MSc Artificial Intelligence
Master Thesis

Self-supervised learning with group equivariant networks for histopathological images

by
Błażej Dolicki
13244752

December 24, 2022

48 ECTS
November 2021 - August 2022

Supervisor:
Yoni Schirris MSc

Examiner:
Dr E Bekkers

Second reader:
Y Schirris

Universiteit van Amsterdam

Contents

Abstract

Applications of deep learning in histopathology have recently become possible due to digitization of medical images which are obtained during standard treatment of cancer patients. This can potentially allow faster, more accurate diagnosis and better survival prediction. However, most of the available data is not annotated which is a necessary requirement for conventional deep learning. Self-supervised learning allows training neural networks without labelled images and therefore is particularly beneficial in digital pathology. We aim to improve self-supervised models for histopathology by leveraging group equivariant networks which are by design invariant to image rotations. We show that while adding rotations as image augmentations to standard convolutional networks improves rotation-invariance of the model, it can harm the performance. Conversely, in some settings, equivariant networks achieve both excellent rotation-invariance and high accuracy on downstream datasets. We make our code publicly available on GitHub [1]

[1] https://github.com/blazejdolicki/rissl

Acknowledgements

Firstly, I would like to thank my supervisor, Yoni Schirris, for always making time to discuss my progress, providing valuable feedback and ensuring that apart from creating meaningful scientific work I also enjoyed the whole process as much as possible. I would also like to thank Dr Erik Bekkers for agreeing to be my examiner. Finally, I am very grateful to my family and friends, for their continuous support throughout this entire period.

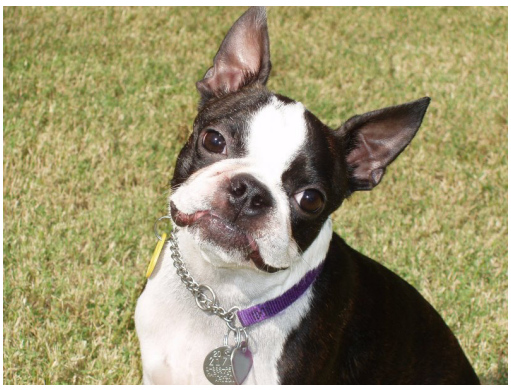
Chapter 1

Introduction

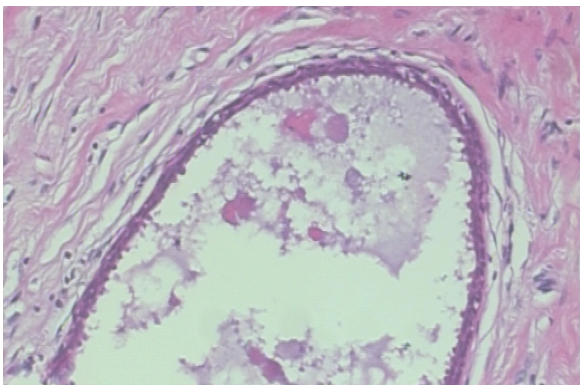
In recent years, pretrained neural networks became a standard in deep learning. Such models are first trained on large datasets with diverse data distribution such as ImageNet (Russakovsky et al., 2015) and then finetuned on smaller, domain-specific datasets. The goal of pretraining is to learn general, useful image representations that can be later tuned for specific areas. It has been shown that using pretrained weights for initialization of training on domain-specific datasets outperforms random weight initialization (i.e. training from scratch) (Donahue et al., 2014; Sharif Razavian et al., 2014). However, the pretraining step requires vast amounts of labelled data which is often difficult or even impossible to obtain. Therefore, self-supervised learning (SSL) was proposed which performs pretraining on data without labels while maintaining results competitive with supervised pretraining and displaying many beneficial properties (Chen et al., 2020a; Caron et al., 2021; Goyal et al., 2022). This is possible, for instance, by changing the objective from predicting the correct label to learning image representations such that similar images have similar representations and different images have different representations. Nevertheless, this task itself is relatively easy and might lead the model to take shortcuts that decrease the quality of learnt representations (e.g. if all images with cats contain black cats, the model might learn that all images with black patches contain cats and will not recognize cats in different colors). For that reason these techniques apply data augmentations to perturb irrelevant information and thus ensure that the model learns meaningful image representations (Chen et al., 2020a). Let us continue with an example image of a cat. A common augmentation is random cropping because the raw RGB values of the image change significantly (preventing the model to find trivial solutions) while the semantic information (which we aim to capture) remains the same - regardless if the crop displays only the head or the paws, there is still a cat in the image. These augmentations make the model insensitive to irrelevant information of the image.

In histopathology, the regular treatment includes extracting a sample of patient's tissue in order to observe it under a microscope and diagnose the disease and its progression. Nowadays, these samples are often digitized which produces whole slide images (WSI). After examination, a pathologist provides a diagnosis which can serve as a label of the whole slide image used in deep learning. Unfortunately, these microscopic images have enormous resolutions (often more than 50,000 x 50,000 pixels) which makes them infeasible to fit in a neural network due to GPU memory constraints. They are often split into smaller images called patches or tiles with smaller resolution such as 256 x 256 pixels. WSI-level labels cannot be trivially converted to tile-level labels because a whole slide image labelled as containing tumor tissue, usually has a subgroup of patches with tumor while the rest of tiles are perfectly healthy (Schirris et al., 2021). Therefore, these circumstances produce enormous datasets of unlabelled images which, as aforementioned, is a perfect application for self-supervised learning.

Even though many new self-supervised methods have been published in recent months, most of them are tailored for benchmark datasets such as ImageNet (Russakovsky et al., 2015). The data distribution of these benchmarks (which often contain natural images) is drastically different from



(a) ImageNet



(b) BreaKHis

Figure 1.1: Examples from a standard dataset (ImageNet) with natural images compared to a histopatholoy dataset (BreaKHis) with microscopic images.

histopathology images and thus the proposed advances might not transfer well from one domain to another. Figure 1.1 shows examples of images from ImageNet (Russakovsky et al., 2015) and from a histopathology dataset (Spanhol et al., 2016). Clearly, there are many properties in which these domains differ. For instance, in natural images there is usually an object in the foreground and a background while in histopathology there is no image depth. The colors in ImageNet can be much more diverse while in histopathology due to staining the images are mostly limited to shades of pink and purple. Natural images tend to have a fixed, intuitive orientation (e.g. in a picture of a house, the roof tends to be in the top part of the image while the doors tend to be in the bottom). Conversely, the orientation in histopathology images, which display human tissue scanned under microscope, can be arbitrary. Regardless how they are rotated, the semantic meaning is preserved (e.g. the scanned image either contains a tumour, or a healthy tissue). While all of these properties could be valuable research directions, in this work we focus on the latter. Namely, we aim to reflect the irrelevance of image rotation which is obvious for human pathologists, but non-trivial for deep learning models - the model should return the same outputs given the same image under different rotations (i.e. it should be rotation-invariant). Previous work in SSL for histopathology (Ciga et al., 2021a) attempted to indirectly learn rotation-invariant representations by adding rotations as image augmentations during pretraining. However, there are no guarantees that this approach assures rotation-invariance and it might even lead the model to learning multiple features that are effectively rotated versions of each other which introduces redundancy in parameters (Olah et al., 2020). To that end, we propose to employ group equivariant networks (Cohen and Welling, 2016a) that explicitly incorporate rotation-invariance into the network architecture. By leveraging inherent symmetries of the data, equivariant models have shown promising results which suggest that they are more sample-efficient (Lafarge et al., 2021) (i.e. they require less data) and demonstrate faster convergence (Weiler and Cesa, 2019)(i.e. less training steps are necessary to reach high performance). Nevertheless, these properties have only been observed for randomly initialized models trained in a supervised manner and it remains unknown if they hold for self-supervised learning.

Therefore, in this work we attempt to replace rotation augmentations by making the network architecture rotation-invariant and verify if enhanced parameter sharing improves their efficiency. We examine our proposed method, Rotation-Invariant Self-supervised Learning (RISSL), on a diverse group of datasets with varying sample size, image size and number of classes.

1.1 Research questions

Given the aforementioned motivation, throughout this work we aim to answer the following research questions (**RQs**):

- **RQ1:** Do rotation augmentations in self-supervised learning empirically enhance rotation-invariance of model predictions? Are they sufficient to match the robustness to rotations of equivariant networks?
- **RQ2:** Does the rotation-invariance provided by equivariant networks translate to improved performance on downstream tasks in self-supervised learning?
- **RQ3:** Given the increased sample-efficiency of equivariant models trained from scratch, we investigate the effect of dataset size during self-supervised pretraining. More specifically, we aim to answer the following research question: Can the equivariant encoder also learn useful self-supervised representations with fewer data samples than a non-equivariant CNN model?

1.2 Thesis structure

The rest of this thesis is structured as follows. After the motivation and research questions presented in Chapter 1, we introduce the theoretical foundations of group theory in Chapter 2 which are essential to understand the recent developments in group equivariant networks. Once we establish such basis, we review prior work in equivariant networks and self-supervised learning in Chapter 3. We commence Chapter 4 by describing specifically the two main methods we combine in this work - MoCo and steerable CNNS. Afterwards, we explain the implementation of our equivariant model and conclude with proposing a new metric to quantitatively evaluate rotation-invariance. Chapter 5 defines the experiments performed to address the research questions, justifies experiment design choices and presents all the information required to reproduce this work. Afterwards, we show the results in Chapter 6. We interpret them in Chapter 7 to answer the research questions, reflect on limitations of our research and provide recommendations for future work. Finally, we summarize and conclude the thesis in Chapter 8. Our code, together with experiment logs and pretrained models is publicly available on GitHub [1].

[1] https://github.com/blazejdolicki/rissl

Chapter 2

Mathematical background

While equivariant neural networks possess quite intuitive benefits and applications in deep learning, they have strong mathematical foundations and to understand their details some familiarity with group theory is required. In this section, we provide the necessary background and terminology together with concrete examples which will allow to understand this work and the rest of the literature concerning equivariant networks.

2.1 Group theory

A **group** $(G, \circ)$ is a set G of elements along with a binary operation $\circ$ satisfying the following properties:

- it is associative, i.e. $\forall g, h, i \in G \quad (g \circ h) \circ i = g \circ (h \circ i)$
- there exists an identity element e such that $e \circ g = g \circ e = g$
- $\forall g \in G$ there exists $g^{-1} \in G$ such that $g \circ g^{-1} = g^{-1} \circ g = e$

A simple example is the set of non-zero real numbers $\mathbb{R}_{\neq 0}$ which forms a group under multiplication $\cdot$. We can verify that it is a group by checking that it fulfills the properties defined above.

It is associative because

$$\forall a, b, c \in \mathbb{R}_{\neq 0} \quad (a \cdot b) \cdot c = a \cdot (b \cdot c) \tag{2.1}$$

for example

$$(4 \cdot 2) \cdot 3 = 4 \cdot (2 \cdot 3) = 24 \tag{2.2}$$

There exists an identity element:

$$\forall a \in \mathbb{R}_{\neq 0} \quad 1 \cdot a = a \cdot 1 = a. \tag{2.3}$$

And finally, for every group element, there exists an inverse:

$$\forall a \in \mathbb{R}_{\neq 0}, \exists a^{-1} = \frac{1}{a} \quad \text{because} \quad a \cdot \frac{1}{a} = \frac{1}{a} \cdot a = 1 \tag{2.4}$$

To simplify the notation, we will often skip the group operator $\circ$.

A group with a finite number of elements in its set is called a **finite group**. Conversely, an **infinite group** has an infinite number of elements in its set. The number of elements in a group is called the **order of a group**. A group $(H, \circ)$ is a **subgroup** of another group $(G, \circ)$ if the set H is a subset of G and these two groups have the same operation $\circ$. In some cases, combining two groups together

creates a new group. Such group G is formally called the **semi-direct product** $G = I \rtimes H$ consisting of subgroups I and H (where one of the subgroups has to be a normal subgroup).

A **group action** of G on a domain X is a mapping $(g, x) \rightarrow g.x$ associating a group element $g \in G$ and an element $x \in X$ with some other element in X in a way that is compatible with group operations $g.(h.x) = (gh).x \quad \forall g, h \in G \wedge x \in X$ etc.). A domain X with such properties is called a G**-space** or G-set.

