed of feature representations from the previous batches. We decided to use a queue size 8 times larger than the batch size as it needs to be a multiple of batch size and to be less than total training data. Regarding the learning rate, we explored different values for every combination. This is explained in further detail in Section 4.3.

Task	**Batch Size**	**Epochs**	**Optimizer**	
Classification	64	170	**Type**	SGD
			Momentum	0,9
			Weight decay	1e-4
Relative Location	32	30	**Type**	SGD
			Momentum	0,9
			Weight decay	1e-4
Rotation Prediction	16	30	**Type**	SGD
			Momentum	0,9
			Weight decay	1e-4
MoCo-V2	32	30	**Type**	SGD
			Momentum	0,9
			Weight decay	1e-4
SwAV	8	30	**Type**	LARS
			Queue Size	64

Table 4.3: Different hyper-parameters that where fixed for all experiments.

4.3 Training and Learning Rate Search

In this last section, we explain the complete process to train our model, the different SSL task combinations that we tested, the learning rate searching process, and the results obtained.

4.3.1 Naive Implementation

In this first subsection, we pretrained our model in the most naive way to check if we have any errors and do our first deductions based on the obtained results. As explained in Section 3.1, the model we built is trained by combining multiple learning tasks. In Tables 4.5, 4.6 and 4.7 are the results obtained for the first training we performed. In this first training, we classified the four different classes with the model pretrained over different tasks with no combinations. For this first case, we trained it with a fixed LR. We trained and tested the model over the complete images and the segmented ones

(with black and gray background).

After training these models, we appreciate two important factors. The main one is that the model is not able to learn about the two classes with the least data, **Indeterminate appearance** and **Atypical appearance**. After searching in the current state-of-the-art [102, 7], we found that other authors also had problems with these classes, and thus making the model try to learn them can worsen the overall performance. We also performed some experiments with a smaller batch size (we can see the results from these in Table 4.4), and when the configuration is like this, the model is not able to learn nothing from this classes, nor in validation, nor in training. As the results for these data are bad both in training and in validation, w concluded that it was not a problem of the model not being able to generalize, and more a problem of the data. We decided to remove them for this work, as the main objective is to check the model's domain adaptation.

Classification over the Full images							
Pretraining	**LR**	**Batch Size**	**Neg Acc**	**Typical Acc**	**Indet Acc**	**Atipical Acc**	**Mean Acc**
From Scratch	0,01	16	74,367	67,869	0,467	0,990	35,923
Imagenet	0,01	16	79,747	82,131	6,075	0,000	**41,988**
RelativeLoc	0,01	16	81,329	84,754	0,000	0,000	41,521
RotationPred	0,01	8	81,013	84,426	0,000	0,000	41,360
Swav	0,01	4	77,840	84,590	0,000	0,000	40,608
MocoV2	0,01	16	82,278	83,607	1,402	0,000	41,822

Table 4.4: Results of the experiments with half of the batch size.

The other main inconvenience that we discovered with these results is that the segmentation with the gray background does not contribute to the learning of the model. To reduce the training load we decided to only work with the most classical approach, which is to have the segmentation with a black background.

Classification over the Full images						
Pretraining	**LR**	**Neg Acc**	**Typical Acc**	**Indet Acc**	**Atypical Acc**	**Mean Acc**
From Scratch	0,01	77,215	80,820	9,813	7,921	43,942
Imagenet	0,01	79,114	77,541	7,009	2,970	41,659
RelativeLoc	0,01	77,215	75,082	17,757	20,792	**47,712**
RotationPred	0,01	78,81	81,803	5,140	8,911	43,666
Swav	0,01	74,051	79,508	10,280	11,881	43,930
MocoV2	0,01	80,063	74,918	14,016	11,881	45,220

Table 4.5: Classification accuracy for the different classes over complete images.

Classification over the Segmented images with black background						
Pretraining	**LR**	**Neg Acc**	**Typical Acc**	**Indet Acc**	**Atypical Acc**	**Mean Acc**
From Scratch	0,01	77,848	80,000	9,813	4,950	43,153
Imagenet	0,01	76,582	80,656	7,944	7,944	43,282
RelativeLoc	0,01	76,899	80	11,215	4,95	43,266
RotationPred	0,01	74,051	76,230	10,280	13,861	43,606
Swav	0,01	79,430	81,311	12,617	6,931	**45,072**
MocoV2	0,01	77,848	78,525	8,411	11,881	44,166

Table 4.6: Classification accuracy for the different classes over segmented images with black background.

Classification over the Segmented images with gray background						
Pretraining	**LR**	**Neg Acc**	**Typical Acc**	**Indet Acc**	**Atypical Acc**	**Mean Acc**
From Scratch	0,01	70,57	80,656	10,748	1,980	40,989
Imagenet	0,01	74,051	82,951	4,206	0,990	40,550
RelativeLoc	0,01	77,532	82,131	4,673	3,96	42,074
RotationPred	0,01	76,899	77,049	12,150	12,871	**44,742**
Swav	0,01	77,215	78,852	8,879	8,911	43,464
MocoV2	0,01	76,899	77,049	11,682	8,911	43,635

Table 4.7: Classification accuracy for the different classes over segmented images with gray background.

4.3.2 Pretraining Over One SSL Task

After removing the two last classes and the images with a gray background, we remade our experiments. This time we perform an extensive search of the learning rate. We need to do this search for the LR of the pretext task as well as for the classification. In Tables 4.8 and 4.9 we can see the best LR for each training scheme along with the accuracy obtained after training it for 150 epochs. We can see in Appendix B more detailed, the results for the SSL task search in Tables B.1 and B.3, and the results for the classification search in Tables B.2 and B.4. The pretext tasks were trained over 20 epochs with every learning rate. Then the better learning rate is chosen and we train the model for another 10 epochs (30 epochs in total). If we have N SSL tasks we repeat this process for each of the N starting with the weights learned from the previous one. With this pretrained model then we change heads to perform classification and train with every learning rate for 70 epochs. Again, we chose the better learning rate and train the model for another 80 epochs (150 epochs in total). In Figure 4.2 we can see a validation plot in which we can appreciate that with around 75 epochs of the total epochs we can have an intuition of which is the best LR.

Figure 4.2: Validation accuracy for the different SSL tasks.

We search for the learning rate in the range of [0,01 - 0,25] with steps of 0,058 (6 different learning rates). As we can see in these tables, in this dataset, training the model from one that was pretrained over a SSL task with data in the same domain as the one used for the final classification, tend to improve the performance compared to training over from scratch or over Imagenet. When trained from scratch, the model is not able to learn good enough visual representations, while training over Imagenet, in which the domain differs a lot, does not give the model enough information to outperform the SSL task pretraining.

Full Image - Classification		
Pretraining	**Best LR**	**Acc 150 epoch**
Scrach	0,039	83,69
Imagenet	0,155	82,75
Moco	0,039	83,89
Swav	0,039	83,97
Rel-Loc	0,039	83,62
Rotation	0,068	**84,72**

Table 4.8: Summarized results for classification accuracy with complete images.

Segmented Black - Classification		
Pretraining	**Best LR**	**Acc 150 epoch**
Scrach	0,058	81,78
Imagenet	0,01	82,73
Moco	0,058	82,639
Swav	0,01	83,716
Rel-Loc	0,01	**83,798**
Rotation	0,106	83,428

Table 4.9: Summarized results for classification accuracy with segmented images.

4.3.3 Pretraining Over Two SSL Tasks

After training our model with one SSL task we start combining two of them. As explained in Section 3.1, we pretrained the model by training it in one SSL task, and then changing heads to training it over another SSL task. In Tables 4.10 and 4.11 we can see the best LR for each training scheme along with the accuracy obtained after training it for 150 epochs. We can see in Appendix B more detailed, the results of this LR search in Tables B.5 and B.7 for the SSL task and in Tables B.6 and B.8 along with the result of training the model with the best LR for 150 epochs. As we can see in these tables, the result of classification after pretraining the model with combined SSL tasks can outperform the pretraining with only one task, and over Imagenet.

Over the results in Table B.6, we can also see that when the model is last pretrained using MoCo-V2 the results obtained have the worst performance, while if we train first over MoCo-V2 they are in line with the other pretraining, achieving the best value in the MoCo-V2 → Relative location combination (85,59 %). We consider this for, in Section 4.3.4, reducing the number of experiments by starting always the pretraining using MoCo-V2 as the first SSL task. We need to reduce the number of combinations because of the computational cost of trying all the possible combinations of SSL tasks with all the LRs.

Full Image Combined II - Classification		
Pretraining	**Best LR**	**Acc 150 epoch**
MoCo + Rotation	0,058	84,7728
MoCo + Rel-Loc	0,01	**85,59**
MoCo + SwAV	0,01	83,67
Rotation + MoCo	0,058	76,087
Rotation + Rel-Loc	0,01	84,338
Rotation + SwAV	0,01	**84,81**
Rel-Loc + Rotation	0,01	84,5476
Rel-Loc + MoCo	0,01	82,7957
Rel-Loc + SwAV	0,058	**85,27**
SwAV + Rotation	0,01	83,89
SwAV + Rel-Loc	0,01	**84,92**
SwAV + MoCo	0,01	82,374

Table 4.10: Summarized results for classification accuracy with complete images and 2 SSL tasks combined.

Segmented Black Combined II - Classification		
Pretraining	**Best LR**	**Acc 150 epoch**
MoCo + Rotation	0,01	**84,5554**
MoCo + Rel-Loc	0,01	84,296
MoCo + SwAV	0,106	81,3488
Rotation + MoCo	0,058	77,3361
Rotation + Rel-Loc	0,01	83,3238
Rotation + SwAV	0,01	**84,4314**
Rel-Loc + Rotation	0,01	**83,7985**
Rel-Loc + MoCo	0,01	81,148
Rel-Loc + SwAV	0,01	83,6823
SwAV + Rotation	0,01	81,2975
SwAV + Rel-Loc	0,01	82,7386
SwAV + MoCo	0,01	**84,3495**

Table 4.11: Summarized results for classification accuracy with segmented images and 2 SSL tasks combined.

4.3.4 Pretraining Over Three SSL Task

As explained in Section 4.3.3, to reduce the number of experiments, we decide to use MoCo-V2 as the base SSL task to pretrain our model with three SSL tasks. The experiments are produced in the same order as in the previous sections, and the results can be seen in Tables 4.10 and 4.11. We can see in Appendix B the detailed results in Tables B.9, B.11, B.10 and B.12. We can see in the results that some combinations outperform the pretraining with two different SSL tasks.

Full Image Combined III - Classification		
Pretraining	**Best LR**	**Acc 150 epoch**
MoCo + Rotation + Rel-Loc	0,058	**85,28**
MoCo + Rotation + SwAV	0,058	84,8
MoCo + Rel-Loc + Rotation	0,01	84,19
MoCo + Rel-Loc + SwAV	0,01	**85,49**
MoCo + SwAV + Rotation	0,01	**85,67**
MoCo + SwAV + Rel-Loc	0,01	83,74

Table 4.12: Summarized results for classification accuracy with complete images and 3 SSL tasks combined.