Every group element can be represented as a matrix which transforms (acts on) an element of a G-space. Formally, we define a **group representation** which is a map $\rho : G \rightarrow \mathbb{R}^{m \times m}$ that assigns to each group element g an invertible matrix such that:

$$\rho(gh) = \rho(g)\rho(h) \quad \forall g, h \in G \tag{2.5}$$

where m is the dimension of the G-space. Representations can be classified into different types. The simplest type of group representations is the trivial representation ρ_{triv} such that $\rho_{triv}(g) = 1, \forall g \in G$.

Every group G has a special type of representation, called the regular representation of G (Serre et al., 1977; Weiler and Cesa, 2019). This representation maps each axis (basis) e_g (where $g \in G$) in the representation space $\mathbb{R}^{|G|}$ to another axis $e_{\tilde{g}}$ (where $\tilde{g} \in G$):

$$\rho_{\text{reg}}(\tilde{g})e_g := e_{\tilde{g}g} \tag{2.6}$$

Let us introduce a number of important groups often appearing in the literature which are summarized in Table 2.1 and can provide helpful examples of the aforementioned terms. The Euclidean group $E(2)$ is a group of transformations of $\mathbb{R}^2$ that preserves Euclidean distances and consists of translations, rotations and reflections. The special Euclidean group $SE(2)$ is a subgroup of $E(2)$ and contains translations and rotations. The special orthogonal group $SO(2)$ is a subgroup of $SE(2)$ and concerns only continuous rotations. $SE(2)$ is equal to a semi-direct product $SE(2) = (\mathbb{R}^2, +) \rtimes SO(2)$. In practice, usually rotation-equivariant networks are not equivariant to the continuous group $SE(2)$, but to its finite subgroups $SE(2, N) = (\mathbb{R}^2, +) \rtimes C_N$ (Lafarge et al., 2021) which includes translations and N rotations. We can extend this group to include reflections which can be formally denoted as $(\mathbb{R}^2, +) \rtimes D_N$.

C_N is a finite cyclic group of order N and can be interpreted as a group of N rotations by angles which are multiplications of $\frac{2\pi}{N}$ (Weiler and Cesa, 2019). For instance, C_4 includes 4 rotations by multiplications of $\frac{2\pi}{N} = \frac{\pi}{2}$, i.e. rotations by 0, 90, 180 and 270 degrees. The group representation of C_N is a rotation matrix:

$$R = \begin{pmatrix} \cos\frac{2\pi k}{N} & -\sin\frac{2\pi k}{N} \\ \sin\frac{2\pi k}{N} & \cos\frac{2\pi k}{N} \end{pmatrix}, k \in \{0, 1, 2, ..., N\} \tag{2.7}$$

Let us provide a simple example with C_4 to improve the intuition behind group actions and representations. Consider the G-space to be a Euclidean plane i.e. $X = \mathbb{R}^2$. If we pick a specific group element of C_4 such as the second rotation ($k = 1$) with its representation R_1 (which we obtain by plugging $k = 1$ and $N = 4$ into Equation 2.7) and a specific point $x = (1, 2) \in \mathbb{R}^2$ then R_1 acts on x and yields a new point in $\mathbb{R}^2$.

$$R_1 x = \begin{pmatrix} \cos\frac{\pi}{2} & -\sin\frac{\pi}{2} \\ \sin\frac{\pi}{2} & \cos\frac{\pi}{2} \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} -2 \\ 1 \end{pmatrix} \tag{2.8}$$

Given that CNNs are equivariant to translations, any group H-equivariant layer that extends the translation group with a semi-direct product is equivariant under group $G = (\mathbb{R}^2, +) \rtimes H$. Therefore, to simplify the notation, when discussing various groups we will refer to the subgroup H, such as C_N or D_N, as translation equivariance is always present.

Group	Type	Elements
$E(2)$	infinite	translations, rotations, reflections
$SE(2)$	infinite	translations, rotations
$SO(2)$	infinite	rotations
$SE(2,N)$	finite	translations, rotations
C_N	finite	rotations
D_N	finite	rotations, reflections

Table 2.1: Relation between several groups related to rotation invariance.

A function f is said to be **equivariant** under the transformation group G and the domain X if

$$\forall x \in X, \forall g \in G : f(g.x) = g.f(x) \tag{2.9}$$

On the other hand, function f is said to be **invariant** if

$$\forall x \in X, \forall g \in G : f(g.x) = f(x) \tag{2.10}$$

In other words, for G-equivariant functions, applying g to the input x and afterwards applying f yields the same result as first applying f to x and subsequently passing the result to g (Figure 2.1). For G-invariant functions, acting with g on x and afterwards passing it to f yields the same results as passing x to f. Invariance is a special case of equivariance.

In the context of deep learning, the function f is a neural network that maps raw pixel values of an image into a vector that is supposed to represent that image. The group element g transforms the image in some way for example by rotating it. In some cases, it is beneficial for the representations produced by f to be invariant to certain groups such as rotations or changes in color and brightness (Figure 2.2). However, even for those cases equivariance should be preserved for as many layers as possible (Cohen and Welling, 2016a). For instance, when creating networks invariant to a group of rotations, in early layers where low-level features are learnt such as edges, the rotation of features is important information that should be retained. Only the final image representation or prediction should be rotation-invariant.

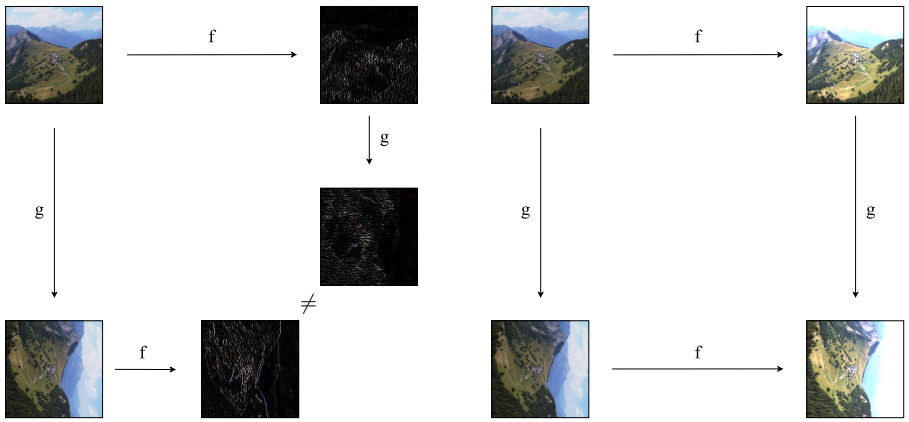


Figure 2.1: On the left, g is a rotation by 90 degrees while f applies an edge detector. The function f is not equivariant to g as $f(g(x)) \neq g(f(x))$. Conversely, on the right, g is a rotation while f increases brightness. These functions are equivariant $f(g(x)) = g(f(x))$.

2.2 Group equivariant CNNs

Let $\Psi = \{\psi^1, ..., \psi^K\}$ be a filter bank consisting of K filters. It is commonly known that in standard CNNs a learnable kernel ψ^k slides over the image (or feature map) which makes them equivariant

Figure 2.2: Objects on the picture differ only in color. Image representations returned by the encoder e_{inv} which is invariant to color will be the same while for the equivariant encoder e_{eqv} they will differ in a predictable manner.

to translations. Mathematically we define a discrete cross-correlation (usually called convolution), denoted with a star:

$$\left[f \star \psi^k\right](t) = \sum_{y \in \mathbb{Z}^2} \psi^k(y - t) f(y) \tag{2.11}$$

where f is the input signal (input image or a feature map from the previous layer), y are the coordinates on the input space and t is the translation offset.

The kernel is transformed by the translation and then applied to f. We can generalize this operation to other transformations which are defined by a group G and obtain the group convolution of the first layer:

$$\left[f \star_G \psi^k\right](g) = \sum_{y \in \mathbb{Z}^2} \psi^k(g^{-1} y) f(y) \tag{2.12}$$

Given that the feature map $f \star_G \psi^k$ is a function on a discrete group G, for all layers apart from the first, the group convolution is:

$$\left[f \star_G \psi^k\right](g) = \sum_{g' \in G} \psi^k(g^{-1} g') f(g') \tag{2.13}$$

where g' is a group element from the previous layer. Group convolutions are G-equivariant (Cohen and Welling, 2016a).

In conclusion, we just showed that convolution is a special case of group convolution where the group acting on the kernel is a group of translations $\mathbb{T}$. This generalization is a crucial step in deriving group CNNs.

We can now define a recipe for creating group equivariant networks. The first hidden layer should lift the input to a feature map with a higher dimension defined on a group G. This corresponds to Equation 2.12 and is often called a "lifting layer" (Lafarge et al., 2021). Afterwards, instead of using a convolution layer, we apply a group convolution layer using Equation 2.13. Similarly to standard convolutions, this layer is followed by a non-linearity and pooling (which have to be equivariant). Such block of group convolution, non-linearity and pooling can be repeatead multiple times. Until this point, we retain group equivariance. Finally, in the last layer we perform pooling over group elements which yields a group invariant representation.

While this general description holds for various groups, we focus on the group of discrete roto-translations $SE(2, N) = (\mathbb{R}^2, +) \rtimes C_N$ where the group equivariant network is equivariant to N rotations and N is a model hyperparameter.

Chapter 3

Related literature

This section describes the relevant literature on self-supervised learning and group equivariant networks. It shows how our work builds upon the current methods and how it aims to address its limitations.

3.1 Self-supervised learning

As discussed in Chapter 1, self-supervised learning allows to train deep learning models without annotated labels which is especially valuable in domains where data annotation is expensive or difficult. Initial work in self-supervised learning leveraged pretext tasks such as predicting colors from a greyscale image (Zhang et al., 2016) or solving jigsaw puzzles (Noroozi and Favaro, 2016). Recently self-supervised pretraining in computer vision started focusing on contrastive learning which aims to pull closer representations of augmentations of the same image (positive examples) while pushing them further apart from augmentations of other images (negative examples) (Oord et al., 2018). This approach was incorporated by MoCo (He et al., 2020) which stores a dynamic dictionary of negative examples updated with a momentum encoder. SimCLR (Chen et al., 2020a) was the first algorithm that surpassed 75% top-1 accuracy on ImageNet among self-supervised methods and heavily relied on image augmentations. However, contrastive approaches present a number of downsides. For example, in order to learn meaningful representations, they require negative examples which are difficult to distinguish from the positive example (Robinson et al., 2021). Finding such examples is non-trivial, so usually the networks use large batch sizes (Chen et al., 2020a) (since negative examples are drawn from images in the same batch) in hope that at least some negative examples will be hard. Moreover, randomly sampling negative examples leads to long training. At that point, using negative examples seemed absolutely necessary because without them the model would collapse to meaningless solutions (for instance, a constant representation independent of the input image would achieve the perfect loss). This assumption was refuted by BYOL (Grill et al., 2020) which beat previous approaches without using negative examples and allowed to avoid many problems related to contrastive learning such as large batches. BYOL introduced a novel objective which aims to minimize the difference in representations of two networks. The online network is trained with stochastic gradient descent while the target network is an exponential moving average of the online network over epochs. For each image, we sample two transformations (one for each network) and pass the transformed images to both networks. In order to reach the objective and make the two representations similar, the online network has to learn features invariant to data augmentations. DINO (Caron et al., 2021) used a similar design with student (i.e. online) and teacher (i.e. target) networks, but introduced sharpening and centering layers which together effectively avoid model collapse. Furthermore, they employ a multi-crop strategy that incentivizes the network to learn local-global relations (Caron et al., 2020).

As discussed in Chapter 1, histopathology is a perfect application for self-supervised learning. Usually the pretrained models are either further finetuned or used as a fixed feature extractor. Fine-

tuning includes adding a classification head to the model and training with a standard supervised loss on a custom dataset. This option is possible for segmentation tasks where pixel-wise annotations are available (Doyle et al., 2022; Chen et al., 2020c; Graham et al., 2020). However, when only WSI-level labels are present, the model weights cannot be further trained with supervised learning, so we extract features from a number of patches corresponding to a whole slide image and then feed those in a small neural network which predicts the WSI-level label (Schirris et al., 2021; Dehaene et al., 2020). Most of the literature directly applied self-supervised pretraining methods to histopathology datasets (Ciga et al., 2022, 2021a; Truong et al., 2021; Sowrirajan et al., 2021), while several works implemented domain-specific ideas. Yang et al. (2021) used a stained separation method (Vahadane et al., 2016) to obtain channels for Hematoxylin and Eosin. In the first stage, they train an encoder-decoder architecture that predicts one channel from another. In the second stage, they extract image representations with the trained encoders and continue training in a contrastive manner similar to BYOL (Grill et al., 2020). Usually in contrastive learning, transformations of the same image are considered positive examples while tranformations of any other images are negative examples. Gildenblat and Klaiman (2019) relaxed this approach and assumed neighbouring patches on a whole slide image to be positive examples (as they often contain cells of the same type) while patches further apart were deemed negative. Xie et al. (2020) focused on nuclei segmentation. They treated two crops of equal size from the same image as positive examples while smaller crops were classified as negative as they contain less nuclei and the nuclei size is larger.

In summary, current self-supervised approaches heavily rely on image augmentations that make the encoder invariant to aspects of the image which are irrelevant for downstream tasks. These augmentations require careful tuning to be effective. Additionally, a lot of negative examples are necessary to learn useful representations. We mitigate those issues by using group equivariant networks which are invariant to rotations by design and can potentially require fewer negative examples due to their data-efficiency.

3.2 Group equivariant networks

Most of the recent work on equivariant networks is based on the group convolution networks (GCNNs) proposed by (Cohen and Welling, 2016a) who introduced this framework and implemented two relatively restrictive groups: the group $p4$ (equivalent to the cyclic group C_4 introduced in Chapter 2) which consists of translations and rotations by 90 degrees and the group $p4m$ (equivalent to D_4) which includes translations, rotations by 90 degrees and reflections. While rotating a kernel by 90 degrees is trivial, using more fine-grained rotations requires some sort of interpolation. Bekkers et al. (2018) use bi-linear interpolation to transform the "base kernel" into N rotated kernels using a fixed set of interpolation matrices where only the base kernel is trainable. On the other hand, Weiler et al. (2018) solve the same problem with steerable equivariant networks (Cohen and Welling, 2016b) which define kernels as a linear combination of kernel basis and avoid interpolation artifacts. Later on, Weiler and Cesa (2019) proposed a general, analytical solution of the kernel constraint of steerable CNNs that enables even stronger parameter sharing than regular CNNs.

Several works in histopathology evaluated rotation-equivariant networks in nuclei segmentation (Chidester et al., 2019; Chen et al., 2020c; Graham et al., 2020) and tissue classification (Lafarge et al., 2021; Veeling et al., 2018; Graham et al., 2020; Linmans et al., 2018; Yan et al., 2021). While for the former task results on multiple datasets were presented, PCam seems to be the only classification dataset used for evaluation. Veeling et al. (2018) applied GCNNs to histopathology images (using only 90 degrees rotations and reflections) on PCam (binary tissue classification). Lafarge et al. (2021) evaluated B-spline CNNs (Bekkers, 2019) on PCam and two more tasks (mitosis detection and nuclei segmentation) and experimented with more rotations (1, 4, 8 and 16). Lafarge et al. (2021) and Veeling et al. (2018) qualitatively compared the rotation-invariance of standard and equivariant CNNs.

To the best of our knowledge, so far group equivariant networks have been only used for randomly initialized models and their significance for pretrained models remains unexplored which motivates our experiments. Similarly, the extent of rotation-invariance of models is unknown thus we propose a new quantitative metric for that purpose. Finally, we use 3 more classification datasets to establish if the benefits of equivariant models hold for other classification datasets than PCam.

Chapter 4

Approach

Our work combines self-supervised learning with group equivariant networks, therefore in the upcoming sections we discuss specific techniques that we employ from these areas. Our approach uses SSL-pretraining with MoCo as explained in Section 4.1, which we train with various architectures, namely a regular encoder and a steerable equivariant network as described in Section 4.2. In Section 4.3 we look into the implementation of the latter and highlight differences between the two architectures. Finally, we elaborate on the Mean Rotation Error which is our proposed metric that measures rotation-invariance (Section 4.4).

4.1 Momentum encoder (MoCo)

Here we discuss the specific self-supervised method used for pretraining the network. This part corresponds to Step 1 shown in Figure 5.2b. We considered using 3 strong SSL techniques with comparable results on ImageNet (Russakovsky et al., 2015): SimCLR (Chen et al., 2020a), MoCov2 (Chen et al., 2020b) and DINO (Caron et al., 2021). We selected MoCov2 as it has empirically proved to have the smallest memory requirements (which is important as group equivariant networks require more memory than standard CNNs) and thanks to its dynamic dictionary its performance is less dependent on the batch size compared to SimCLR. To be precise, we use MoCov2 (Chen et al., 2020b) which includes minor engineering improvements into the MoCo (Chen et al., 2020b) framework. These enhancements were first proposed by Chen et al. (2020a) and have showcased increased performance. The first change is adding Gaussian blur into the set of image augmentations used in pretraining and the second change alters the projection head by replacing a single feed forward layer with two linear layers intertwined with ReLU. Below we present the foundations proposed in the original MoCo paper (He et al., 2020).

As discussed in Section 3.1, it has been shown that larger numbers of negative examples lead to higher performance. Previous work, given an image, uses all the other images from the same batch as negative examples due to the simplicity of this approach and its strong empirical performance (Chen et al., 2020a). However, this makes the number of negatives directly tied to the batch size and using large batches has various caveats. MoCo solves this issue by introducing a dictionary that stores image vectors from multiple batches. This dictionary is dynamically updated by adding images from the new batch and removing the oldest batch at every step, in a first-in, first-out fashion. Ideally, the keys should be created with the same or similar encoder in order to stay consistent (similar views should yield similar key vectors).

Formally, at every step, an image x^+ is sampled and perturbed with random transformations into two views v^q and v^{k+}. We obtain the query vector $q = e_q(v^q)$ from a query encoder e_q and the positive key vector $k^+ = e_k(v^{k+})$ from a key encoder e_k. The other images x^- within the batch are encoded into key vectors $k^- = e_k(v^{k-})$ and appended to the dictionary. The goal is to train the encoder e_q using back-propagation which will perform well on downstream tasks. On the one hand, updating

e_k with back-propagation is infeasible for large dictionary size as the gradients have to propagate to every view. On the other hand, a simple method of reusing e_q as e_k (without gradients) has shown to be ineffective (He et al., 2020) because the keys are encoded with encoders from different steps which leads to inconsistency. To enhance the consistency between keys, MoCo (He et al., 2020) proposes that the parameters of the key encoder should be an exponential moving average of the query encoder instead of its direct copy

$$\theta_k = m\theta_k + (m-1)\theta_q \tag{4.1}$$

where the momentum m determines how rapidly θ_k changes. For $m = 1$, e_k does not change at all and for $m = 0$, e_k equals e_q. MoCo optimizes the contrastive InfoNCE loss (Oord et al., 2018):

$$\mathcal{L}_q = -\log \frac{\exp\left(q \cdot k^+/\tau\right)}{\sum_{i=0}^{K} \exp\left(q \cdot k_i^-/\tau\right)} \tag{4.2}$$

where K is the dictionary size and τ is the temperature hyperparameter.

4.2 Steerable networks

In this work we use steerable CNNs which generalize the concept of group equivariant CNNs (Cohen and Welling, 2016a) to a more flexible framework which enables stronger parameter sharing and decreased computational costs. In order to preserve equivariance, steerable CNNs (Cohen and Welling, 2016b) introduce a kernel constraint:

$$\rho_{out}(h)\Psi = \Psi\rho_{in}(h), \quad \forall h \in H \tag{4.3}$$

which means that for a filter bank Ψ of a given layer (mentioned in Section 2.2), the linear group representations (defined in Section 2.1) of inputs ρ_{in} and the linear group representations of outputs ρ_{out} must match. Since Equation 4.3 is linear in Ψ, the space of filter banks satisfying it is a vector space. Therefore, every filter bank Ψ in this space can be decomposed into a linear combination of a basis $\psi_1^B, \ldots, \psi_n^B$ as

$$\Psi = \sum_i \alpha_i \psi_i \tag{4.4}$$

for learnable parameters α_i (Cohen and Welling, 2016b). This basis can be precomputed before training.

Feature spaces of steerable networks are defined as steerable feature fields $f : \mathbb{R}^2 \to \mathbb{R}^d$. Every steerable feature field is associated with a transformation law determined by its type ρ. ρ is a group representation (defined in Section 2.1) that specifies how the d channels are combined together (Weiler and Cesa, 2019), e.g. a trivial, regular or quotient representation.

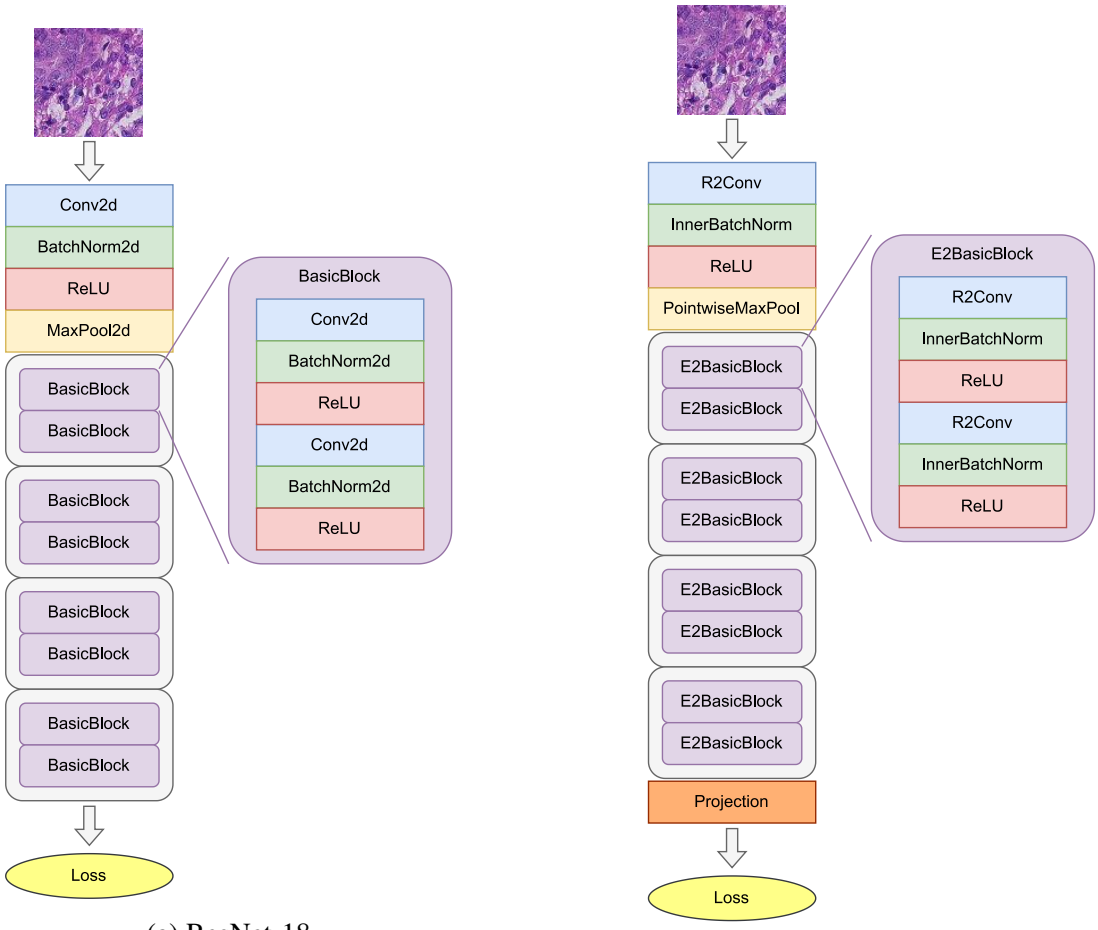
The type of feature field determines the computational and memory cost of the model. Regular feature fields are more expressive and tend to obtain the best performance. Quotient feature fields reduce the computational and memory cost, but yield inferior results due to decreased expressiveness. Regular steerable CNNs are equivalent to GCNNs (Cohen and Welling, 2016a).