Segmented Black Combined III - Classification		
Pretraining	**Best LR**	**Acc 150 epoch**
MoCo + Rotation + Rel-Loc	0,01	84,38
MoCo + Rotation + SwAV	0,01	**85,14**
MoCo + Rel-Loc + Rotation	0,01	83,48
MoCo + Rel-Loc + SwAV	0,01	**84,64**
MoCo + SwAV + Rotation	0,01	84,67
MoCo + SwAV + Rel-Loc	0,01	**84,71**

Table 4.13: Summarized results for classification accuracy with segmented images and 3 SSL tasks combined.

4.4 Performance Overview and Qualitative Results

In this section, we summarize the results from the previous section along with the visualization of the different models' attention. This section serves as a way to have a global vision of the system and how the results have been improving by adding tasks to the pretraining.

In Table 4.14 we can see all the results obtained for one, two, and three SSL tasks combined, for both full images, and segmented ones. We marked the better accuracies for a single task, two tasks combined and three tasks combined pretraining. We can also see in this table that most of the models have a better performance when trained and tested over complete images, supporting that the model is biased towards the information outside of the lung area.

	Full Image	Segmented	Full/Segmented
Scrach	83,690	81,780	Full
Imagenet	82,750	82,730	Full
MoCo	83,890	82,640	Full
Rot-Pred	83,970	83,428	Full
Rel-Loc	83,620	**83,799**	Segmented
SwAV	**84,720**	83,716	Full
MoCo + Rot-Pred	84,773	**84,555**	Full
MoCo + Rel-Loc	**85,590**	84,296	Full
MoCo + SwAV	83,670	81,349	Full
Rot-Pred + MoCo	76,087	77,336	Segmented
Rot-Pred + Rel-Loc	84,338	83,324	Full
Rot-Pred + SwAV	84,810	84,431	Full
Rel-Loc + Rot-Pred	84,548	83,799	Full
Rel-Loc + MoCo	82,796	81,148	Full
Rel-Loc + SwAV	85,270	83,682	Full
SwAV + Rot-Pred	83,890	81,298	Full
SwAV + Rel-Loc	84,920	82,739	Full
SwAV + MoCo	82,374	84,350	Segmented
MoCo + Rot-Pred + Rel-Loc	85,280	84,380	Full
MoCo + Rot-Pred + SwAV	84,800	**85,140**	Segmented
MoCo + Rel-Loc + Rot-Pred	84,190	83,480	Full
MoCo + Rel-Loc + SwAV	85,490	84,640	Full
MoCo + SwAV + Rel-Loc	**85,670**	84,670	Full
MoCo + SwAV + Rot-Pred	83,740	84,710	Segmented

Table 4.14: Accuracy obtained with every combination of SSL tasks. The left-most column indicates the configuration, in which the training order is from left to right, i.e, the last row indicates a model that has been pretrained first over MoCoV2, then SwAV, and finally over Rotation prediction. The right-most column indicates if that configuration performs better over complete or segmented images.

4.4.1 Model's Attention

In this subsection, we display different images that correspond to the model's class attention maps. This CAMs can be found in Figure 4.4 for the complete and segmented images. In Appendix C we can find the plots for every training scheme (Figures C.2, C.3, C.2, C.4, C.5, C.6, C.7, C.8). In the title of each of these images, we indicated the AIL metric (explained in Subsection 3.3.1) along with the predicted label and the ground truth (in Figure C.1 in Appendix C, we can see one of these images in a bigger shape to be able to read the text). Finally, in Table 4.15 and in Figure 4.3 we can see the mean AIL for each model configuration. We can see that mostly when the model has been trained over segmented images, the AIL value increases, meaning that the model focuses more on the inside area of the lungs.

Model	**Full**	**Segmented**
Scratch	37.3083	46.3289
Imagenet	38.158	43.4132
MoCo	37.4007	44.1397
Rot-Pred	42.8201	**52.9003**
Rel-Loc	36.3305	41.7388
SwAV	**45.9535**	47.871
MoCo + Rot-Pred	47.9275	42.6116
MoCo + Rel-Loc	39.5732	45.4199
MoCo + SwAV	41.7999	**51.1816**
Rot-Pred + MoCo	31.8748	48.3771
Rot-Pred + Rel-Loc	44.4802	43.1016
Rot-Pred + SwAV	41.4614	49.0282
Rel-Loc + Rot-Pred	**48.6326**	46.4846
Rel-Loc + MoCo	39.4196	48.0898
Rel-Loc + SwAV	46.2610	49.7293
SwAV + Rot-Pred	47.1613	45.6833
SwAV + Rel-Loc	43.0599	43.0684
SwAV + MoCo	36.5135	46.1692
MoCo + Rot-Pred + SwAV	30.6949	42.1145
MoCo + Rot-Pred + Rel-Loc	38.8207	39.6664
MoCo + Rel-Loc + Rot-Pred	38.5112	48.6078
MoCo + Rel-Loc + SwAV	**46.3053**	43.1919
MoCo + SwAV + Rel-Loc	40.8918	**53.8027**
MoCo + SwAV + Rot-Pred	40.1955	44.2062
Mean	40.8982	**46.1219**

Table 4.15: AIL of each model and comparing training with complete or segmented images.

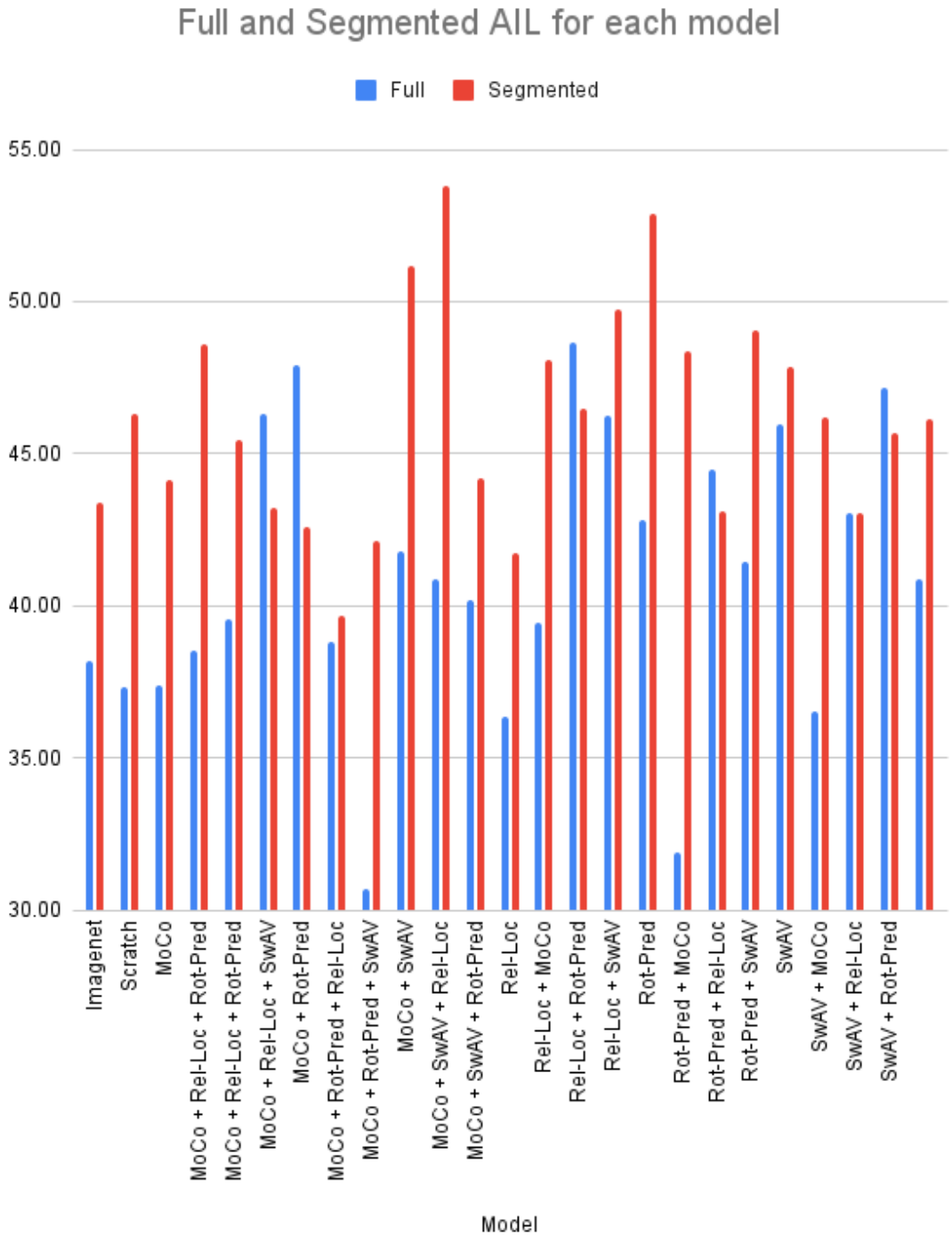


Figure 4.3: Plot representing the AIL of each model and comparing training with complete or segmented images.

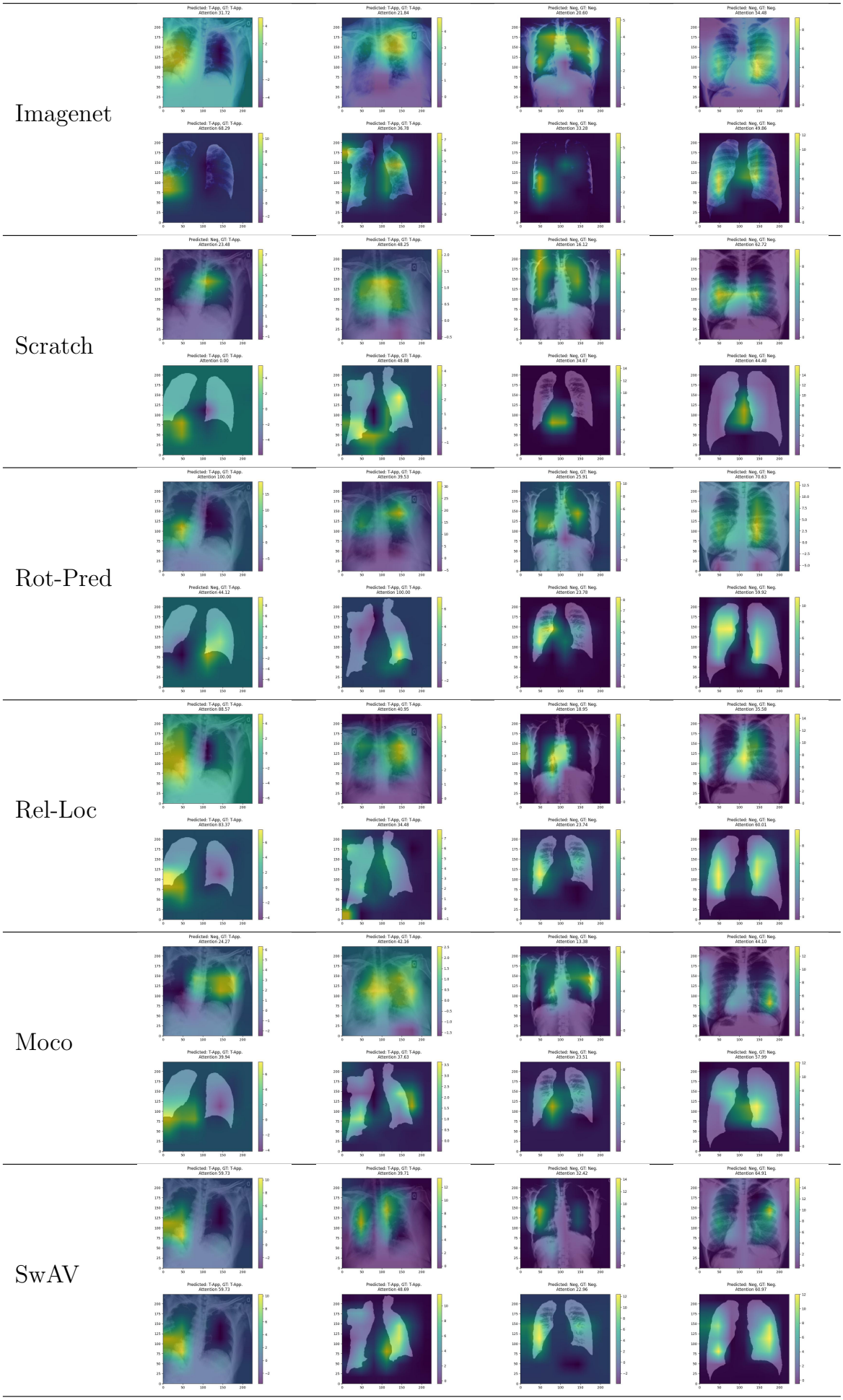


Figure 4.4: Class attention maps for the single task pretraining over complete images.