The implementation of steerable networks is covered in Section 4.3.

4.3 Implementing equivariant ResNet

In this section, we discuss how to move from any CNN network to its equivariant counterpart. We leverage the *e2cnn* library (Weiler and Cesa, 2019) and provide implementation for many architectures

from the ResNet family (ResNet-18, ResNet-34, ResNet-50, ResNext-50, WideResNet-28), but the same steps apply to any other architecture. Below we use cursive for names which refer to Python libraries or classes implemented inside them.



(a) ResNet-18

(b) E2-ResNet-18

Figure 4.1: Visual comparison of standard ResNet-18 and equivariant E2-ResNet-18 models. Before model outputs and after the last BasicBlock, both models additionally contain adaptive average pooling layer and a linear layer mapping the features to class probabilities.

We visualize the standard ResNet-18 and our equivariant E2-ResNet-18 in Figure 4.1. For vanilla ResNet, we use the implementation provided in the *torchvision* library [1]. On the highest level, ResNet architecture starts with the initial convolution mapping 3 input channels to the first layer of feature maps followed by batch normalization, ReLU and max-pooling. The rest of the network consists of four large modules. Each of these modules contains multiple ResNet blocks (two for ResNet-18) which in turn are built from the most atomic parts of the network - convolution layers, batch normalization layers etc. There are two kinds of building blocks, *BasicBlock* (used in ResNet-18 and ResNet-34) and *Bottleneck* (used in ResNet-50 and larger networks). Given that we employ ResNet-18 in our experiments, we will discuss the example of *BasicBlock* but implementation of *Bottleneck* is analogous and is available in the published repository [2]. If equivariance is maintained for every layer, the whole network is equivariant (Cohen and Welling, 2016a). Therefore, we substitute every kind of layer with its equivariant equivalent provided in the *e2cnn* library. A variant of *BasicBlock* with equivariant layers becomes a *E2BasicBlock* which in turn makes up *E2ResNet*. In the upcoming sections, we discuss details of particular equivariant layers.

[1] https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py

[2] https://github.com/blazejdolicki/rissl/blob/main/models/e2_resnet.py

In conventional CNNs, every layer l has a feature space $F_l \in \mathbb{R}^{H_l \times W_l \times C_l}$ (we ignore the batch dimension for simplicity) where H_l and W_l are the height and width of the feature map and C_l is the number of channels. These channels in standard CNNs correspond to feature fields in steerable CNNs (defined in Section 2.1). The feature space of each layer is defined with the *FieldType* object which includes its G-space, the group that acts on it and the types ρ of all its feature fields.

In order to maintain the equivariance, every layer needs to satisfy Equation 4.3. The *e2cnn* library implements a *GeometricTensor* class which wraps the standard PyTorch *Tensor* together with an appropriate *FieldType*. It allows dynamic typechecking which ensures that Equation 4.3 holds for every layer and equivariance is preserved throughout the network.

4.3.1 R2Conv: Steerable convolution layer implementation

Steerable convolution layer is implemented in the *R2Conv* class. Most of its parameters are the same as for a standard convolution: kernel size, padding, stride, dilation etc. Additionally, we supply the input and output *FieldTypes*. For the very first steerable convolution, the feature space must be lifted (see Section 2.2). Therefore, in its input field type, each input representation type is trivial (Section 4.2) and the number of representations is equal to number of channels of the input image. The output field uses regular representations to perform lifting. All the other steerable convolution layers use regular representations (in our implementation) until we map features to invariance at the very end. During each training step, the bases are expanded with the learnt weights into convolution kernels (Equation 4.4) for the forward pass and the weights are updated with back-propagation. During inference, the weights do not change anymore, so the basis can be expanded only once (before the inference starts) and perform predictions without the additional overhead.

As shown by Mohamed et al. (2020), a stride larger than 1 may break equivariance due to the convolution being performed on different points in the feature map if the height and width of the map are of even size. To maintain equivariance we resize the height and width of input images such that the feature maps are odd-sized.

4.3.2 Other layers (ReLU and PointwiseMaxpool, InnerBatchNorm)

For regular representations, any point-wise non-linearity such as ReLU preserves equivariance (Weiler and Cesa, 2019), so it does not require any additional adjustments to remain equivariant. The same applies to max-pooling. Therefore, *ReLU* and *PointwiseMaxPool* implemented in e2cnn and displayed in Figure 4.1 are simply wrappers that make the corresponding PyTorch modules compatible with geometric tensors. *InnerBatchNorm* is slightly modified compared to the standard implementation of batch normalization as it computes the statistics not only over the batch and the spatial dimensions, but also over d channels within the same field (Section 2.1).

4.3.3 Projection to invariant representations

In the final layers of the network, it is necessary to project equivariant layers to final invariant representations. For early GCNNs, this could be only done via group pooling i.e. aggregating over the orientation dimension using mean or max operator (Veeling et al., 2018; Lafarge et al., 2021; Bekkers et al., 2018). Steerable CNNs allow such projection by convolving from regular to trivial representation (*conv2triv*) and that is the approach we employ as it maintains the same dimensionality of the output representation regardless of the hyperparameter N in the equivariant model.

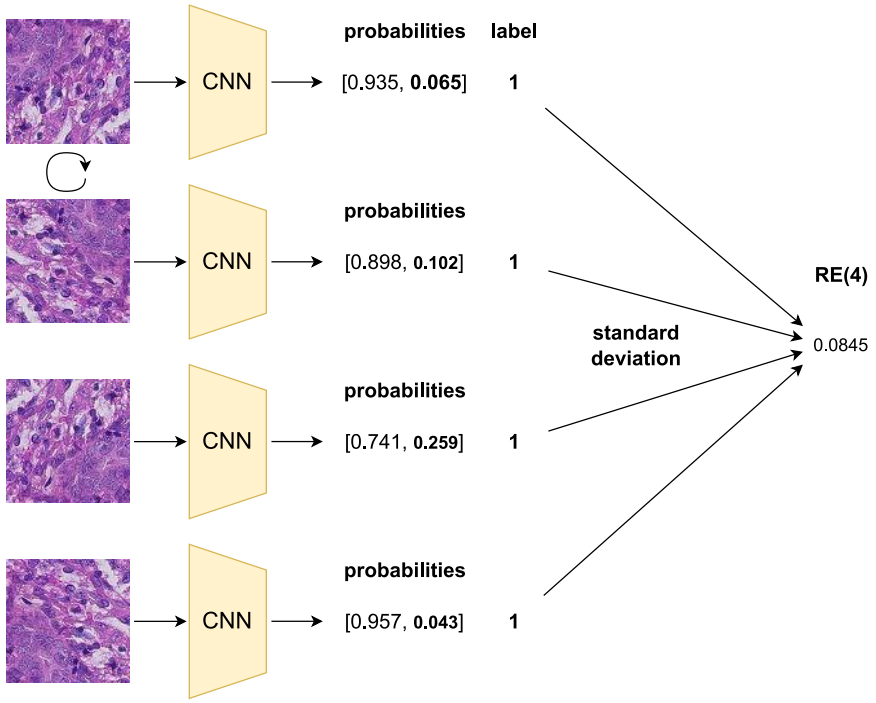


Figure 4.2: Visualized calculation of Rotation Error for a single image with $r = 4$. The average of Rotation Errors over all images in a dataset yields Mean Rotation Error.

4.4 Mean Rotation Error

Due to lack of quantitative methods for measuring rotation-invariance, we propose a new metric which we term the Mean Rotation Error (MRE) with a parameter r which determines how many rotations are sampled. The lower the MRE, the higher the model invariance. We design the metric so that it has a finite range and clear interpretation. Mathematically, it can be defined as:

$$MRE(r) = \frac{1}{N}\sum_{n=1}^{N}\sigma(p_{nr}) \quad \text{where} \quad p_{nr} = P(y = t_n|X_{nr}, \theta) \in \mathbb{R}^r \tag{4.5}$$

and it essentially is a quantitative representation of the prediction variation figures of Lafarge et al. (2021). Intuitively, for every image $x_n \in \mathbb{R}^{W\times H\times C}$ (where W is the width of the image, H is its height and C number of input channels - $C = 3$ for RGB images) in the test set, we rotate it r times with an angle of $\frac{2\pi}{r}$ to obtain r rotated images which can be represented together in a matrix $X_{nr} \in \mathbb{R}^{W\times H\times C\times r}$. We pass each rotated image through the network parametrized by θ and obtain its predicted class probabilities $P(y|X_{nr}, \theta)$. Afterwards, we take predicted probabilities of the correct class $p_{nr} = P(y = t_n|X_{nr})$ and calculate their standard deviation $\sigma(p_{nr})$. This standard deviation is the Rotation Error of a single image, $RE(r)$. Finally, we average those standard deviations for all N images in the test set to obtain $MRE(r)$. If the standard deviation $\sigma(p_{nr})$ per image is low, it means that there is little difference between predictions over orientations. By using probabilities which by definition are in range [0,1], the standard deviations (and also their mean) are also within a finite range. In the best case, when all predictions are the same for each orientation, the standard deviation is 0. In the worst case, when all predictions are at either extreme, the standard deviation is 0.5.

Figure 4.2 displays calculation of RE(4) for a single image in a binary classification task. The image is rotated 4 times (by 90 degrees) and each rotation is passed through the model to obtain predicted probabilities. Given that the correct label of the image is 1, we focus on the second probability of every rotation which yields a list [0.065, 0.102, 0.259, 0.043]. Computing standard deviations for these values gives the Rotation Error of the image equal to 0.0845. Repeating these steps for every image in the dataset and averaging, leads to the Mean Rotation Error.

Using MRE, we compare rotation-invariance of 3 models pretrained with SSL: non-equivariant network without rotation augmentations, non-equivariant network with rotation augmentations and the equivariant network.

4.4.1 Limitations

The proposed metric gives reliable results for $r = 1, 2$ or 4, but given larger r it becomes more noisy for multiple reasons. While for $r = 4$, the image can be perfectly rotated by 90 degrees, for smaller rotations (larger r) interpolation has to be employed. Moreover, the corners of the image are rotated outside of the grid while the corners of the grid become empty.

Furthermore, the importance of MRE depends on the number of possible classes in the dataset. If MRE is equal to 0.1, it shouldn't significantly matter for a dataset with 2 classes because changing the prediction by 0.1 is unlikely to change the predicted class (unless the class probabilities are both close to 0.50). However, for a datasets with 1000 classes the probability is distributed among not 2, but 1000 options, so a change of 0.1 is much more likely to affect the prediction.

Chapter 5

Experimental setup

This chapter covers our experimental design and all the details necessary to reproduce our results. It starts with a description of our main experiments in Section 5.1. The datasets used in this work are described in Secton 5.2. All training pipelines are discussed in Section 5.3. Section 5.4 mentions training details including selected hyperparameters.

5.1 Experiments overview

We perform several experiments to answer research questions from Section 1.1. We use two architectures, namely 1) the standard ResNet-18 (trained with and without rotation augmentations) and 2) its equivariant equivalent, E2-ResNet-18, without rotation augmentations. This results in 3 models present in the experiments:

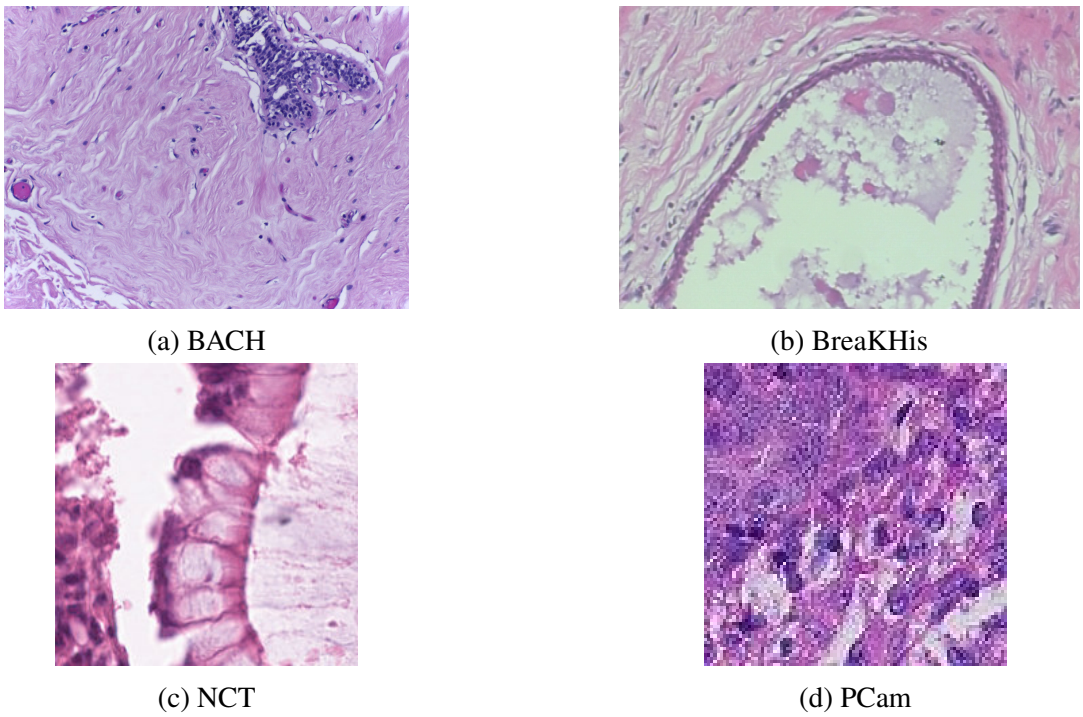
- Non-equivariant model without (w/o) rotations - standard ResNet-18 without rotations as image augmentations
- Non-equivariant model with (w/) rotations - standard ResNet-18 with rotations as image augmentations
- Equivariant model - steerable, equivariant E2-ResNet-18 without rotations as image augmentations

When stated that a model does or does not have rotation augmentations, we keep this setting consistent for all methods and training steps (random initialization, pretraining, finetuning and linear evaluation). For finetuning and linear evaluation, we always pretrain the models with MoCov2 (Chen et al., 2020b) (Section 5.4.1). We only pretrain on PCam which is the largest dataset considered in this work and perform downstream evaluation on all 4 datasets introduced in Section 5.2

For the first research question (**RQ1**), in Section 6.1, we pretrain each model with self-supervised learning and measure its rotation-invariance with Mean Rotation Error. In Section 6.2, we examine the relation between rotation-invariance and performance of models pretrained with SSL for two evaluation protocols described in Section 5.3 - linear evaluation and finetuning (**RQ2**). Additionally, we benchmark finetuning against a simple baseline - randomly initialized models. Finally, in Section 6.3 we assess if equivariant SSL encoders are more sample-efficient than standard CNNs by testing them in low-data regimes (**RQ3**).

5.2 Datasets

We use datasets with diverse properties such as sample size, tissue type, image resolution and number of classes to thoroughly evaluate our proposed method in different circumstances. Images in all of our



(a) BACH (b) BreaKHis

(c) NCT (d) PCam

Figure 5.1: Example images from all datasets.

datasets are small crops pre-extracted from giga-pixel whole slide images stained with Hematoxylin and Eosin (HE). In other words, they fall under the category of ”tile-level classification” as opposed to ”WSI-level classification” (Schirris et al., 2021).

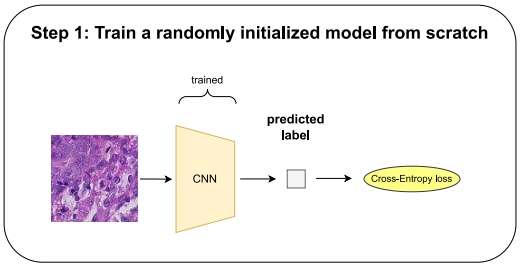
5.2.1 BACH

The BACH dataset (Aresta et al., 2019) consists of 400 images of size 2048 $\times$ 1536 pixels at 0.42 microns per pixel and concerns breast cancer. Each image is assigned one of four classes: normal, benign, in situ carcinoma and invasive carcinoma. Given the small number of images, we apply k-fold cross-validation (k=5) when tuning hyperparameters to obtain reliable results. During the challenge, the authors made available a test set with 100 images for which the labels are not provided and when the challenge took place, the test score could be obtained by submitting predictions to the leaderboard [1]. However, currently that is not possible anymore, therefore we sample 10% (40 examples) of the labelled images as our test set. The remaining 90% is first used during hyperparameter tuning with cross-validation and then during training of the final model with the selected optimal parameters (as described in Section 5.4.2).

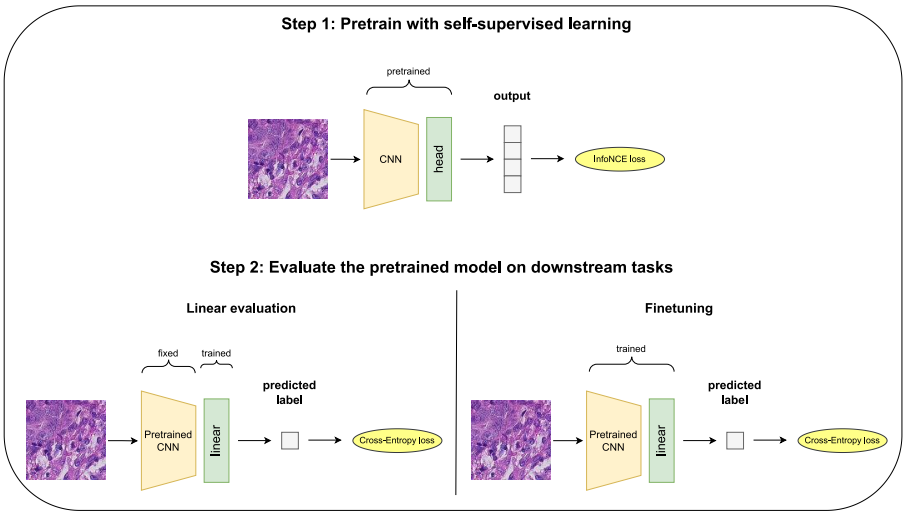
5.2.2 BreakHis

BreakHis (Spanhol et al., 2016) includes 7909 patches of size 700 $\times$ 460 pixels taken from WSIs of breast tumor tissue. The data is labelled as either benign or malignant, and images belong to one of four magnifying factors (40x, 100x, 200x, and 400x) which correspond to 0.49, 0.20, 0.10, 0.05 microns per pixel, respectively.

[1] https://iciar2018-challenge.grand-challenge.org/



(a) Training pipeline for a randomly initialized model.



(b) Training pipeline for pretraining with self-supervised learning and evaluating on downstream task.

Figure 5.2: Diagrams describing random initialization and self-supervised evaluation protocols. Zoom in for better image resolution.

5.2.3 NCT-CRC-HE-100K

The NCT-CRC-HE-100K dataset (Kather et al., 2018) contains 107,180 patches of size 224 × 224 pixels of colorectal cancer and healthy tissue, scanned at 0.5 microns per pixel. Following Ciga et al. (2021b), we use 75000 patches in the training set, 25000 in validation, and 7180 in the test set. Each image exlusively has one of nine classes: Adipose (ADI), background (BACK), debris (DEB), lymphocytes (LYM), mucus (MUC), smooth muscle (MUS), normal colon mucosa (NORM), cancer-associated stroma (STR), and colorectal adenocarcinoma epithelium (TUM).

5.2.4 PCam

PCam (Veeling et al., 2018) is derived from the Camelyon16 challenge (Ehteshami Bejnordi et al., 2017). It consists of 327,680 color images (train - 262,144, validation - 32,768, test - 32,768) with a binary label indicating presence of metastatic cancer tissue. The images are relatively small (96 x 96 px) at 0.243 microns per pixel. A positive label indicates that the center 32x32 px region of a patch contains at least one pixel of tumor tissue. Tumour tissue in the outer region does not influence the label. That might introduce noise because there may be tumour tissue in a negatively labelled image.

5.3 Training pipelines

Figure 5.2 summarizes the training pipelines of random initialization and self-supervised pretraining together with its evaluation protocols. We provide further details below.

5.3.1 Random initialization (supervised training from scratch)

The baseline performance in our experiments is obtained with a randomly initialized network which aims to predict the correct class label and to optimize with a standard Cross-Entropy loss (Figure 5.2a).

5.3.2 Self-supervised pretraining

The other setup consists of two steps: pretraining and evaluation on downstream tasks. During pretraining (Step 1 in Figure 5.2b), the model consists of two main parts - the backbone (yellow trapezoid) and the projection head (green rectangle). The backbone is the major part of the network that encodes images into image representations. After pretraining it is reused in evaluation protocols (Step 2). Throughout this work we also refer to it as the encoder or feature extractor. The projection head is a small neural networks on top of the backbone and for MoCov2 it consists of 2 linear layers with a ReLu in between them. It is only used during the pretraining steps and discarded before Step 2. During the forward pass, the model returns a vector output which is passed to the contrastive InfoNCE loss. We perform pretraining with MoCo implemented with the *vissl* library (Goyal et al., 2021). Once pretraining is finished, we test the pretrained model on downstream datasets using two evaluation protocols - linear evaluationa and finetuning.

5.3.3 Linear evaluation

In the case of linear evaluation we use a frozen (i.e. parameters are not updated) feature extractor (pretrained in Step 1) and only train the parameters of a randomly initialized linear classification layer ($f_\phi \in \mathbb{R}^{H \times C}$, where H is the hidden feature dimension and C is the number of classes). This is presented on the left of Step 2 in Figure 5.2b. Therefore, the results in this protocol are very dependent on the quality of the pretrained model. If the pretrained model is poor, a learnable linear layer on top of it will not be able to fix it. This evaluation method indicates the inherent quality of the pretrained model and suggests how the model will perform on tasks that require a feature extractor such as WSI-level multiple instance learning (Schirris et al., 2021; Dehaene et al., 2020).

5.3.4 Finetuning

On the other hand, finetuning concerns initializing the model with pretrained weights and training the whole model end-to-end including the pretrained parameters (on the right of Step 2 in Figure 5.2b). In this case, the impact of the pretrained model is lower than in linear evaluation, as it merely serves as a better starting point for training compared to random initialization. Even if the pretrained trunk does not provide useful image representations, the architecture is expressive enough to converge to high performance. Finetuning often outperforms training from scratch, especially in low-data regimes (El-Nouby et al., 2021).

5.4 Training details

We proceed to the training details of the experiments where we motivate our choice of the network architecture and report hyperparameters used in every part of the pipeline to enable reproducibility of our study.

		Datasets			
Equivariant	**Hyperparameter**	**BACH**	**BreaKHis**	**NCT**	**Pcam**
✗	Learning rate	0.0001	0.0001	0.001	0.001
	Optimizer	Adam	Adam	Adam	Adam
	Weight decay	0.00001	0.0001	0.0001	0.0001
✓	Learning rate	0.001	0.01	0.001	0.01
	Optimizer	Adam	Adam	Adam	Adam
	Weight decay	0.000001	0.00001	0.0001	0.00001

Table 5.1: Optimal hyperparameters for randomly initialized ResNet-18 and E2-ResNet-18.

5.4.1 Model selection

In order to use group equivariant networks, the network choice is limited to CNN-based architectures (as opposed to recently popularized Transformer-based models). Following previous work we use standard ResNets (He et al., 2016). We perform preliminary experiments to select the size of the model from ResNet-18, ResNet-50 and ResNext-50 (Xie et al., 2017) based on GPU memory requirements, training time and performance. The former was especially important in this work, as group equivariant networks have higher memory requirements than standard CNNs. First, we compared the accuracy of the three architectures on PCam averaged over 5 different sets of hyperparameters with equivariant models trained from scratch. ResNet-18 outperformed the other alternatives while it required twice less time to train. Furthermore, during pretraining the peak memory consumption for ResNet-18 and ResNet-50 was respectively 8 and 25.4 GBs. Given these results, we decide to use ResNet-18 in our experiments and compare it to its equivariant counterpart, E2-ResNet-18. We ensure the parameters of ResNet-18 and E2-ReNet-18 are approximately matched by multiplying the number of representations by square root of the group order N ((Weiler and Cesa, 2019). To verify the benefit of rotation image augmentations, we evaluate ResNet-18 without and with such transformations.