Chapter 5

Conclusions and Future Work

In this chapter, we finish this work by, in Section 5.1, summarizing all the work that has been done for this thesis to work. In Section 5.2, we expose the conclusions that can be obtained from the results of Chapter 4. In Section 5.1, we summarize all the work that has been done for this thesis to work. Finally, in Section 5.3 we propose some future work that can be done using this work as a base or changing some of the design decisions that were explained in Chapter 3.

5.1 Work Done

To the aim of this project, we implemented a SSL pretraining scheme that combines multiple pretext tasks in a sequential order to create a more complex and accurate visual representation of the data. To prove that combining SSL tasks gives the model more information, we did all the experiments with combinations of up to three SSL tasks. We tested this pretrained model over X-Ray lung images from the SIIM-FISABIO-RSNA dataset that we collected and preprocessed. To check the amount of information that this dataset has towards the region outside the lungs, we performed a semantic segmentation and trained our model with complete and segmented images. Finally, we visualized the attention maps of the different models and created a custom metric (AIL), so we could assure that the models' attention focuses more on the lungs region when trained with the segmented images.

5.2 Conclusions

In this thesis, we studied the problems that come along with using deep learning solutions with medical images. Databases from the medical field tend to be short in labeled data for training a DL model. However, to avoid this problem we implemented a model that is trained using a combination of SSL tasks to surpass this lack of labels.

The main conclusion that we obtained by doing this work, is that combining SSL tasks in the training scheme can improve a pretraining over Imagenet or from a single SSL task. Pretraining in Imagenet we obtained an 83.69% mean accuracy when trained over complete images, while, in our best configuration training with this same type of images, that is MoCo-V2 $\rightarrow$ SwAV $\rightarrow$ Relative Location we obtained an 85.67%, thus

achieving a **+1.98** accuracy. With the segmented images we achieve a similar performance increase, having an 83.799% when pretrained over Imagenet, and an 85.14% when using our best configuration for this case which is MoCo-V2 → Rotation Prediction → SwAV, achieving a **+1.341** accuracy.

We could also see that when trained over segmented images, the accuracy performance is worst than with complete images because the amount of information is less. However, the difference is not big (-0.53% accuracy difference) and on the other hand, we could see that it focuses more on the lung region. Knowing this, using the segmented model could be beneficial for inference over images that we do not the origin of, and may be biased. Another conclusion that we can extract from the attention results is that the models that focus more attention on the lung region and that have been trained with complete images may be the more resilient to the changes in the inference images, as they are not as affected by the bias as the others.

Finally, during the first experiments, we saw that the learning process over this dataset is highly influenced by the batch size. When the batch size was too small the model was unable to learn anything from the images labeled as atypical or indeterminate appearance.

5.3 Future Work

In this section, we detail some ideas that we were not able to perform in this thesis, but that may be interesting for future works that take this as a base or that want to modify it.

Firstly, using a bigger database to do the pretraining, either the one that then will be used for classification, as we did, or one in the same domain so the model can generalize, as in Ridzuan et al. [7] work. Also, using different or more combinations of SSL tasks. In this work, we have explored some of the options but there is still a large number of SSL tasks to test [103, 104, 105]. Then, perform more and more complex data augmentation over the dataset, to be able to expand the capacity of the model. Another idea would be to use a more complex backbone, like Densenet as explained in Section 3.1. Then, as a method to check how much the information from outside of the lungs region contributes to the final prediction, we propose to implement an inverted segmentation, in which the lungs are removed from the image and then do the classification.

Regarding the attention of the model, we propose to implement the CKA (Centered Kernel Alignment) [106] metric in future works. This measures the similarity of representations for different network layers. With this, we could measure the similarity of different pretrainings, and if the performance is related to the representation. This could be useful to propose a better sorting of the SSL tasks in the training scheme.

Finally, as the database that we used is aimed to perform detection, we could check if the model's attention is focused on the regions where the diseases are labeled. Another work would be to implement a metric similar to the AIL that we created, but

taking into consideration these areas from the detection dataset. Finally, we would also propose to use this training scheme for different final tasks, like object detection, and check its performance.

Bibliography

[1] A. Anwar, "Difference between AlexNet, VGGNet, ResNet and Inception," Jan. 2022. ix, 6

[2] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark Analysis of Representative Deep Neural Network Architectures," *IEEE Access*, vol. 6, pp. 64270–64277, 2018. Conference Name: IEEE Access. ix, 7

[3] A. Das, H. Agrawal, C. L. Zitnick, D. Parikh, and D. Batra, "Human Attention in Visual Question Answering: Do Humans and Deep Networks Look at the Same Regions?," *arXiv:1606.03556 [cs]*, June 2016. arXiv: 1606.03556. ix, 10

[4] C. Doersch and A. Zisserman, "Multi-task Self-Supervised Visual Learning," Tech. Rep. arXiv:1708.07860, arXiv, Aug. 2017. arXiv:1708.07860 [cs] type: article. ix, 12, 13, 14

[5] R. Selvan, E. B. Dam, N. S. Detlefsen, S. Rischel, K. Sheng, M. Nielsen, and A. Pai, "Lung Segmentation from Chest X-rays using Variational Data Imputation," *arXiv:2005.10052 [cs, eess, stat]*, July 2020. arXiv: 2005.10052. ix, 22

[6] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, M. Lungren, and A. Ng, "CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning," Nov. 2017. ix, 24

[7] M. Ridzuan, A. A. Bawazir, I. G. Navarette, I. Almakky, and M. Yaqub, "Challenges in COVID-19 Chest X-Ray Classification: Problematic Data or Ineffective Approaches?," Jan. 2022. Number: arXiv:2201.06052 arXiv:2201.06052 [cs, eess]. xi, 16, 28, 40

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, May 2017. 3, 5, 13

[9] J. Raj, A. Hoque, and D. A. Saha, "Various Methodologies for Micro-Video Recommendation System: A Survey," SSRN Scholarly Paper 3598885, Social Science Research Network, Rochester, NY, May 2020. 5

[10] A. F. Anta, L. N. Chiroque, P. Morere, and A. Santos, "Sentiment Analysis and Topic Detection of Spanish Tweets: A Comparative Study of NLP Techniques," p. 8. 5

[11] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A Survey on Contrastive Self-Supervised Learning," *Technologies*, vol. 9, p. 2, Mar. 2021. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute. 5, 9, 12

[12] D. Ravì, C. Wong, F. Deligianni, M. Berthelot, J. Andreu-Perez, B. Lo, and G.-Z. Yang, "Deep Learning for Health Informatics," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, pp. 4–21, Jan. 2017. Conference Name: IEEE Journal of Biomedical and Health Informatics. 5

[13] P. P. Shinde and S. Shah, "A Review of Machine Learning and Deep Learning Applications," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–6, Aug. 2018. 5

[14] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," *Briefings in Bioinformatics*, vol. 18, pp. 851–869, Sept. 2017. 5

[15] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs]*, Apr. 2015. arXiv: 1409.1556. 5

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," pp. 770–778, 2016. 5, 21

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *arXiv:1409.4842 [cs]*, Sept. 2014. arXiv: 1409.4842. 5

[18] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *Multiple Classifier Systems*, Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 1–15, Springer, 2000. 5

[19] O. Cordón, P. Kazienko, and B. Trawiński, "Special Issue on Hybrid and Ensemble Methods in Machine Learning," *New Generation Computing*, vol. 29, pp. 241–244, July 2011. 5

[20] S. Dodge and L. Karam, "A Study and Comparison of Human and Deep Learning Recognition Performance under Visual Distortions," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–7, July 2017. 5

[21] N. Kühl, M. Goutier, L. Baier, C. Wolff, and D. Martin, "Human vs. supervised machine learning: Who learns patterns faster?," Tech. Rep. arXiv:2012.03661, arXiv, Nov. 2020. arXiv:2012.03661 [cs] type: article. 5

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, Dec. 2015. 5

[23] W. N. Price and I. G. Cohen, "Privacy in the age of medical big data," *Nature Medicine*, vol. 25, pp. 37–43, Jan. 2019. Number: 1 Publisher: Nature Publishing Group. 8

[24] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, R. Kim, R. Raman, P. C. Nelson, J. L. Mega, and D. R. Webster, "Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs," *JAMA*, vol. 316, pp. 2402–2410, Dec. 2016. 8

[25] "Deep Learning at Chest Radiography: Automated Classification of Pulmonary Tuberculosis by Using Convolutional Neural Networks | Radiology." 8

[26] J. Lao, Y. Chen, Z.-C. Li, Q. Li, J. Zhang, J. Liu, and G. Zhai, "A Deep Learning-Based Radiomics Model for Prediction of Survival in Glioblastoma Multiforme," *Scientific Reports*, vol. 7, p. 10353, Sept. 2017. Number: 1 Publisher: Nature Publishing Group. 8

[27] Z. Li, Y. Wang, J. Yu, Y. Guo, and W. Cao, "Deep Learning based Radiomics (DLR) and its usage in noninvasive IDH1 prediction for low grade glioma," *Scientific Reports*, vol. 7, p. 5467, July 2017. Number: 1 Publisher: Nature Publishing Group. 8

[28] P. Lambin, E. Rios-Velazquez, R. Leijenaar, S. Carvalho, R. G. van Stiphout, P. Granton, C. M. Zegers, R. Gillies, R. Boellard, A. Dekker, and H. J. Aerts, "Radiomics: Extracting more information from medical images using advanced feature analysis," *European journal of cancer (Oxford, England : 1990)*, vol. 48, pp. 441–446, Mar. 2012. 8

[29] S. M. Abbas and D. S. N. Singh, "Region-based Object Detection and Classification using Faster R-CNN," in *2018 4th International Conference on Computational Intelligence & Communication Technology (CICT)*, pp. 1–6, Feb. 2018. 8

[30] R. Li and J. Yang, "Improved YOLOv2 Object Detection Model," in *2018 6th International Conference on Multimedia Computing and Systems (ICMCS)*, pp. 1–6, May 2018. ISSN: 2472-7652. 8