5.4.2 Hyperparameters

In this section we explain the design of hyperparameter tuning and the final hyperparameters selected for all setups. Hyperparameter tuning and pretraining were only performed on a single seed to limit computation time while all results in Chapter 6 are averaged over 3 seeds (7, 187, 389).

Equivariant network

We reuse equivariant-specific hyperparameters from (Weiler and Cesa, 2019) which remain the same for all runs. The order of the cyclic group N is 4, the Gaussian radial profiles of the steerable basis are of width $\sigma = 0.45$ and the frequency cutoff F is 1.0. Our E2-ResNet-18 is not equivariant to reflections.

Random initialization

For large datasets (PCam and NCT), we use batch size of 512 to decrease training time (following (Veeling et al., 2018)) while for smaller datasets the batch size is 64. We train the model from scratch for 100 epochs with early stopping if the validation accuracy doesn't improve after 20 epochs. For PCam, we divide the initial learning rate by 10 after the 40^{th} and 80^{th} epoch (following (Veeling et al., 2018)). For other datasets, the learning rate is automatically divided by 10 when there is no improvement in validation accuracy for 10 epochs. Unless stated otherwise in the sections below, optimal parameters for training from scratch are reused for finetuning and linear evaluation.

We find optimal learning rate, weight decay and optimizer via hyperparameter tuning for every dataset both for ResNet and E2ResNet. We select optimal hyperparameters based on accuracy on validation set. We leverage grid search with possible learning rates equal to 0.1, 0.01, 0.001 or 0.0001, weight decays 0.0001, 0.00001 or 0.000001 and either Adam (Kingma and Ba, 2014), or SGD (Robbins and Monro, 1951) optimizers. Each tuning run is trained for 50 epochs.

Pretraining

Given the small image size (96x96) of the pretraining dataset (PCam), we resize the images to a standard size of 224x224 during pretraining (during finetuning and linear evaluation, the images maintain the original size to make the comparison with training from scratch more fair). Furthermore, we use image augmentations following Chen et al. (2020b): color jitter with brightness, contrast and saturation equal to 0.4, hue of 0.1 and probability of 0.8, conversion to grayscale with probability of 0.2, Gaussian blur with probability 0.5 and radius ranging from 0.1 to 2.0, random horizontal flip with probabiliy of 0.5 and a discrete rotation by 90 degrees (depending on the model). The batch size equal to 256. Following (Chen et al., 2020b), we optimize with SGD with Nesterov momentum (Nesterov, 1983) of 0.9 and weight decay 0.0001. We train for 200 epochs, the initial learning rate is 0.03 and is divided by 10 after the 120^{th} and 160^{th} epochs.

Finetuning

The optimal hyperparameters are reused for finetuning, except for learning rates for Breakhis and BACH. In case of those datasets, we observed rapid training and validation loss divergence, therefore we additionally tuned the learning rate (over the following values: 0.1, 0.01, 0.001, 0.0001, 0.00001 and 0.000001) separately for each of the 3 models mentioned in Section 5.1 (non-equivariant network without and with rotation augmentations and equivariant network). The optimal learning rate during finetuning was 0.0001 for all models for BreaKHis and BACH.

Linear evaluation

For linear evaluation, we sweep over learning rates, starting with the learning rate used by (Chen et al., 2020b) and decreasing by one order of magnitude (30, 1, 0.1, 0.01, 0.001) because the training dynamics of this protocol are vastly different than training the model end-to-end. This was performed only on non-equivariant model with rotations and the optimal learning rate was then used on all models. The hyperparameter tuning was performed 10 epochs to save computational time. The optimal learning rates were 0.1 for BACH and 1.0 for all other datasets. All experiments with linear evaluation ran for 100 epochs and the learning rate was divided by 10 after the 30^{th} and 60^{th} epochs.

5.4.3 Cross-validation

For BreaKHis, NCT and PCam we used single training-validation-test splits while for BACH we performed five-fold cross-validation during hyperparameter tuning due to its particularly small size (400 examples). We utilize 90% of the dataset for cross-validation while 10% is the test set. For every fold a new model is trained and evaluated on a different validation set. We select hyperparameters with the best average accuracy over folds. Once the optimal hyperparameters are found, we train the final model on all data used during cross-validation (360 examples) and evaluate on the test set (40 examples). In that step, we don't use early stopping (as there is no validation set anymore), but instead train for the average number of epochs when the model obtained top accuracy during cross-validation.

Chapter 6

Results

In this section, we report results of the experiments described in Section 5.1 to address the research questions from Section 1.1. These results are later interpreted in Chapter 7 to identify key findings and relate them to previous literature.

6.1 Empirical rotation invariance

In order to answer the first research question, we quantitatively measure rotation-invariance of pretrained models using Mean Rotation Error (Section 4.4) for 4 and 16 sampled rotations. We use linear evaluation which is more correlated with the quality of pretrained representations than finetuning (Section 5.3). Let us recall that intuitively MRE(r) indicates how much the predicted probability of the correct class changes between r rotations of the same image.

Table 6.1 shows that MRE(4) for non-equivariant models ranges from 0.036 to 0.084. Regarding the effect of rotation augmentations, ResNet-18 without rotation augmentations consistently achieves slightly higher MRE(4) than the same model with augmentations for all datasets. Equivariant networks obtain near-perfect equivariance with MRE(4) below 0.0001 in all cases.

When the discretized number of rotations N (a hyperparameter of the equivariant model) is lower than r, the equivariance can be only approximated. To examine the quality of this approximation we also evaluate MRE for $r = 16$ (while in our equivariant model $N = 4$). In that case, the MRE considerably increases to values ranging from 0.107 to 0.271. NCT stands out as the only dataset for which the metric is above 0.2 for all three models. For all datasets, E2-ResNet obtains similar

Method/Dataset [# classes]	BACH [4]	BreaKHis [2]	NCT [9]	PCam [2]
MRE(4)				
Non-equivariant	0.056	0.058	0.058	0.084
Non-equivariant^R	0.036	0.038	0.033	0.069
Equivariant	0.00003	0.00004	0.00002	0.00007
MRE(16)				
Non-equivariant	0.109	0.108	0.271	0.174
Non-equivariant^R	0.153	0.107	0.244	0.155
Equivariant	0.147	0.113	0.251	0.140

Table 6.1: Comparison of mean rotation error on test set. Models are pretrained with self-supervised learning and evaluated with linear classification. Networks denoted with ^{R} use rotation augmentations. We show results averaged over 3 random seeds. We add number of classes corresponding to each dataset in square brackets as it helps to interpret the importance of MRE as mentioned in Section 4.4.

MRE(16) to ResNet but it remains unclear how much this is affected by the limitations of MRE for $r > 4$ described in Section 4.4.

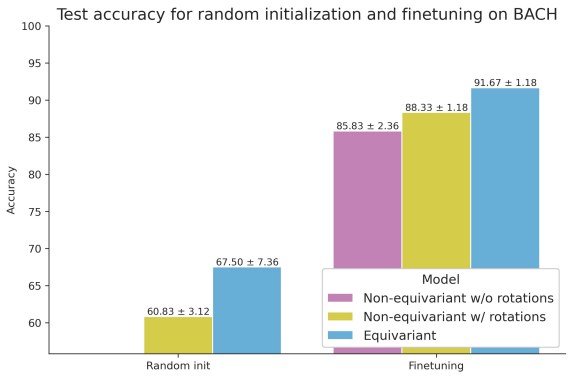
6.2 Model performance

6.2.1 Random initialization vs finetuning

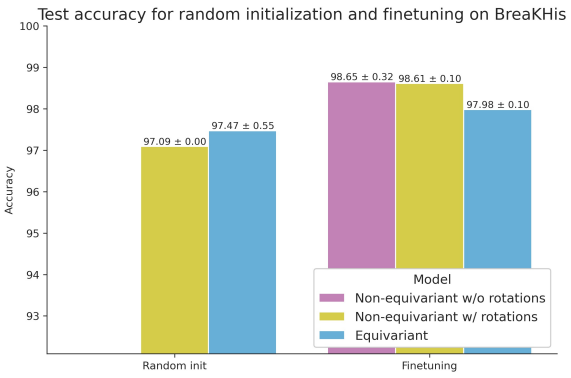
Next, we compare the performance between equivariant and non-equivariant models (with and without rotation augmentations) which were either randomly initialized, or finetuned on a downstream task after pretraining with SSL. Previous work (Veeling et al., 2018; Lafarge et al., 2021), showed that adding rotation augmentations for randomly initialized models tends to increase performance, therefore we refrain from training a model without rotations for this setup. The results are summarized in Table 6.2 and visually displayed in Figure 6.1 for each dataset. This section partially

Method/Dataset	BACH	BreaKHis	NCT	PCam
Non-equivariant - randomly initialized^R	60.83 ± 3.12	97.09 ± 0.00	91.30 ± 1.60	87.44 ± 0.89
Equivariant - randomly initialized	67.50 ± 7.36	97.47 ± 0.55	90.19 ± 0.41	**90.16** ± 0.69
Non-equivariant - finetuning ^{R}	85.83 ± 2.36	**98.65** ± 0.32	91.17 ± 0.34	86.69 ± 0.82
Non-equivariant - finetuning	88.33 ± 1.18	98.61 ± 0.10	90.62 ± 0.92	86.28 ± 0.89
Equivariant - finetuning (RISSL)	**91.67** ± 1.18	97.97 ± 0.10	**93.24** ± 0.26	89.64 ± 0.77

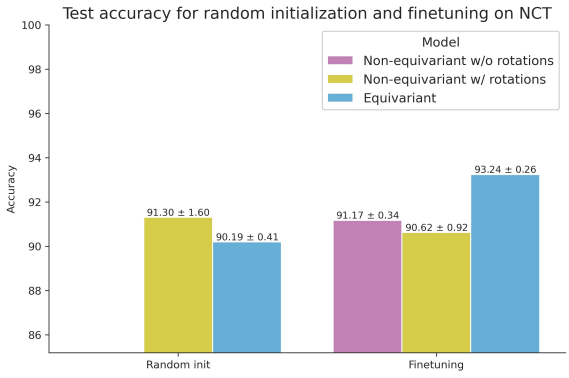
Table 6.2: Test accuracy for randomly initialized and finetuned models on 4 datasets. Models with ^{R} use rotation augmentations. We show results averaged over 3 random seeds accompanied by standard deviation.



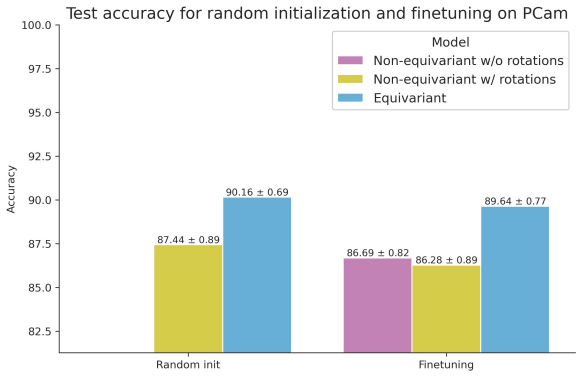
(a) BACH



(b) BreaKHis



(c) NCT



(d) PCam

Figure 6.1: Test accuracy for random initialization and finetuning of non-equivariant and equivariant models on each dataset.

answers the second research question (**RQ2**) about the connection between rotation invariance and performance in self-supervised learning. Table 6.2 shows that for 3 out of 4 datasets the finetuning accuracy decreases after adding image rotations (from 86.65 to 86.61, from 91.17 to 90.62 and from 86.69 to 86.28 for BreaKHis, NCT and PCam respectively). Furthermore, 3 datasets (BACH, BreaKHis and PCam) benefit from equivariant architecture when randomly initialized and also 3 of them (BACH, NCT, PCam) show the same pattern after finetuning. Finally, finetuning the equivariant model (RISSL) reaches the highest performance among all methods for 2 datasets (BACH and NCT).

Randomly initialized models demonstrate especially low scores on BACH, but E2-ResNet yields substantial improvement in accuracy from 60.83 to 67.50. Pretraining with self-supervised learning considerably boosts it for all models, with RISSL reaching performance of 91.67.

For BreaKHis dataset, equivariant networks yield similar or worse results than standard networks both in training from scratch and finetuning. On the other hand, finetuning a pretrained model improves the performance for each model (from 97.09 to 98.65 and from 97.47 to 97.97).

We have found NCT to be a relatively easy task with validation accuracy reaching 99%, but with the biggest gap between the validation and test accuracy among all four datasets. In this scenario, self-supervised pretraining combined with the equivariant model outperforms all other setups on the test set with accuracy of 93.24.

PCam displays clear advantage of steerable CNNs in contrast with standard CNNs in random initialization (from 87.44 to 90.16) and finetuning (from 86.69 to 89.63). On the other hand, every randomly initialized model is superior to its finetuned equivalent.

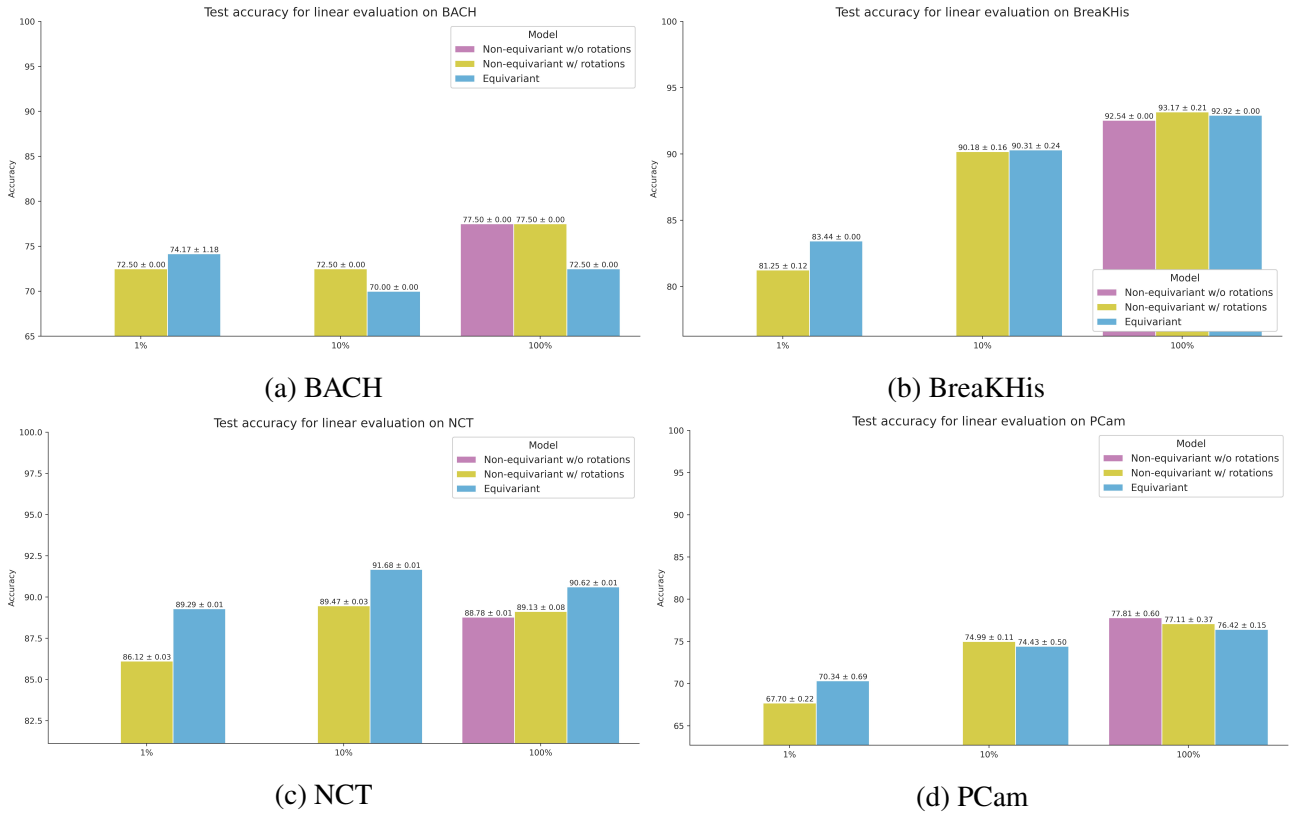
6.2.2 Linear evaluation

Linear evaluation is not directly comparable with random initialization or finetuning for which the model is trained end-to-end, therefore we consider it in a separate section. Instead it is a better indication of the inherent quality of the pretrained representations and can give a better idea of how the model would perform as a fixed feature extractor. First, we discuss performance for this protocol when using 100% of pretraining data and in the next section we focus on low data regimes. These results complete the answer to **RQ2** and are displayed in Table 6.3 and Figure 6.2.

Apart from NCT, the performance on all datasets noticibly drops in linear evaluation compared to random initialization or finetuning. Adding rotation augmentations to ResNet-18 slightly helps on 2 datasets (from 92.54 to 93.17 on BreaKHis and from 88.78 to 89.13 on NCT), it decreases the accuracy on PCam (from 77.81 to 77.11) and regardless of rotation augmentations, the score of both non-equivariant models on BACH is 77.5. E2-ResNet-18 consistently struggles with this protocol (again with the exception of NCT) displaying especially steep drop in BACH by 5.00 and modest decline for BreaKHis and PCam. NCT remains to be the outlier with the steerable CNNs surpassing standard CNNs by 1.49.

Data regime	Method	BACH	BreaKHis	NCT	PCam
1%	Non-equivariant - linear^R	72.50 ± 0.00	81.25 ± 0.12	86.12 ± 0.03	67.70 ± 0.22
	Equivariant - linear	**74.17** ± 1.18	**83.44** ± 0.00	**89.29** ± 0.01	**70.34** ± 0.69
10%	Non-equivariant - linear^R	**72.50** ± 0.00	90.18 ± 0.16	89.47 ± 0.03	**74.99** ± 0.11
	Equivariant - linear	70.00 ± 0.00	**90.31** ± 0.24	**91.68** ± 0.01	74.43 ± 0.50
100%	Non-equivariant - linear	**77.50** ± 0.00	92.54 ± 0.00	88.78 ± 0.01	**77.81** ± 0.60
	Non-equivariant - linear^R	**77.50** ± 0.00	**93.17** ± 0.21	89.13 ± 0.08	77.11 ± 0.37
	Equivariant - linear	72.50 ± 0.00	92.92 ± 0.00	**90.62** ± 0.01	76.42 ± 0.15

Table 6.3: Linear evaluation for models pretrained on 1%, 10% and 100% of pretraining data. Best results per dataset and data regime are in bold.



(a) BACH

(b) BreaKHis

(c) NCT

(d) PCam

Figure 6.2: Test accuracy for linear evaluation of non-equivariant and equivariant models on each dataset.

6.3 Self-supervised encoders in low data regimes

It has been shown that equivariant networks are more sample-efficient (i.e. they achieve high performance with fewer data samples) than standard CNNs when training the model from scratch (Lafarge et al., 2021). We aim to verify if this holds true for encoders pretrained with self-supervision (**RQ3**) by pretraining on 1%, 10% and 100% of the data (which correspond to 2621, 26214 and 262144 data samples respectively) and training a linear layer on 100% of downstream datasets. To ensure that the differences are solely caused by dataset size, we maintain the same number of pretraining iterations for each regime.

For 10% of pretraining data, we observe similar trends as for encoders trained in Section 6.2.2. The differences between ResNet-18 and E2-ResNet-18 are negligble for BreaKHis (90.18 vs 90.31) and PCam (74.99 vs 74.43). NCT displays a large boost for the equivariant network from accuracy of 89.47 to 91.68 (which interestingly is higher than performance reached with random initialization) and for BACH the situation is reversed - the score falls from 72.5 to 70.00. Conversely, encoders pretrained with 1% of the data noticeably benefit from equivariant architecture in all downstream tasks, the accuracy increases by 1.67, 2.19, 3.17 and 2.64 for BACH, BreaKHis, NCT and PCam respectively.

Chapter 7

Discussion

Here we reflect on the results obtained in Chapter 6, discuss key findings and address research questions from Section 1.1. Based on the new insights, we point out the limitations of this work and future directions that can push forward the current understanding of self-supervised learning and equivariant networks.

The first research question (**RQ1**) investigates whether rotations as image augmentations applied to standard CNNs empirically improve rotation-invariance in self-supervised learning and if their invariance is comparable to equivariant networks. **The experiments demonstrate that rotation augmentations consistently increase rotation-invariance for all datasets, but they are still far from matching the invariance of equivariant networks**. The latter obtain near-perfect MRE(4) which is expected given that robustness to rotations is incorporated into their architecture by design. Interestingly, even non-equivariant models without rotation augmentations achieve relatively low MRE(4) (on average 0.064) which can be loosely interpreted that usually the predicted probability changes by less than 0.06. Such difference is unlikely to change prediction of the model especially for binary classification. This finding stands in opposition to previous works (Veeling et al., 2018; Lafarge et al., 2021) which claim that predictions of standard CNNs (in histopathology) are very sensitive to rotations. Since equivariant networks are only explicitly invariant to rotations by $\frac{2\pi}{N}$, the differences in MRE(16) diminish between the equivariant network (where $N = 4$) and the other two models. This is partly caused by limitations of MRE for larger r such as interpolation (considered in Section 4.4).

Next research question (**RQ2**) verifies the connection between robustness to rotations and performance. **By comparing the Mean Rotation Error (Table 6.1) and the accuracy for linear evaluation and finetuning (Table 6.2 and 6.3), we find that there is practically no correlation between rotation invariance and performance** given our experimental setup and hyperparameters. Finetuning results show that for most of the datasets added rotation augmentations are not only unhelpful, but even deteriorate the performance which is particularly surprising as it displays an opposite trend to results of randomly initialized models in prior literature (Veeling et al., 2018; Lafarge et al., 2021). On the other hand, all datasets apart from BreaKHis benefit from equivariant networks during finetuning. Linear evaluation provides even more decisive evidence. While there ResNet-18 with rotations is more competitive against ResNet-18 without them, E2-ResNet-18 predominantly reaches lower performance. Especially the latter outcome feels baffling, as we expected equivariant networks to yield a better feature extractor due to their previously reported data-efficiency (Lafarge et al., 2021) and rapid convergence (Weiler and Cesa, 2019). We consider two possible reasons for this phenomenon.

Firstly, equivariant networks have different training dynamics compared to non-equivariant networks. As explained in Section 5.4.2, reusing hyperparameters from random initialization in finetuning can lead to loss divergence, but after finding a more suitable learning rate, E2-ResNet-18 outperforms ResNet-18. This suggests that similarly tuning the hyperparameters specifically for linear evaluation can yield better results.

Secondly, as mentioned in Section 5.3, finetuning may alleviate the shortcomings of a poorly

pretrained encoder. Conversely, linear evaluation does not possess enough expressiveness to considerably improve the model after pretraining, it can only linearly combine features of the self-supervised image representations. Combining this knowledge with strong results in finetuning and poor accuracy in linear evaluation leads us to a hypothesis that the pretraining phase of the equivariant network was suboptimal and there is a potential for improvement given more careful tuning and model selection. Selecting the optimal pretrained model over epochs remains an open problem in self-supervised learning, therefore we simply pretrain all models for 200 epochs and use the final checkpoint. Given that equivariant networks have been shown to converge faster than non-equivariant models when randomly initialized (Weiler and Cesa, 2019), it is possible that the same happens during pretraining and therefore E2-ResNet reaches optimal performance earlier and after 200 epochs it is more overfit than standard ResNet (which translates into worse downstream accuracy in linear evaluation).