[31] D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013* (K. Mori, I. Sakuma, Y. Sato, C. Barillot, and N. Navab, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 411–418, Springer, 2013. 9

[32] H. Su, F. Xing, X. Kong, Y. Xie, S. Zhang, and L. Yang, "Robust Cell Detection and Segmentation in Histopathological Images Using Sparse Reconstruction and Stacked Denoising Autoencoders," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), Lecture Notes in Computer Science, (Cham), pp. 383–390, Springer International Publishing, 2015. 9

[33] J. Xu, L. Xiang, Q. Liu, H. Gilmore, J. Wu, J. Tang, and A. Madabhushi, "Stacked Sparse Autoencoder (SSAE) for Nuclei Detection on Breast Cancer Histopathology Images," *IEEE Transactions on Medical Imaging*, vol. 35, pp. 119–130, Jan. 2016. Conference Name: IEEE Transactions on Medical Imaging. 9

[34] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative Adversarial Networks: An Overview," *IEEE Signal Processing Magazine*, vol. 35, pp. 53–65, Jan. 2018. Conference Name: IEEE Signal Processing Magazine. 9

[35] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification," *Neurocomputing*, vol. 321, pp. 321–331, Dec. 2018. 9

[36] M. J. M. Chuquicusma, S. Hussein, J. Burt, and U. Bagci, "How to fool radiologists with generative adversarial networks? A visual turing test for lung cancer diagnosis," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pp. 240–244, Apr. 2018. ISSN: 1945-8452. 9

[37] J. G. A. Barbedo, "Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification," *Computers and Electronics in Agriculture*, vol. 153, pp. 46–53, Oct. 2018. 9

[38] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A Comprehensive Survey on Transfer Learning," *arXiv:1911.02685 [cs, stat]*, June 2020. arXiv: 1911.02685. 9

[39] R. Sousa, "Transfer Learning: Current Status, Trends and Challenges," p. 2. 9

[40] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness," *arXiv:1811.12231 [cs, q-bio, stat]*, Jan. 2019. arXiv: 1811.12231. 9

[41] A. López-Cifuentes, M. Escudero-Viñolo, J. Bescós, and García-Martín, "Semantic-Aware Scene Recognition," *Pattern Recognition*, vol. 102, p. 107256, June 2020. arXiv: 1909.02410. 9

[42] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4L: Self-Supervised Semi-Supervised Learning," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, (Seoul, Korea (South)), pp. 1476–1485, IEEE, Oct. 2019. 9, 12

[43] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," Tech. Rep. arXiv:2002.05709, arXiv, June 2020. arXiv:2002.05709 [cs, stat] type: article. 9

[44] X. Chen, H. Fan, R. Girshick, and K. He, "Improved Baselines with Momentum Contrastive Learning," Mar. 2020. Number: arXiv:2003.04297 arXiv:2003.04297 [cs]. 10

[45] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum Contrast for Unsupervised Visual Representation Learning," pp. 9729–9738, 2020. 10

[46] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments,"

in *Advances in Neural Information Processing Systems*, vol. 33, pp. 9912–9924, Curran Associates, Inc., 2020. 10, 27

[47] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised Visual Representation Learning by Context Prediction," *arXiv:1505.05192 [cs]*, Jan. 2016. arXiv: 1505.05192. 10

[48] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised Representation Learning by Predicting Image Rotations," *arXiv:1803.07728 [cs]*, Mar. 2018. arXiv: 1803.07728. 10

[49] J. L. Elman, "Learning and development in neural networks: the importance of starting small," *Cognition*, vol. 48, pp. 71–99, July 1993. 12

[50] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, (New York, NY, USA), pp. 41–48, Association for Computing Machinery, June 2009. 12

[51] Y. Chen, Y. Liu, Y. Cheng, and V. O. K. Li, "A Teacher-Student Framework for Zero-Resource Neural Machine Translation," *arXiv:1705.00753 [cs]*, May 2017. arXiv: 1705.00753. 12

[52] M. Kumar, B. Packer, and D. Koller, "Self-Paced Learning for Latent Variable Models," in *Advances in Neural Information Processing Systems*, vol. 23, Curran Associates, Inc., 2010. 12

[53] L. Jiang, D. Meng, S.-I. Yu, Z. Lan, S. Shan, and A. Hauptmann, "Self-Paced Learning with Diversity," p. 9. 12

[54] C. L. Srinidhi and A. L. Martel, "Improving Self-Supervised Learning With Hardness-Aware Dynamic Curriculum Learning: An Application to Digital Pathology," pp. 562–571, 2021. 12

[55] A. Murali, L. Pinto, D. Gandhi, and A. Gupta, "CASSL: Curriculum Accelerated Self-Supervised Learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6453–6460, May 2018. ISSN: 2577-087X. 12

[56] O. Wiles, A. S. Koepke, and A. Zisserman, "Self-supervised learning of a facial attribute embedding from video," Tech. Rep. arXiv:1808.06882, arXiv, Aug. 2018. arXiv:1808.06882 [cs] type: article. 12

[57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. 12

[58] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," Tech. Rep. arXiv:1409.0473, arXiv, May 2016. arXiv:1409.0473 [cs, stat] type: article. 12

[59] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The Long-Document Transformer," Tech. Rep. arXiv:2004.05150, arXiv, Dec. 2020. arXiv:2004.05150 [cs] type: article. 12

[60] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," Tech. Rep. arXiv:1312.6034, arXiv, Apr. 2014. arXiv:1312.6034 [cs] type: article. 12

[61] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), Lecture Notes in Computer Science, (Cham), pp. 818–833, Springer International Publishing, 2014. 12

[62] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation," *PLOS ONE*, vol. 10, p. e0130140, July 2015. Publisher: Public Library of Science. 12

[63] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning Deep Features for Discriminative Localization," pp. 2921–2929, 2016. 12, 23

[64] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019. 12

[65] K. Sirotkin, M. E. Viñolo, P. Carballeira, and J. C. SanMiguel, "Improved skin lesion recognition by a Self-Supervised Curricular Deep Learning approach," *arXiv:2112.12086 [cs]*, Dec. 2021. arXiv: 2112.12086. 12, 13, 14, 20

[66] S. Liu, E. Johns, and A. Davison, "End-To-End Multi-Task Learning With Attention," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 12

[67] J. Zhu, Y. Li, and S. K. Zhou, "Aggregative Self-Supervised Feature Learning," *ArXiv*, 2020. 12

[68] Y. Wang, J. Zhang, M. Kan, S. Shan, and X. Chen, "Self-Supervised Equivariant Attention Mechanism for Weakly Supervised Semantic Segmentation," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Seattle, WA, USA), pp. 12272–12281, IEEE, June 2020. 12

[69] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010. 13

[70] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *Computer Vision – ECCV 2012* (D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), vol. 7576, pp. 746–760, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. Series Title: Lecture Notes in Computer Science. 13

[71] P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Scientific Data*, vol. 5, p. 180161, Dec. 2018. 13

[72] N. C. F. Codella, D. Gutman, M. E. Celebi, B. Helba, M. A. Marchetti, S. W. Dusza, A. Kalloo, K. Liopyris, N. Mishra, H. Kittler, and A. Halpern, "Skin lesion analysis toward melanoma detection: A challenge at the 2017 International symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC)," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pp. 168–172, Apr. 2018. ISSN: 1945-8452. 13

[73] N. Gessert, M. Nielsen, M. Shaikh, R. Werner, and A. Schlaefer, "Skin lesion classification using ensembles of multi-resolution EfficientNets with meta data," *MethodsX*, vol. 7, p. 100864, 2020. 14

[74] P. Lakhani, J. Mongan, C. Singhal, Q. Zhou, K. P. Andriole, W. F. Auffermann, P. Prasanna, T. Pham, M. Peterson, P. J. Bergquist, T. S. Cook, S. F. Ferraciolli, G. C. d. A. Corradi, M. Takahashi, S. S. Workman, M. Parekh, S. Kamel, J. H. Galant, A. Mas-Sanchez, E. C. Benítez, M. Sánchez-Valverde, L. Jaques, M. Panadero, M. Vidal, M. Culiáñez-Casas, D. M. Angulo-Gonzalez, S. G. Langer, M. d. l. I. Vaya, and G. Shih, "The 2021 SIIM-FISABIO-RSNA Machine Learning COVID-19 Challenge: Annotation and Standard Exam Classification of COVID-19 Chest Radiographs.," tech. rep., OSF Preprints, Oct. 2021. type: article. 15

[75] M. d. l. I. Vayá, J. M. Saborit, J. A. Montell, A. Pertusa, A. Bustos, M. Cazorla, J. Galant, X. Barber, D. Orozco-Beltrán, F. García-García, M. Caparrós, G. González, and J. M. Salinas, "BIMCV COVID-19+: a large annotated dataset of RX and CT images from COVID-19 patients," *arXiv:2006.01174 [cs, eess]*, June 2020. arXiv: 2006.01174. 15

[76] E. B. Tsai, S. Simpson, M. P. Lungren, M. Hershman, L. Roshkovan, E. Colak, B. J. Erickson, G. Shih, A. Stein, J. Kalpathy-Cramer, J. Shen, M. Hafez, S. John, P. Rajiah, B. P. Pogatchnik, J. Mongan, E. Altinmakas, E. R. Ranschaert, F. C. Kitamura, L. Topff, L. Moy, J. P. Kanne, and C. C. Wu, "The RSNA International COVID-19 Open Radiology Database (RICORD)," *Radiology*, vol. 299, pp. E204–E213, Apr. 2021. Publisher: Radiological Society of North America. 15

[77] A. Afifi, N. E. Hafsa, M. A. S. Ali, A. Alhumam, and S. Alsalman, "An Ensemble of Global and Local-Attention Based Convolutional Neural Networks for COVID-19 Diagnosis on Chest X-ray Images," *Symmetry*, 2021. 15

[78] D. Arias-Garzón, J. A. Alzate-Grisales, S. Orozco-Arias, H. B. Arteaga-Arteaga, M. A. Bravo-Ortiz, A. Mora-Rubio, J. M. Saborit-Torres, J. M. Serrano, M. d. l. I. Vayá, O. Cardona-Morales, and R. Tabares-Soto, "COVID-19 detection in X-ray images using convolutional neural networks," *Machine Learning with Applications*, 2021. 15

[79] Q. Yao, L. Xiao, P. Liu, and S. K. Zhou, "Label-Free Segmentation of COVID-19 Lesions in Lung CT," *IEEE Transactions on Medical Imaging*, 2021. 15

[80] A. J. DeGrave, J. D. Janizek, and S.-I. Lee, "AI for radiographic COVID-19 detection selects shortcuts over signal," *Nature Machine Intelligence*, vol. 3, pp. 610–619, July 2021. Number: 7 Publisher: Nature Publishing Group. 15, 17

[81] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," pp. 4700–4708, 2017. 16, 21

[82] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context Encoders: Feature Learning by Inpainting," pp. 2536–2544, 2016. 16

[83] J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, J. Seekins, D. A. Mong, S. S. Halabi, J. K. Sandberg, R. Jones, D. B. Larson, C. P. Langlotz, B. N. Patel, M. P. Lungren, and A. Y. Ng, "CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 590–597, July 2019. Number: 01. 16

[84] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. 19

[85] M. Contributors, "MMSelfSup: OpenMMLab Self-Supervised Learning Toolbox and Benchmark," 2021. 20

[86] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019. 20

[87] X. Xu, J. Lin, Y. Tao, and X. Wang, "An Improved DenseNet Method Based on Transfer Learning for Fundus Medical Images," in *2018 7th International Conference on Digital Home (ICDH)*, pp. 137–140, Nov. 2018. 21

[88] T. Chauhan, H. Palivela, and S. Tiwari, "Optimization and fine-tuning of DenseNet model for classification of COVID-19 cases in medical imaging," *International Journal of Information Management Data Insights*, vol. 1, p. 100020, Nov. 2021. 21

[89] Z. Huang, X. Zhu, M. Ding, and X. Zhang, "Medical Image Classification Using a Light-Weighted Hybrid Neural Network Based on PCANet and DenseNet," *IEEE Access*, vol. 8, pp. 24697–24712, 2020. Conference Name: IEEE Access. 21

[90] T. Zhou, X. Ye, H. Lu, X. Zheng, S. Qiu, and Y. Liu, "Dense Convolutional Network and Its Application in Medical Image Analysis," *BioMed Research International*, vol. 2022, p. e2384830, Apr. 2022. Publisher: Hindawi. 21

[91] L. Wen, X. Li, and L. Gao, "A transfer convolutional neural network for fault diagnosis based on ResNet-50," vol. 32, no. 10, pp. 6111–6124. 21

[92] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos, and P. de Geus, "Malicious software classification using transfer learning of ResNet-50 deep neural network," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1011–1014. 21

[93] A. S. B. Reddy and D. S. Juliet, "Transfer learning with ResNet-50 for malaria cell-image classification," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0945–0949. 21

[94] G. Pleiss, D. Chen, G. Huang, T. Li, L. van der Maaten, and K. Q. Weinberger, "Memory-efficient implementation of DenseNets." Number: arXiv:1707.06990. 21

[95] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), Lecture Notes in Computer Science, (Cham), pp. 234–241, Springer International Publishing, 2015. 22

[96] "DICOM." 22

[97] P. Mildenberger, M. Eichelberg, and E. Martin, "Introduction to the DICOM standard," *European Radiology*, vol. 12, pp. 920–927, Apr. 2002. 22

[98] "Portable Network Graphics (PNG) Specification (Second Edition)." 23

[99] D. Mason, scaramallion, mrbean bremen, rhaxton, J. Suever, Vanessasaurus, D. P. Orfanos, G. Lemaitre, A. Panchal, A. Rothberg, M. D. Herrmann, J. Massich, J. Kerns, K. v. Golen, T. Robitaille, S. Biggs, moloney, C. Bridge, M. Shun-Shin, B. Conrad, pawelzajdel, M. Mattes, Y. Lyu, F. C. Morency, T. Cogan, H. Meine, and J. Wortmann, "pydicom/pydicom: pydicom 2.3.0," Mar. 2022. 23

[100] S. Ruder, "An overview of gradient descent optimization algorithms," Tech. Rep. arXiv:1609.04747, arXiv, June 2017. arXiv:1609.04747 [cs] type: article. 26, 27

[101] Y. You, I. Gitman, and B. Ginsburg, "Large Batch Training of Convolutional Networks," Tech. Rep. arXiv:1708.03888, arXiv, Sept. 2017. arXiv:1708.03888 [cs] type: article. 27

[102] S. Haque and J. H. Chan, "The Effect of PreTraining Thoracic Disease Detection Systems on Large-Scale Chest X-Ray Domain Datasets," in *The 12th International Conference on Computational Systems-Biology and Bioinformatics*, CSBio2021, (New York, NY, USA), pp. 44–47, Association for Computing Machinery, Oct. 2021. 28

[103] M. Noroozi and P. Favaro, "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), Lecture Notes in Computer Science, (Cham), pp. 69–84, Springer International Publishing, 2016. 40

[104] B. Jia, J. Brandt, R. Mech, B. Kim, and D. Manocha, "LPaintB: Learning to Paint from Self-Supervision," Sept. 2019. Number: arXiv:1906.06841 arXiv:1906.06841 [cs]. 40

[105] X. Zhan, J. Xie, Z. Liu, Y.-S. Ong, and C. C. Loy, "Online Deep Clustering for Unsupervised Representation Learning," pp. 6688–6697, 2020. 40

[106] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of neural network representations revisited," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 3519–3529, PMLR. ISSN: 2640-3498. 40

Appendix

Appendix A

Appendix Environment

In this appendix we show the environment configurations in which we performed our experiments.

GPU Configuration

```
+-----------------------------------------------------------------------------+\
 NVIDIA-SMI 418.39       Driver Version: 418.39       CUDA Version: 10.1
-------------------------------+----------------------+----------------------+\
 GPU  Name        Persistence-M Bus-Id        Disp.A  Volatile Uncorr. ECC
 Fan  Temp  Perf  Pwr:Usage/Cap         Memory-Usage  GPU-Util  Compute M.
==============================+======================+======================
   0  GeForce RTX 208...  Off   00000000:01:00.0 Off                    N/A
 55%   59C    P0    67W / 260W       0MiB / 10989MiB       0%      Default
+-------------------------------+----------------------+----------------------+
```

Environment Configuration

```
# packages in environment at /media/DiscoLocal/MUDLAVSP/Ivan/Kirill/env:
#
# Name                    Version                   Build  Channel
_libgcc_mutex             0.1                        main
_openmp_mutex             4.5                        1_gnu
_pytorch_select           0.1                        cpu_0
absl-py                   1.0.0                    pypi_0    pypi
addict                    2.4.0                    pypi_0    pypi
backcall                  0.2.0                    pypi_0    pypi
blas                      1.0                         mkl
ca-certificates           2022.2.1              h06a4308_0
cachetools                5.0.0                    pypi_0    pypi
certifi                   2021.10.8         py37h06a4308_2
charset-normalizer        2.0.12                   pypi_0    pypi
cudatoolkit               10.1.243              h6bb024c_0
cycler                    0.11.0                   pypi_0    pypi
debugpy                   1.6.0                    pypi_0    pypi
decorator                 5.1.1                    pypi_0    pypi
entrypoints               0.4                      pypi_0    pypi
faiss-gpu                 1.6.1                    pypi_0    pypi
fonttools                 4.29.1                   pypi_0    pypi
freetype                  2.11.0                h70c0345_0
future                    0.18.2                   pypi_0    pypi
```

```
giflib                    5.2.1              h7b6447c_0
google-auth               2.6.0                  pypi_0    pypi
google-auth-oauthlib      0.4.6                  pypi_0    pypi
grad-cam                  1.3.7                  pypi_0    pypi
grpcio                    1.44.0                 pypi_0    pypi
idna                      3.3                    pypi_0    pypi
importlib-metadata        4.11.2                 pypi_0    pypi
intel-openmp              2021.4.0         h06a4308_3561
ipykernel                 6.13.0                 pypi_0    pypi
ipython                   7.32.0                 pypi_0    pypi
jedi                      0.18.1                 pypi_0    pypi
joblib                    1.1.0                  pypi_0    pypi
jpeg                      9d                 h7f8727e_0
jupyter-client            7.2.2                  pypi_0    pypi
jupyter-core              4.9.2                  pypi_0    pypi
kiwisolver                1.3.2                  pypi_0    pypi
lcms2                     2.12               h3be6417_0
ld_impl_linux-64          2.35.1             h7274673_9
libffi                    3.3                he6710b0_2
libgcc-ng                 9.3.0              h5101ec6_17
libgomp                   9.3.0              h5101ec6_17
libpng                    1.6.37             hbc83047_0
libstdcxx-ng              9.3.0              hd4cf53a_17
libtiff                   4.2.0              h85742a9_0
libuv                     1.40.0             h7b6447c_0
libwebp                   1.2.2              h55f646e_0
libwebp-base              1.2.2              h7f8727e_0
lz4-c                     1.9.3              h295c915_1
markdown                  3.3.6                  pypi_0    pypi
matplotlib                3.5.1                  pypi_0    pypi
matplotlib-inline         0.1.3                  pypi_0    pypi
mkl                       2021.4.0          h06a4308_640
mkl-service               2.4.0           py37h7f8727e_0
mkl_fft                   1.3.1           py37hd3c417c_0
mkl_random                1.2.2           py37h51133e4_0
mmcls                     0.21.0                 pypi_0    pypi
mmcv-full                 1.4.6                  pypi_0    pypi
mmdet                     2.22.0                 pypi_0    pypi
mmselfsup                 0.5.0                   dev_0    <develop>
ncurses                   6.3                h7f8727e_2
nest-asyncio              1.5.5                  pypi_0    pypi
ninja                     1.10.2          py37hd09550d_3
numpy                     1.21.2          py37h20f2e39_0
numpy-base                1.21.2          py37h79a1101_0
oauthlib                  3.2.0                  pypi_0    pypi
opencv-python             4.5.5.62               pypi_0    pypi
openssl                   1.1.1m             h7f8727e_0
packaging                 21.3                   pypi_0    pypi
pandas                    1.3.5                  pypi_0    pypi
parso                     0.8.3                  pypi_0    pypi
pexpect                   4.8.0                  pypi_0    pypi
pickleshare               0.7.5                  pypi_0    pypi
pillow                    9.0.1           py37h22f2fdc_0
pip                       21.2.2          py37h06a4308_0
prompt-toolkit            3.0.29                 pypi_0    pypi
protobuf                  3.19.4                 pypi_0    pypi
psutil                    5.9.0                  pypi_0    pypi
ptyprocess                0.7.0                  pypi_0    pypi
```

pyasn1	0.4.8	pypi_0	pypi
pyasn1-modules	0.2.8	pypi_0	pypi
pycocotools	2.0.4	pypi_0	pypi
pygments	2.11.2	pypi_0	pypi
pyparsing	3.0.7	pypi_0	pypi
python	3.7.11	h12debd9_0	
python-dateutil	2.8.2	pypi_0	pypi
pytz	2022.1	pypi_0	pypi
pyyaml	6.0	pypi_0	pypi
pyzmq	22.3.0	pypi_0	pypi
readline	8.1.2	h7f8727e_1	
requests	2.27.1	pypi_0	pypi
requests-oauthlib	1.3.1	pypi_0	pypi
rsa	4.8	pypi_0	pypi
scikit-learn	1.0.2	pypi_0	pypi
scipy	1.7.3	pypi_0	pypi
seaborn	0.11.2	pypi_0	pypi
setuptools	58.0.4	py37h06a4308_0	
six	1.16.0	pyhd3eb1b0_1	
sklearn	0.0	pypi_0	pypi
sqlite	3.37.2	hc218d9a_0	
tensorboard	2.8.0	pypi_0	pypi
tensorboard-data-server	0.6.1	pypi_0	pypi
tensorboard-plugin-wit	1.8.1	pypi_0	pypi
terminaltables	3.1.10	pypi_0	pypi
threadpoolctl	3.1.0	pypi_0	pypi
tk	8.6.11	h1ccaba5_0	
torch	1.7.1+cu101	pypi_0	pypi
torchaudio	0.7.2	pypi_0	pypi
torchvision	0.8.2+cu101	pypi_0	pypi
tornado	6.1	pypi_0	pypi
tqdm	4.63.0	pypi_0	pypi
traitlets	5.1.1	pypi_0	pypi
ttach	0.0.3	pypi_0	pypi
typing_extensions	3.10.0.2	pyh06a4308_0	
urllib3	1.26.8	pypi_0	pypi
wcwidth	0.2.5	pypi_0	pypi
werkzeug	2.0.3	pypi_0	pypi
wheel	0.37.1	pyhd3eb1b0_0	
xz	5.2.5	h7b6447c_0	
yapf	0.32.0	pypi_0	pypi
zipp	3.7.0	pypi_0	pypi
zlib	1.2.11	h7f8727e_4	
zstd	1.4.9	haebb681_0	

Appendix B

Appendix Results

In this appendix we show the tables from the results in more detail, indicationg all the LR that we tested and the different accuracies obtained.

Full Image - SSL Task				
LR	**Moco**	**Swav**	**Rel-Loc**	**Rotation**
0,01	8,259	7,889	**95,56**	**99,09**
0,039	**8,122**	7,774	93,8	99,06
0,068	8,38	7,699	94,29	98,72
0,097	8,217	7,592	94,82	98,5
0,126	8,281	7,559	95,38	99,06
0,155	8,306	7,483	94,15	97,97
0,184	8,398	**7,467**	91,39	98,81
0,213	8,341	7,468	93,92	96,78
0,242	8,423	7,599	92,63	97,5
0,271	8,362	7,583	93,88	97,72
0,3	8,662	7,61	93,82	97,78

Table B.1: SSL task performance with different learning rates over complete images. Relative Location and Rotation prediction are measured in accuracy, while MoCo and SwAV are over their loss.

Full Image - Classification						
LR	**Scrach**	**Imagenet**	**Moco**	**Swav**	**Rel-Loc**	**Rotation**
0,01	81,55	78,8	83,03	82,78	82,23	78,37
0,039	**83,69**	81,28	**83,71**	**83,29**	**83,66**	81,04
0,068	81,83	80,49	80,49	81,49	82,72	**81,59**
0,097	78,5	77,26	80,51	81,64	79,12	78,1
0,126	82,36	75,07	82,18	80,95	79,9	81,02
0,155	81,93	**82,18**	82,1	81,65	77,45	79,02
0,184	69,35	79,74	74,75	75,3	78,9	79,39
0,213	76,75	74,83	50	69,07	74,7	77,82
0,242	51,11	72,75	61,36	80,79	72,96	62,54
0,271	79,78	79,14	79,97	80,83	72,71	78,29
0,3	77,94	78,9	72,28	74,39	78,64	70,72
150 epochs	83,69	82,75	83,89	83,97	83,62	84,72

Table B.2: Classification accuracy with different learning rates and different pretrainings over complete images.

Segmented Black - SSL Task				
LR	**Moco**	**Swav**	**Rel-Loc**	**Rotation**
0,01	8,077	7,931	**86,88**	98,34
0,058	**8,067**	7,744	84,29	98,25
0,106	8,083	7,66	84,13	**98,44**
0,154	8,164	7,664	85,19	98,06
0,202	8,458	7,65	85,54	98,22
0,25	8,422	**7,595**	84,59	97,94

Table B.3: SSL task performance with different learning rates over segmented images. Relative Location and Rotation prediction are measured in accuracy, while MoCo and SwAV are over their loss.

Segmented black - Classification						
LR	**Scrach**	**Imagenet**	**Moco**	**Swav**	**Rel-Loc**	**Rotation**
0,01	73,03	**82,73**	78,02	**83,63**	**83,08**	77,91
0,058	**81,78**	77,09	**82,61**	82,87	72,43	74,18
0,106	73,42	74,81	79,95	79,77	78,43	**82,36**
0,154	62,53	67,24	70,49	75,7	70,12	78,79
0,202	76,79	73,78	63,95	65,79	72,54	78,22
0,25	72,39	51,5	71,83	77,17	52,19	73,12
150 epochs	81,78	82,73	82,6396	83,716	83,7985	83,428

Table B.4: Classification accuracy with different learning rates and different pretrainings over segmented images.

Full image Combined I - SSL Task												
LR	**Mocov2**			**Rotation**			**Rel-Loc**			**Swav**		
	Rotation	Rel-Loc	Swav	MocoV2	Rel-Loc	Swav	Rotation	MocoV2	Swav	Rotation	Rel-Loc	MocoV2
0,01	98,19	93,61	7,934	8,623	**93,74**	7,935	98,31	**7,785**	7,935	98,09	93,67	**8,349**
0,039	97,63	**94,39**	9,823	**8,327**	93,25	7,824	97,72	8,33	7,823	98,06	**95,09**	8,432
0,068	98,44	91,95	7,769	8,374	86,2	7,771	98,66	8,328	7,77	98,41	90,52	8,534
0,097	**98,84**	92,62	7,733	8,418	89,77	7,731	98,56	8,341	7,73	**99,22**	92,59	8,57
0,126	97,06	91,4	7,694	8,46	89,38	7,698	98,06	8,382	7,701	98,31	91,05	8,561
0,155	98,78	93,77	**7,632**	8,438	86,12	**7,636**	**98,81**	8,419	7,629	98,16	89,78	8,629
0,184	98,5	91,4	7,65	8,447	91,76	7,641	98,03	8,443	**7,657**	98,38	91,29	8,677
0,213	98,56	93,02	7,671	8,461	91,38	7,694	98,59	8,489	7,653	98,47	90,09	8,626
0,242	98,09	91,63	7,687	8,518	92,35	7,687	98,03	8,496	7,689	98,09	91,18	8,644
0,271	97,44	91,39	7,689	8,514	89,55	7,681	97,22	8,554	7,683	97,63	91,91	8,67
0,3	97,28	89,52	7,71	8,542	89,74	7,718	97,28	8,564	7,704	97,25	87,73	8,701

Table B.5: SSL task performance with different learning rates over complete images. The order for the SSL task is from top to bottom. The second column refers to a model first pretrained with MoCoV2 and then with Rotation prediction. Relative Location and Rotation prediction are measured in accuracy, while MoCo and SwAV are over their loss.

Full Image Combined I - Classification												
LR	**Mocov2**			**Rotation**			**Rel-Loc**			**Swav**		
	Rotation	**Rel-Loc**	**Swav**	**MocoV2**	**Rel-Loc**	**Swav**	**Rotation**	**MocoV2**	**Swav**	**Rotation**	**Rel-Loc**	**MocoV2**
0,01	82,24	**82,61**	**83,61**	76,74	**84,31**	**83,1**	**82,63**	**83,98**	82,14	**81,52**	**84,69**	**82,07**
0,058	**83,06**	82,17	80,6	**80,3**	77,38	80,78	81,88	61,81	**82,68**	80,93	78,85	59,8
0,106	79,93	72,8	76,16	79,49	68,59	75,93	81,35	62,31	76,02	79,57	79,91	71,14
0,154	77,75	80,28	81,18	78,61	75,61	80,13	82,17	81,69	75,98	75,34	78,2	77,54
0,202	79,02	80,52	72,65	70,12	71,73	73,73	80,83	58,43	81,02	81,17	71,42	80,71
0,25	79,88	73,89	68,89	78,5	64,19	78,12	79,46	62,78	69,97	69,54	50	57
150 epoch	84,7728	**85,59**	83,67	76,087	84,338	**84,81**	84,5476	82,7957	**85,27**	83,89	**84,92**	82,374

Table B.6: Classification accuracy with different learning rates and different combined pre-trainings over complete images. The order for the SSL task is from top to bottom. The second column refers to a model first pretrained with MoCoV2 and then with Rotation prediction.

Segmented Black Combined II - SSL												
LR	**Mocov2**			**Rotation**			Rel-Loc			Swav		
	Rotation	Rel-Loc	Swav	MocoV2	Rel-Loc	Swav	Rotation	MocoV2	Swav	Rotation	Rel-Loc	MocoV2
0,01	98,97	**87,02**	7,82	**8,15**	86,07	7,935	99,13	**7,945**	7,767	**99,06**	86,55	**8,337**
0,058	**99,44**	80,27	7,575	8,359	**86,36**	7,701	**99,56**	8,495	7,528	98,94	**87,02**	8,468
0,106	98,5	83,78	**7,507**	8,415	85,53	7,594	97,66	8,524	7,514	98,31	84,21	8,662
0,154	97,63	84,34	7,536	8,427	85,34	7,583	97,16	8,567	7,465	97,91	83,09	8,768
0,202	98,38	84,64	7,58	8,449	84,41	7,583	97,34	8,632	**7,463**	98,56	84,34	8,678
0,25	97,13	85,22	7,607	8,513	84,67	**7,558**	97,56	8,692	7,661	97,31	82,33	8,866

Table B.7: SSL task performance with different learning rates over segmented images. The order for the SSL task is from top to bottom. The second column refers to a model first pretrained with MoCoV2 and then with Rotation prediction. Relative Location and Rotation prediction are measured in accuracy, while MoCo and SwAV are over their loss.

Segmented Black Combined II - Classification												
LR	**Mocov2**			**Rotation**			**Rel-Loc**			**Swav**		
	Rotation	**Rel-Loc**	**Swav**	**MocoV2**	**Rel-Loc**	**Swav**	**Rotation**	**MocoV2**	**Swav**	**Rotation**	**Rel-Loc**	**MocoV2**
0,01	**83,57**	**82,82**	59,46	80,21	**83,07**	**83,59**	**83,94**	**83,77**	**83,03**	**83,09**	**83**	**82,34**
0,058	80,26	78,98	75,77	**83,1**	80,19	80,08	74,91	75,37	81,49	74,18	80,01	73,18
0,106	53,15	76,09	**82,45**	62,83	79,06	69,15	68,87	66,96	51,49	72,75	76,75	71,33
0,154	80,87	64,43	72,63	80,18	71,01	73,18	80,58	72,59	72,08	59,4	76,8	56,37
0,202	50	75,28	72,68	74,23	52,79	72,21	70,68	70,43	73,29	72,63	61,85	71,89
0,25	76,76	58,53	52,99	70,41	65,26	73,09	67,41	60,01	69,02	77,73	60,49	61,5
150 epoch	**84,5554**	84,296	81,3488	77,3361	83,3238	**84,4314**	**83,7985**	81,148	83,6823	81,2975	82,7386	**84,3495**

Table B.8: Classification accuracy with different learning rates and different combined pre-trainings over segmented images. The order for the SSL task is from top to bottom. The second column refers to a model first pretrained with MoCoV2 and then with Rotation prediction.

Full image Combined III - SSL Task						
Mocov2						
LR	**Rotation**		**Rel-Loc**		**Swav**	
	Rel-Loc	**Swav**	**Rotation**	**Swav**	**Rel-Loc**	**Rotation**
0,01	**94,52**	7,994	**99,59**	7,889	**94,9**	98,78
0,058	81,87	7,737	98,41	**7,669**	81,21	**99,53**
0,106	90,54	**7,682**	97,38	7,767	81,99	98,84
0,154	88,59	7,793	99,34	7,851	86,55	98,19
0,202	90,27	7,837	98,41	7,818	88,27	98,97
0,25	88,04	7,803	98,16	7,88	88,37	97,5

Table B.9: SSL task performance with different learning rates over complete images. The order for the SSL task is from top to bottom. The second column refers to a model first pretrained with MoCoV2, then with rotation prediction, and finally with relative location. Relative Location and Rotation prediction are measured in accuracy, while MoCo and SwAV are over their loss.

Full image Combined III - Classification						
Mocov2						
LR	**Rotation**		**Rel-Loc**		**Swav**	
	Rel-Loc	**Swav**	**Rotation**	**Swav**	**Rel-Loc**	**Rotation**
0,01	81,83	81,84	**81,46**	**85,22**	**84,08**	**84,67**
0,058	**84,07**	**83,88**	81,36	83,08	81,58	82,84
0,106	81,45	81,4	78,34	82,46	80,46	81,92
0,154	82,91	69,02	76,52	73,32	80,2	81,88
0,202	73,46	81,03	77,52	80,04	69,77	84,28
0,25	79,59	79,25	75,44	76,08	64,13	79,63
150 epoch	85,28	84,8	84,19	85,49	85,67	83,74

Table B.10: Classification performance with different learning rates over complete images. The order for the SSL task is from top to bottom. The second column refers to a model first pretrained with MoCoV2, then with Rotation prediction, and finally with relative location. Relative Location and Rotation prediction are measured in accuracy, while MoCo and SwAV are over their loss.

Segmented Black Combined III - SSL Task						
Mocov2						
LR	**Rotation**		**Rel-Loc**		**Swav**	
	Rel-Loc	**Swav**	**Rotation**	**Swav**	**Rel-Loc**	**Rotation**
0,01	87,32	7,827	98,91	7,722	**86,37**	98,94
0,058	**87,83**	7,557	**99,5**	7,508	86,32	**99,28**
0,106	87,56	7,521	98,63	7,449	83,65	97,91
0,154	84,98	**7,491**	96,06	7,456	83,14	97,19
0,202	84,42	7,542	95,97	**7,434**	83,58	98,5
0,25	84,22	7,509	97,78	7,467	84,13	97,47

Table B.11: SSL task performance with different learning rates over segmented images. The order for the SSL task is from top to bottom. The second column refers to a model first pretrained with MoCoV2, then with Rotation prediction, and finally with relative location. Relative Location and Rotation prediction are measured in accuracy, while MoCo and SwAV are over their loss.

Segmented Black Combined III - Classification						
Mocov2						
LR	**Rotation**		**Rel-Loc**		**Swav**	
	Rel-Loc	**Swav**	**Rotation**	**Swav**	**Rel-Loc**	**Rotation**
0,01	**83,05**	**83,64**	**84**	**84,02**	**83,35**	**83,55**
0,058	77,45	72,43	73,12	81,42	80,97	71,21
0,106	80,64	80,4	72,26	75,84	71,2	80,4
0,154	76,22	79,96	72,29	66,66	78,01	56,99
0,202	75,53	73,35	73,41	79,36	77,28	67,03
0,25	60,36	72,5	62,64	64,73	52,42	61,3
150 epoch	84,38	85,14	83,48	84,64	84,67	84,71

Table B.12: Classification performance with different learning rates over segmented images. The order for the SSL task is from top to bottom. The second column refers to a model first pretrained with MoCoV2, then with Rotation prediction, and finally with relative location. Relative Location and Rotation prediction are measured in accuracy, while MoCo and SwAV are over their loss.

Appendix C

Appendix Attention

In this appendix we show the class attention maps for all the models.

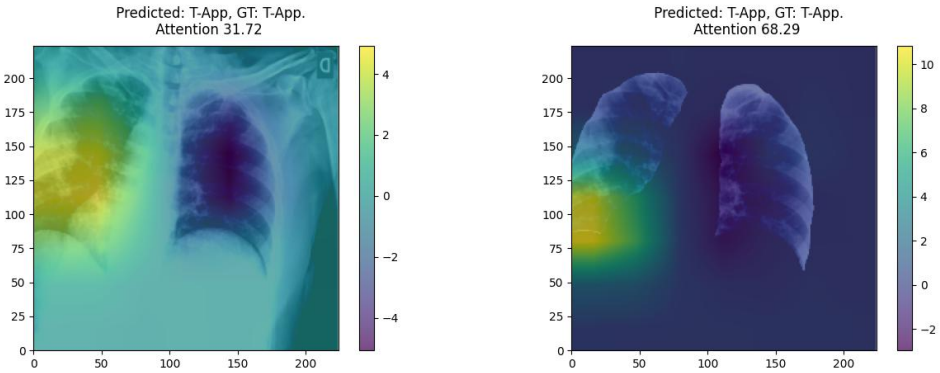


Figure C.1: Bigger image of the attention maps

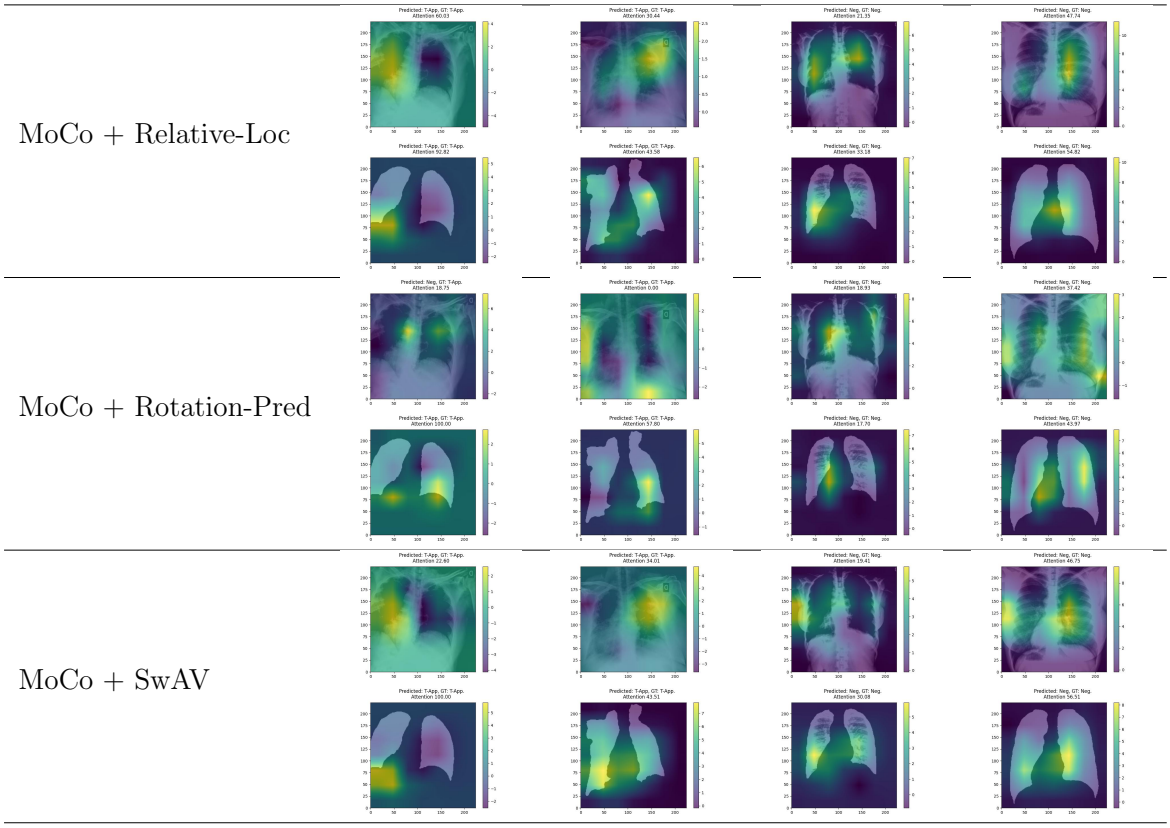


Figure C.2: Class attention maps for combined pretraining over complete images (Starting with MoCo-V2).

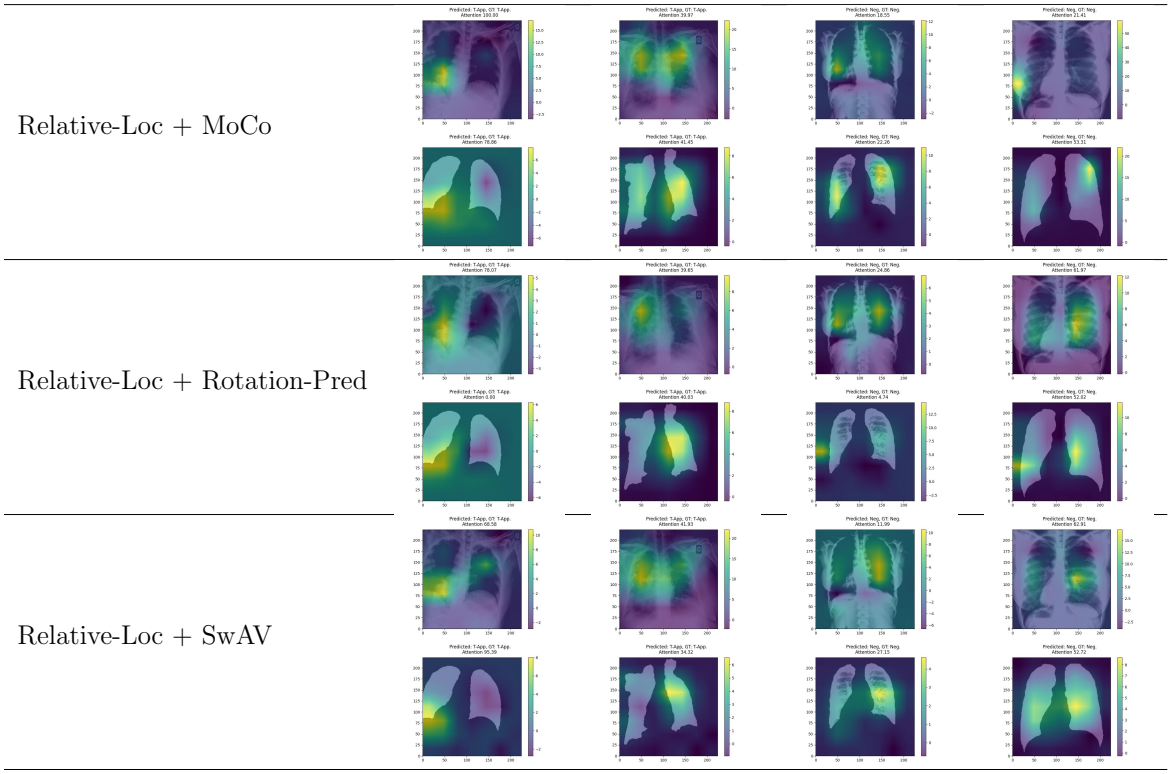


Figure C.3: Class attention maps for combined pretraining over complete images (Starting with Relative-Location).

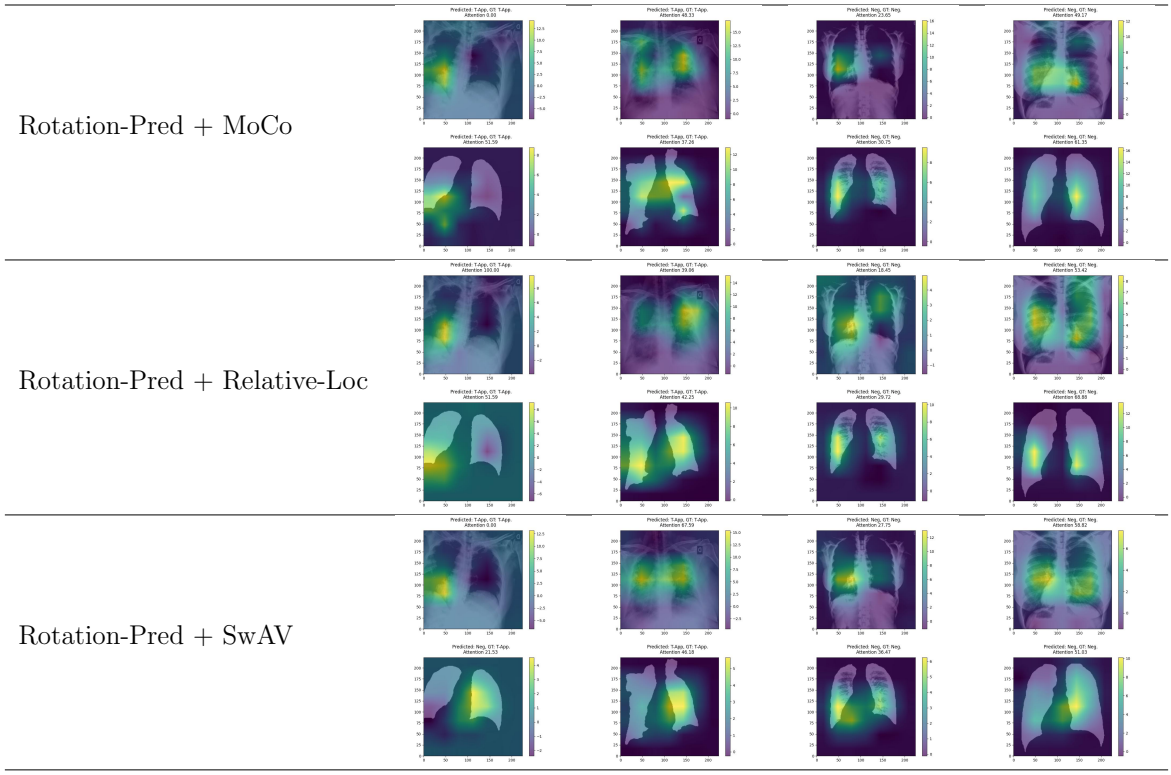


Figure C.4: Class attention maps for combined pretraining over complete images (Starting with Rotation-Prediction).

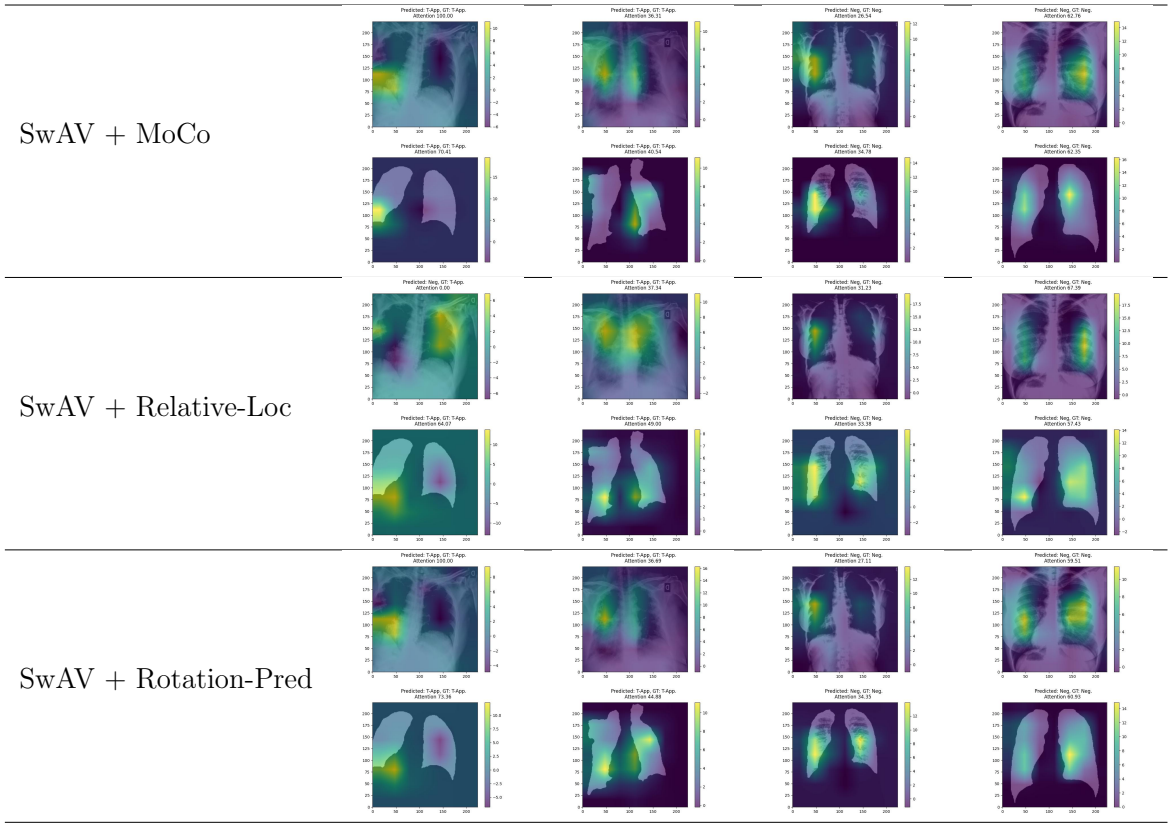


Figure C.5: Class attention maps for combined pretraining over complete images (Starting with SwAV).

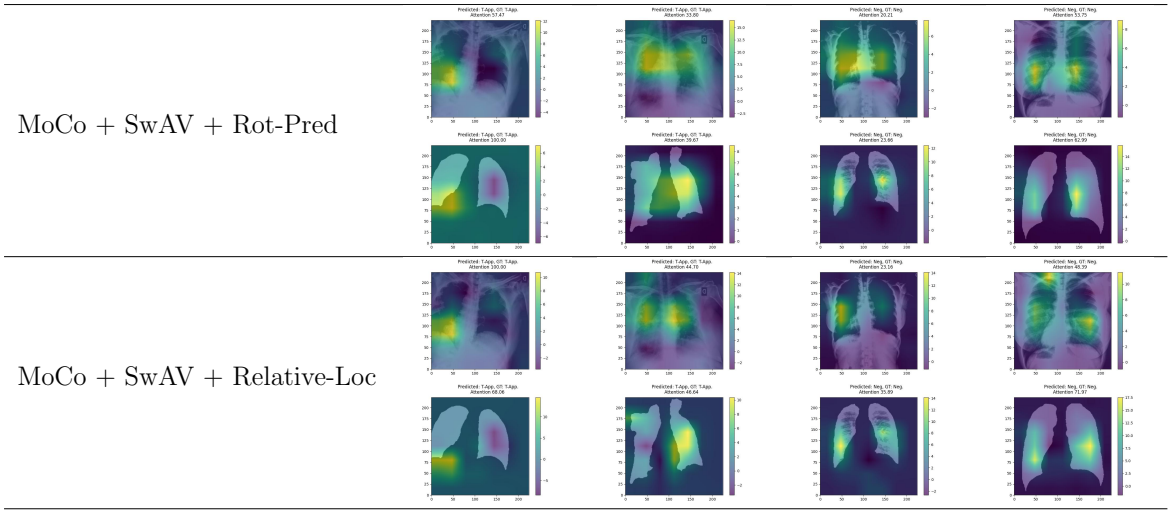


Figure C.6: Class attention maps for combined pretraining over complete images (Starting with MoCo + SwAV).

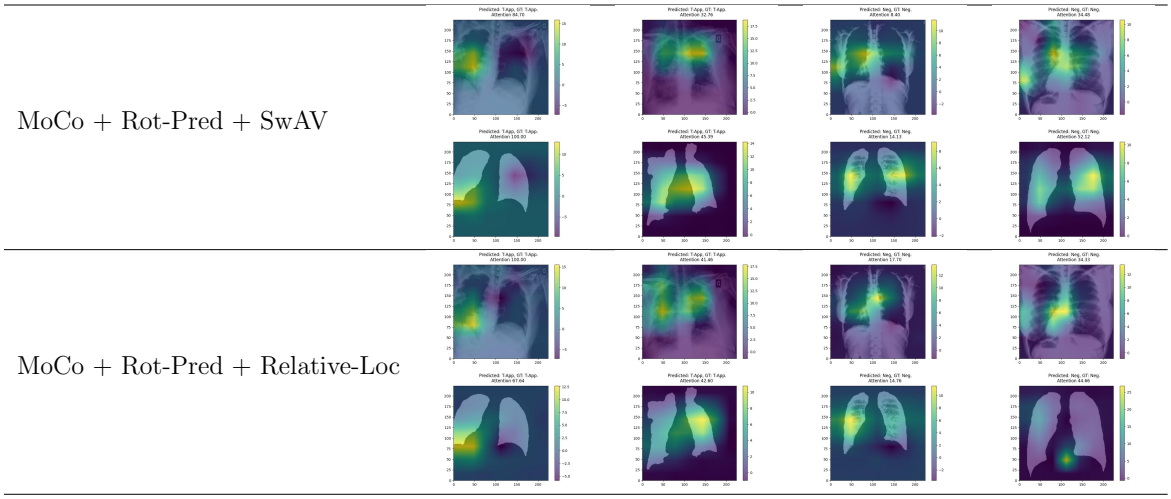


Figure C.7: Class attention maps for combined pretraining over complete images (Starting with MoCo + Rotation-Prediction).

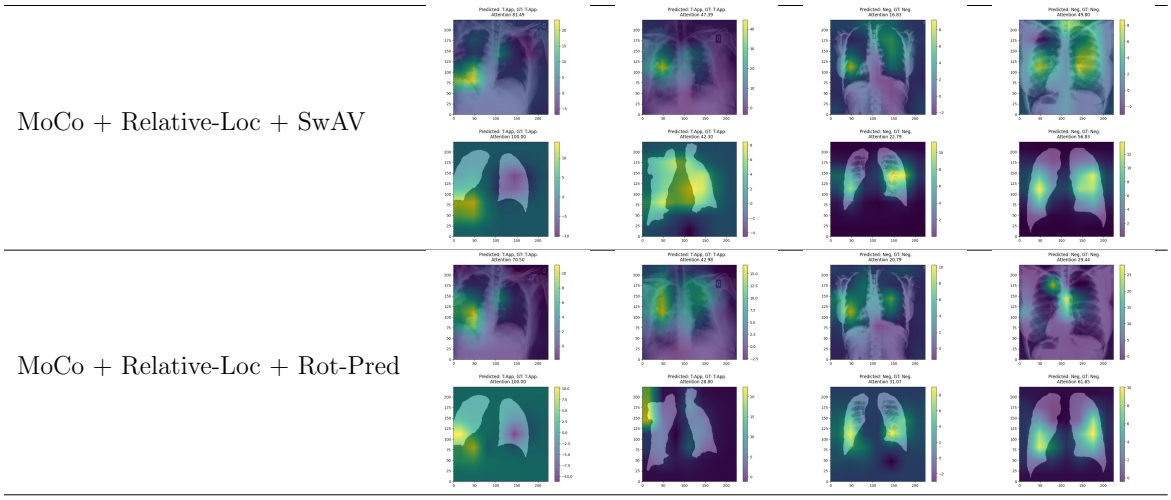


Figure C.8: Class attention maps for combined pretraining over complete images (Starting with MoCo + Relative-Location).