Additionally, let us focus on particular datasets in the effort to identify key factors determining the usefulness of self-supervised pretraining and equivariant architectures. BACH shows the largest gains among all datasets both from equivariant architectures and self-supervised pretraining. This gives clear evidence that equivariant networks should be preferred for small datasets and confirms previous findings about data-efficiency (Lafarge et al., 2021). Although large leaps in accuracy after pretraining are common in cases of data scarcity, the fact that this trend holds despite pretraining the model on a dataset with vastly different image resolution and task highlights strong transferability of self-supervised learning. BreaKHis appears to profit from equivariance the least among all datasets. A possible reason for this outcome can be hypothesized after looking at the training loss over epochs. All models trained on this dataset eventually converge to accuracy larger than 99%. We observe that the loss saturates much faster for equivariant networks which, in turn, leads to infinitesimal gradients and thus learning effectively stops after a few initial epochs. For standard CNNs, the training takes longer which permits more exploration of the loss surface. This suggests that equivariant network are unnecessary for easy tasks that can be almost perfectly learnt by simpler models. Conversely, equivariant finetuning yields the best results for NCT which seems to be an easy task, but with a large disparity between validation and test set. Possibly, explicit invariance to rotations in this case incentivizes learning more generalizable features. Finally, PCam obtains high performance of equivariant model in random initialization and finetuning. However, it appears that the large size of the dataset renders pretraining redundant as the scores of respective models are higher when trained from scratch.

Finally, we consider self-supervised pretraining in low-data regimes (**RQ3**). **Our results with self-supervised encoders pretrained on varying samples of data appear to be inconclusive and further research is necessary.** While using the full data have shown poor performance of E2-ResNet-18, encoders trained on 10% of PCam are quite similar. Conversely, equivariant models self-supervised with the smallest fraction of data are consistently superior than non-equivariant architectures. Experimenting with pretraining on other regimes should shed more light on the topic.

This work has explored the potential benefits of utilizing equivariant architectures in self-supervised learning, but there remain multiple limitations and valuable directions to investigate. As aforementioned, there is a high likelihood that a better strategy for pretrained model selection might yield further gains from self-supervised learning. Other benefits could stem from more elaborate tuning of linear evaluation, as we only sweep over multiple learning rates for ResNet-18 with rotations and select the learning rate with best performance after 10 epochs. Moreover, while state-of-the-art models are pretrained on millions or even billions of images from various sources, we use a dataset with 300,000 images, which are relatively homogeneous, in order to limit training time and required computational budget. Validating our findings with a larger dataset would be a valuable next step. Furthermore, we used $N = 4$ for all equivariant models. Experimenting with larger N could lead to further performance gains as shown by Lafarge et al. (2021).

Chapter 8

Conclusions

In this study, we leverage the inherent rotation-invariance of histopathological images in order to develop better self-supervised models specifically tailored to this domain. Our experiments are conducted on 4 tile-level classification datasets with diverse sample size, image resolutions and tasks. We empirically show that while simple rotation augmentations improve robustness to rotations, they can often degrade the downstream performance of models pretrained with self-supervised learning. Therefore, we advice researchers to avoid them in self-supervised learning. Group equivariant architectures achieve almost perfect rotation invariance and appear to be useful during finetuning in particular for challenging tasks with few data samples which are common in medical applications, but on datasets that are easy to optimize we recommend resorting to standard convolutional networks as the gains are diminishing. Experiments with linear evaluation indicate that equivariant encoders do not outperform non-equivariant feature extractors and their sample-efficiency during pretraining phase requires further experiments, thus the advantage of equivariant encoders in tasks requiring fixed feature extractors (such as WSI-level classification) currently appears questionable. We publish our code and pretrained models in hope to foster future research in self-supervised learning and equivariant networks for digital pathology.

Bibliography

Guilherme Aresta, Teresa Araújo, Scotty Kwok, Sai Saketh Chennamsetty, Mohammed Safwan, Varghese Alex, Bahram Marami, Marcel Prastawa, Monica Chan, Michael Donovan, Gerardo Fernandez, Jack Zeineh, Matthias Kohl, Christoph Walz, Florian Ludwig, Stefan Braunewell, Maximilian Baust, Quoc Dang Vu, Minh Nguyen Nhat To, Eal Kim, Jin Tae Kwak, Sameh Galal, Veronica Sanchez-Freire, Nadia Brancati, Maria Frucci, Daniel Riccio, Yaqi Wang, Lingling Sun, Kaiqiang Ma, Jiannan Fang, Ismael Kone, Lahsen Boulmane, Aurélio Campilho, Catarina Eloy, António Polónia, and Paulo Aguiar. 2019. Bach: Grand challenge on breast cancer histology images. *Medical Image Analysis*, 56:122–139.

Erik J. Bekkers. 2019. B-spline cnns on lie groups. *CoRR*, abs/1909.12057.

Erik J Bekkers, Maxime W Lafarge, Mitko Veta, Koen AJ Eppenhof, Josien PW Pluim, and Remco Duits. 2018. Roto-translation covariant convolutional networks for medical image analysis. In *International conference on medical image computing and computer-assisted intervention*, pages 440–448. Springer.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. 2020. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020a. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709.

Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. 2020b. Improved baselines with momentum contrastive learning. *CoRR*, abs/2003.04297.

Yiqi Chen, Xuanya Li, Kai Hu, Zhineng Chen, and Xieping Gao. 2020c. Nuclei segmentation in histopathology images using rotation equivariant and multi-level feature aggregation neural network. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 549–554. IEEE.

Benjamin Chidester, That-Vinh Ton, Minh-Triet Tran, Jian Ma, and Minh N Do. 2019. Enhanced rotation-equivariant u-net for nuclear segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0.

Ozan Ciga, Tony Xu, and Anne L Martel. 2021a. Resource and data efficient self supervised learning. *arXiv preprint arXiv:2109.01721*.

Ozan Ciga, Tony Xu, and Anne L. Martel. 2021b. Self supervised contrastive learning for digital histopathology.

Ozan Ciga, Tony Xu, and Anne Louise Martel. 2022. Self supervised contrastive learning for digital histopathology. *Machine Learning with Applications*, 7:100198.

Taco Cohen and Max Welling. 2016a. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR.

Taco S. Cohen and Max Welling. 2016b. Steerable cnns. *CoRR*, abs/1612.08498.

Olivier Dehaene, Axel Camara, Olivier Moindrot, Axel de Lavergne, and Pierre Courtiol. 2020. Self-supervision closes the gap between weak and strong supervision in histology. *arXiv preprint arXiv:2012.03583*.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR.

Shannon Doyle, Francesco Dal Canton, Jelle Wesseling, Clara I Sánchez, and Jonas Teuwen. 2022. Mammary duct detection using self-supervised encoders. In *Medical Imaging 2022: Computer-Aided Diagnosis*, volume 12033, pages 215–218. SPIE.

Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes van Diest, Bram van Ginneken, Nico Karssemeijer, Geert Litjens, Jeroen A. W. M. van der Laak, , and the CAMELYON16 Consortium. 2017. Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer. *JAMA*, 318(22):2199–2210.

Alaaeldin El-Nouby, Gautier Izacard, Hugo Touvron, Ivan Laptev, Hervé Jegou, and Edouard Grave. 2021. Are large-scale datasets necessary for self-supervised pre-training? *arXiv preprint arXiv:2112.10740*.

Jacob Gildenblat and Eldad Klaiman. 2019. Self-supervised similarity learning for digital pathology. *arXiv preprint arXiv:1905.08139*.

Priya Goyal, Quentin Duval, Jeremy Reizenstein, Matthew Leavitt, Min Xu, Benjamin Lefaudeux, Mannat Singh, Vinicius Reis, Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Ishan Misra. 2021. Vissl. `https://github.com/facebookresearch/vissl`.

Priya Goyal, Quentin Duval, Isaac Seessel, Mathilde Caron, Mannat Singh, Ishan Misra, Levent Sagun, Armand Joulin, and Piotr Bojanowski. 2022. Vision models are more robust and fair when pretrained on uncurated images without supervision. *arXiv preprint arXiv:2202.08360*.

Simon Graham, David Epstein, and Nasir Rajpoot. 2020. Dense steerable filter cnns for exploiting rotational symmetry in histology images. *IEEE Transactions on Medical Imaging*, 39(12):4124–4136.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Jakob Nikolas Kather, Niels Halama, and Alexander Marx. 2018. 100,000 histological images of human colorectal cancer and healthy tissue. *URL: https://doi. org/10.5281/zenodo*, 1214456.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Maxime W Lafarge, Erik J Bekkers, Josien P W Pluim, Remco Duits, and Mitko Veta. 2021. Roto-translation equivariant convolutional networks: Application to histopathology image analysis. *Medical Image Analysis*, 68:101849.

Jasper Linmans, Jim Winkens, Bastiaan S Veeling, Taco S Cohen, and Max Welling. 2018. Sample efficient semantic segmentation using rotation equivariant convolutional networks. *arXiv preprint arXiv:1807.00583*.

Mirgahney Mohamed, Gabriele Cesa, Taco S Cohen, and Max Welling. 2020. A data and compute efficient design for limited-resources deep learning. *arXiv preprint arXiv:2004.09691*.

Yurii E Nesterov. 1983. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547.

Mehdi Noroozi and Paolo Favaro. 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer.

Chris Olah, Nick Cammarata, Chelsea Voss, Ludwig Schubert, and Gabriel Goh. 2020. Naturally occurring equivariance in neural networks. *Distill*. Https://distill.pub/2020/circuits/equivariance.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Joshua Robinson, Li Sun, Ke Yu, Kayhan Batmanghelich, Stefanie Jegelka, and Suvrit Sra. 2021. Can contrastive learning avoid shortcut solutions? *Advances in neural information processing systems*, 34:4974–4986.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Yoni Schirris, Efstratios Gavves, Iris Nederlof, Hugo Mark Horlings, and Jonas Teuwen. 2021. Deepsmile: Self-supervised heterogeneity-aware multiple instance learning for dna damage response defect classification directly from h&e whole-slide images. *arXiv preprint arXiv:2107.09405*.

Jean-Pierre Serre et al. 1977. *Linear representations of finite groups*, volume 42. Springer.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813.

Hari Sowrirajan, Jingbo Yang, Andrew Y Ng, and Pranav Rajpurkar. 2021. Moco pretraining improves representation and transferability of chest x-ray models. In *Medical Imaging with Deep Learning*, pages 728–744. PMLR.

Fabio Alexandre Spanhol, Luiz S Oliveira, Caroline Petitjean, and Laurent Heutte. 2016. Breast cancer histopathological image classification using convolutional neural networks. In *2016 international joint conference on neural networks (IJCNN)*, pages 2560–2567. IEEE.

Tuan Truong, Sadegh Mohammadi, and Matthias Lenga. 2021. How transferable are self-supervised features in medical image classification tasks? In *Proceedings of Machine Learning for Health*, volume 158 of *Proceedings of Machine Learning Research*, pages 54–74. PMLR.

Abhishek Vahadane, Tingying Peng, Amit Sethi, Shadi Albarqouni, Lichao Wang, Maximilian Baust, Katja Steiger, Anna Melissa Schlitter, Irene Esposito, and Nassir Navab. 2016. Structure-preserving color normalization and sparse stain separation for histological images. *IEEE transactions on medical imaging*, 35(8):1962–1971.

Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. 2018. Rotation equivariant CNNs for digital pathology.

Maurice Weiler and Gabriele Cesa. 2019. General e(2)-equivariant steerable cnns. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Maurice Weiler, Fred A Hamprecht, and Martin Storath. 2018. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Xinpeng Xie, Jiawei Chen, Yuexiang Li, Linlin Shen, Kai Ma, and Yefeng Zheng. 2020. Instance-aware self-supervised learning for nuclei segmentation. In *International conference on medical image computing and computer-assisted intervention*, pages 341–350. Springer.

Xiankun Yan, Jianrui Ding, and Heng-Da Cheng. 2021. A novel adaptive fuzzy deep learning approach for histopathologic cancer detection. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 3518–3521. IEEE.

Pengshuai Yang, Zhiwei Hong, Xiaoxu Yin, Chengzhan Zhu, and Rui Jiang. 2021. Self-supervised visual representation learning for histopathological images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 47–57. Springer.

Richard Zhang, Phillip Isola, and Alexei A Efros. 2016. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer.