Journal of Pathology Informatics 14 (2023) 100305

Contents lists available at ScienceDirect

Journal of Pathology Informatics

journal homepage: www.elsevier.com/locate/jpi

# Investigation of semi- and self-supervised learning methods in the histopathological domain

Benjamin Voigt [a,b,*,1], Oliver Fischer [a,b], Bruno Schilling [b], Christian Krumnow [b], Christian Herta [b]

[a] Charité - Universitätsmedizin Berlin, Corporate Member of Freie Universität Berlin and Humboldt-Universität zu Berlin, Charitéplatz 1, 10117 Berlin, Germany
[b] University of Applied Sciences (HTW) Berlin, Center for Biomedical Image and Information Processing, Ostendstraße 25, 12459 Berlin, Germany

ARTICLE INFO

*Keywords:*
Neural networks
Deep learning
Self-supervised learning
Semi-supervised learning
Computational pathology
Tissue analysis

ABSTRACT

Training models with semi- or self-supervised learning methods is one way to reduce annotation effort since they rely on unlabeled or sparsely labeled datasets. Such approaches are particularly promising for domains with a time-consuming annotation process requiring specialized expertise and where high-quality labeled machine learning datasets are scarce, like in computational pathology. Even though some of these methods have been used in the histopathological domain, there is, so far, no comprehensive study comparing different approaches. Therefore, this work compares feature extractors models trained with state-of-the-art semi- or self-supervised learning methods PAWS, SimCLR, and SimSiam within a unified framework. We show that such models, across different architectures and network configurations, have a positive performance impact on histopathological classification tasks, even in low data regimes. Moreover, our observations suggest that features learned from a particular dataset, i.e., tissue type, are only in-domain transferable to a certain extent. Finally, we share our experience using each method in computational pathology and provide recommendations for its use.

## Introduction

Supervised learning is still the dominant paradigm for building machine learning applications. A large, diverse, and qualitatively annotated dataset is needed for a robust and adequate prediction performance of classifiers trained with such a paradigm. For natural images, such datasets are publicly available[1] and have significantly contributed to the success of deep learning.[2] Models trained on such datasets are publicly available for standard deep learning frameworks as part of their corresponding model zoo. Those models typically consist of a feature extractor and an additional classification head, where the latter is usually highly trained on the specific task, and the former can be transferred to and reused for other applications in different data domains.

A problem with such a transfer approach arises when a certain proportion of the learned features are not helpful for the classification task, and on the other hand, necessary nuances in the data are not faithfully differentiated. Usually, this is the case in the histopathological domain, where the images differ significantly, e.g., from natural images.[3] Therefore, the need for appropriate datasets in the domain to apply the supervised learning approaches is significant. Unfortunately, publicly available labeled histopathological datasets suitable for machine learning barely exist due to the required expertise for annotations and legal regulations that restrict the use of such data. Furthermore, the few existing datasets are either small or specialized for one particular task. Consequently, one of these datasets cannot represent the tremendous variety of tissue types, tissue characteristics, and changes due to different diseases.

In general, a remedy to overcome the need for a diverse large-scale labeled dataset is using unlabeled data to improve the prediction (classification or regression) performance. Thus, different approaches utilizing semi- or self-supervised learning paradigms have been developed to improve the model performance on various downstream tasks. Some of these approaches have recently been transferred to or optimized for the field of histopathology (Section Related Work).

This work explores the effectiveness of such learning paradigms aside from supervised learning in computational pathology. For this, we compare the performance of feature extractors resulting from 3 different state-of-the-art pretraining methods on different histopathological classification tasks. For a more comprehensive study, we chose methods with opposing learning paradigms, i.e., PAWS[4] as a semi-supervised learning method, SimCLR[5] as a contrastive self-supervised learning method (SSL), and SimSiam[6] as a non-contrastive self-supervised learning method (NCSSL). In Section Models and pretraining protocol, we provide a unified review of the details of each method from which their similarities and differences become evident.

* Corresponding author.
*E-mail address:* benjamin.voigt@htw-berlin.de (B. Voigt).
[1] *URL:* https://github.com/bensnajdar/histopathology-ssl.

http://dx.doi.org/10.1016/j.jpi.2023.100305
Received 12 January 2023; Received in revised form 3 March 2023; Accepted 3 March 2023
Available online 12 March 2023

We trained feature extractors using publicly available histopathological datasets with each method and evaluated them on various in-domain classification tasks.[7–9] To allow for comparability of the different methods, we hereby aligned the corresponding training protocols as much as possible. We especially use a unified finetuning protocol for all 3 methods that allows comparing the impact of the different pretraining methods directly. Furthermore, we explored how applicable the methods were to a new data domain and investigated them with different data augmentation strategies suited for histopathological data. We examined the performance difference between the feature extractor, finetuned models, and models trained from scratch in a standard supervised fashion on different amounts of labeled data to study the benefit of pretraining in such settings. Furthermore, we analyzed how the representations of the feature extractors transfer to additional in-domain data, i.e., other downstream tasks.

Our results show that differences in the performance of the different learning paradigms under the same conditions are observable.[2] The results indicate that pretraining with any method on in-domain data seems beneficial. However, we found that SimCLR yields the most stable performance, while SimSiam has the best overall results. Surprisingly, despite explicitly using label information, PAWS showed the weakest performance in our experiments.

## Related work

Due to the challenges associated with creating a usable labeled dataset for machine learning, early efforts were made to use fewer or no labels, i.e., semi-supervised learning or unsupervised learning. In particular, established unsupervised learning methods, like autoencoder, random forest, clustering, transfer learning, or newer ones, e.g., generative adversarial networks, have been studied in the histopathological domain for some time. McAlpine et al[10] reviews some of these methods and their application. However, in this study, we investigate a learning paradigm that recently came into focus: self-supervised learning.[11]

**Self-supervised learning** is well suited for computer vision tasks with large neural networks. It divides into 2 classes contrastive self-supervised learning method (SSL) and non-contrastive self-supervised learning method (NCSSL).

SSL generates representations (views) of samples by using data augmentation techniques. Representations of the same samples denote positive pairs and representations of different inputs as negative pairs. In this context, these representations are typically mapped via a CNN, also called a feature extractor, into an embedding space. A contrastive loss function minimizes the distance between embeddings of positive representations while the distance to embeddings of negative representations is maximized. Widespread implementations of this concept are CPC,[12] SimCLR,[5] and MoCo.[13] These approaches are already present in different studies on histopathological tasks and indicate a benefit if a feature extractor is pretrained within the data domain. Saillard et al[14] trained feature extractors using MoCo on a public cancer dataset[15] to show that these consistently outperform their counterparts pretrained using ImageNet. Baykaner et al[16] presented their DIME pipeline to evaluate pretrained CNNs using SimCLR as self-supervised learning methods on an extensive, diverse whole slide images (WSI) dataset created from The Cancer Genome Atlas (TCGA). They investigated the feature embedding space of these models in a detailed graphical fashion using UMAP visualizations, McInnes and Healy.[17] Lu et al[18] combined CPC with multiple instance learning to classify breast cancer on the publicly available BACH dataset.[19] Additional papers, which differ mainly in using different datasets for pretraining or other downstream tasks also show the benefit.[20–22]

NCSSL methods utilize pretraining without the use of negative examples. Possible implementations usually fulfill domain-agnostic or domain-specific subtasks to train a feature extractor. An agnostic task can be any transformation that does not require expert or domain knowledge. For example, an image could be tiled and shuffled. The task would be to arrange the tiles in the correct order to create the original or the identification of different rotations of a sample. A histopathological-specific task could be to order different magnification levels of a WSI patch. Such self-supervised strategies are used as pretraining by Koohbanani et al,[23] and Srinidhi et al.[24] Another class of NCSSL methods attempts to learn embedding space mappings, such that different augmented views of an image are mapped closely together. BYOL[25] uses 2 distinct networks to create such representations, while SimSiam utilizes Siamese Networks.[26] These methods have also been used for pathological tasks.[27–29] Furthermore, SimTriplet is another implementation designed explicitly for histopathological data. This approach emerged as a further development of SimSiam using the multi-view nature of histopathological WSIs. In addition to 2 augmented views from 1 patch, SimTriplet crops a second patch within the spatial neighborhood of the first one from the WSI, which is included in the similarity measure. Since tissue is assumed to be locally similar, SimTriplet adds additional positive pair information, resulting in better embedding space clustering.

So far, only a few works have analyzed self-supervised learning methods for histopathological downstream tasks in greater detail, e.g. Ciga et al.[30] They pretrained multiple CNN architectures on 57 fully unlabeled histopathology datasets drawn from TCGA using SimCLR and tested the performance on different downstream classification and regression tasks. In addition, they investigated the transferability of feature extractors between different tissue types, analyzed the impact of augmentation stacks and patch resolution on pretraining, and the impact of dataset size on performance in the downstream task. We serve as a complementary study to their work and expand this by evaluating additional methods.

**Semi-supervised learning** approaches allow the usage of unlabeled data combined with a small amount of labeled data. This technique is particularly suitable for the medical field, where insufficient resources with the necessary expertise and a lack of time lead to a bottleneck in annotating data. Methods, which implement the semi-supervised learning paradigm, are numerous. We limit ourselves here to give some examples of the application of this paradigm in the context of artificial neural networks and histopathological data.

Wang et al[31] created a semi-supervised learning pipeline for nuclei detection. The approach reconstructs images from detection maps created from unlabeled data. The spatial consistency between the original image and the reconstruction is used as a regularizing effect in the actual training of the detection network. In addition, several works exist utilizing unlabeled and labeled data to train generative adversarial networks (GANs) to solve the stain normalization problem, i.e., a color shift between different digitalized tissues caused by the staining and scanning process.[32–34] Finally, an interesting approach was recently published by Liu et al.[35] They used a small amount of labeled training data to stabilize the pseudo-label prediction of an unsupervised trained feature extractor. Instead of using a few data points with classical direct labels, it is also possible to use so-called weak-labels corresponding to assigning a high-level label for multiple unlabeled data (e.g., images) belonging to one category; see, for instance Sikaroudi et al.[36]

Besides these application examples, we utilized a recent development method named PAWS,[4] which shows exceptional classification performance on ImageNet with a drastically reduced amount of labeled data. However, to the authors' knowledge, PAWS has been no application in the medical domain, especially in the analysis of histopathological images.

## Models, material, and methods

This section presents the datasets and the 2 protocols used for training models in the pretraining and finetuning. Both protocols combined form our full experimental pipeline.

In this regard, the pretraining protocol specifies our environment to train models with self- and semi-supervised methods. Such models consist of an encoder that maps (encodes) the input into a high-dimensional space and a method-specific head. The encoder part is further trained

[2] Experiment protocols and implementation of the study can be found in the GitHub repository https://github.com/bensnajdar/histopathology-ssl.

using supervised learning on a so-called downstream task, i.e., a specific histopathological problem. The finetune protocol unifies the settings for such downstream training over all experiments and enables us to evaluate the performance of an encoder as a feature extractor. It also ensures the comparability of the learned encoders despite different self- and semi-supervised methods used in the pretraining.

In addition, the self- and semi-supervised methods and their necessary adaptations, which we applied, are briefly reviewed.

### *Datasets*

All data used are publicly available. We selected 2 datasets with different tissues, classes, and histopathological tasks for pretraining: Kather and PCam. The Lizard dataset was used exclusively to explore and evaluate the resulting encoders. In the following, we provide a brief introduction to all datasets. Additionally, Table 1 shows the training dataset and split sizes used in this work.

**PCam**: The PatchCamelyon is extracted from histopathological slides of lymph node sections provided by the Camelyon16 Challenge.[37] The original hematoxylin and eosin (H&E) stained tissue is digitized using a 40x objective resulting in a pixel resolution of 0.243 microns. This process was performed in 2 different laboratories using different scanners, resulting in color differences in the images, i.e., a potential distributional shift in the data. PCams patch-based dataset is sampled from the WSIs, keeping the initial split into train and test sets. Validation images are drawn from 20% of the training set. It contains 262.144 examples for the training set and 32.768 for each validation and test set. Images are 96x96 pixels and increased to 10x magnification by undersampling. Therefore, the resolution of the images changes to about 0.972 microns/pixel. Patches containing mostly background are filtered out in the sampling process. The resulting publicly available dataset is a balanced binary classification dataset with the positive class indicating the presence of metastatic tissue in the center of an image.

**NCT-CRC-HE-100K**: After its creator, this dataset is often referred to as **Kather**. The training set consists of 100.000 non-overlapping images of $224 \times 224$ pixels scanned at 0.5 μm/pixel spatial resolution from H&E stained histological images of human colorectal cancer and normal tissue. It is divided into 9, approximately balanced, tissue 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). In addition, the dataset is available in an authentic and stain-normalized variant. We used only the latter. One-tenth of the training images were split-off for validation purposes. As a test set, the co-published CRC-HE-7K dataset was utilized as recommended by Kather et al.[8] The 7180 images are also from patients with colorectal adenocarcinoma but do not overlap with the training data. Their resolution is the same at about 0.5 microns/pixel.

**Lizard**: The Lizard dataset is a large-scale histopathological dataset, for instance, segmentation by Graham et al.[9] Images are 224 × 224 pixels with 20x magnification. Originally, the dataset is designed for instance segmentation. Based on the publicly available dataset, we created an object classification task by cropping the images such that the label-defining object is positioned in the image center. In addition, we applied zero padding to increase the patch size, if necessary, back to the initial resolution.[3] The unfiltered dataset includes 6 classes: neutrophil, epithelial, lymphocyte, plasma, eosinophil, and connective tissue. In preprocessing, we removed the neutrophil (4116 samples) and eosinophil (2979 samples) classes to balance the dataset since these 2 classes were vastly underrepresented. Our train, validation, and test sets are created from distinct images and contain 29 7245, 64 901, and 62 672 samples, respectively. The resolution of the patches is about 1.167 microns/pixel.

**Table 1**
Total and average annotated samples per class for each split size used in the downstream tasks, given in %. Data points were uniformly sampled for each split.

| Data | Split in % | Samples | Avg. samples per class |
|---|---|---|---|
| Kather | 100 | 90 000 | 10 000 |
| | 8 | 7200 | 800 |
| | 0.8 | 720 | 80 |
| | 0.2 | 180 | 20 |
| PCam | 100 | 262 144 | 131 072 |
| | 8 | 20 971 | 10 485 |
| | 0.8 | 2097 | 1048 |
| | 0.2 | 524 | 262 |
| Lizard | 100 | 297 245 | 74 311 |
| | 8 | 23 780 | 5945 |
| | 0.8 | 2378 | 594 |
| | 0.2 | 594 | 149 |

### *Models and pretraining protocol*

We adapted each of the self- and semi-supervised methods to get the best performance possible in the new data domain. However, we implemented an open protocol for the model training to increase interpretability and comparability in the evaluation process between these methods.

The network architectures of each learning method can be split into an encoder $f$, also denoted as feature extractor, followed by a projection head $h$, depending on the method (Fig. 1). The ResNet family[38] is used as the base architecture for $f$. The networks' multi-layer perceptrons (MLPs) were adjusted to match the structure of the proposed method. We trained the encoder with 3 different architectures to investigate how model complexity affects the quality of the feature extractor later in the experiments. The ResNet50 is studied as a standard, as by most other works. Since the histopathological datasets are often small, ResNet18 is also used since it has fewer parameters and is built on a simpler structure, not using the bottleneck approach like more complex ResNet networks. In addition, we trained models with a Wide_ResNet28w2, which has slightly more parameters than the ResNet18 and operates with more channels in the convolution blocks than a default architecture. Therefore, in terms of complexity, this results in the following ascending order: ResNet18, Wide_ResNet28w2, and ResNet50. The projection heads $h$ used are small MLPs. Each method uses an individual structure for $h$, and we left these MLPs as proposed in the original publications.

We performed a broad hyperparameter search for each architecture–method–data triplet to select optimizer parameters, augmentation stack, and method-specific parameters to adapt the methods to the domain. The latter are fixed after generally well-working values were found. Depending on the method, the searches are performed using grid search and more advanced techniques like Adaptive-Asha.[39] We selected the final training settings (Appendix A.2), based on the feature space clustering estimated by UMAP or t-SNE projections[17,40] and model performance measured by method-specific proxy metrics described in the corresponding method sections. In all final training settings, an SGD optimizer, either with or without a LARC/LAMB optimizer, is used, and training is performed using specific random seeds used consistently for all methods, which especially ensures that all methods use the same initialization of the encoders, and thus reduces corresponding variations effect between the methods.

All models were trained with input dimensions of $96 \times 96$, corresponding to the smallest image patch size of the utilized datasets. Resulting in a change of the microns/pixel values for datasets based on an image size of 224 by a factor of 7/3. Besides image scaling, several other data augmentation techniques were applied to the images.

We investigated different augmentation stacks for each method in the pretraining process. Starting from the originally published stack, we designed modified variations of these stacks by adjusting parameters and adding transformations, which proved beneficial for the data domain in the past.[41] Appendix A.1 describes each stack in more detail.

In essence, all methods use augmentation techniques to create different views of the same input and then optimize the encoder parameters such

[3] A script to reproduce the lizard classification dataset is also included in the GitHub repository (https://github.com/bensnajdar/histopathology-ssl/blob/main/utils/create_lizard_dataset.py).

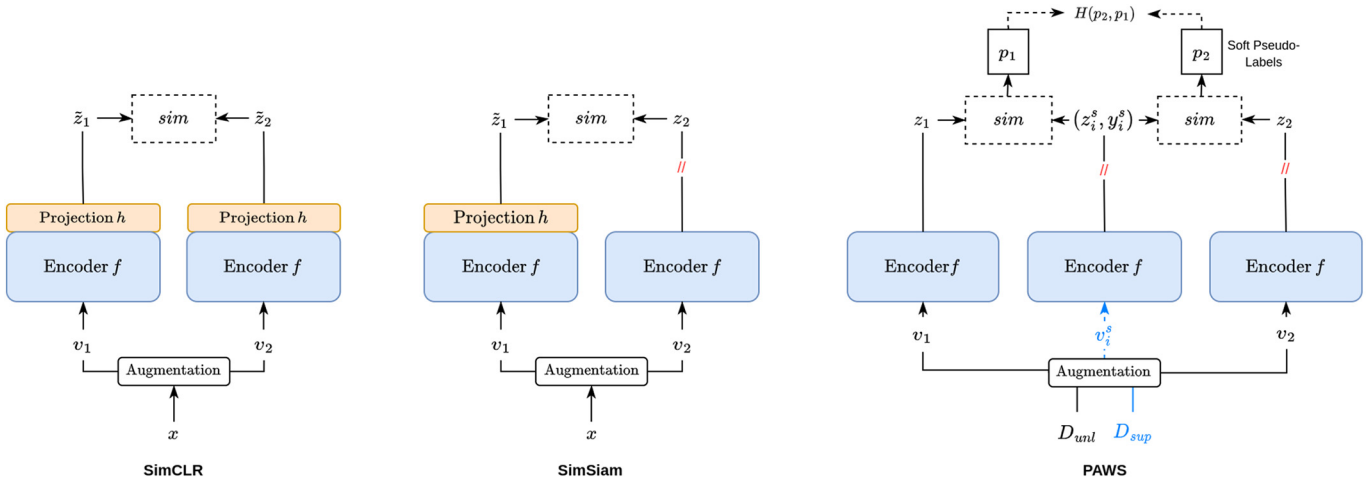



**Fig. 1.** Semi- and self-supervised methods used for the model training. Each method forwards different views of an image (or images) through a network structure based on a backbone CNN network *f*, i.e., the encoder, and an MLP projection head *h*. The resulting representations are compared by a similarity function *sim* and assessed by an individual loss function; see the related method section for details. Crossed-out paths do not contribute to the parameter update.

that those views are mapped in a meaningful way into an embedding space by either comparing them to labeled data (PAWS), maximizing their distance in embedding space from additional negative examples (SimCLR) or clustering embedding of similar inputs (SimSiam, SimCLR).

*PAWS*

PAWS is a semi-supervised learning method meaning that it utilizes partially labeled data. The training data $D$ consists of a labeled support set $D_{\text{sup}}$ and a much larger set of unlabeled data, denoted by $D_{\text{unl}}$. The basic idea of PAWS can be described in 3 major steps; see Fig. 1. First, 2 views are generated for an image drawn from $D_{\text{unl}}$ via an augmentation stack, including random operations, and mapped through the encoder network $f$ into embedding vectors. Additionally, a balanced batch of annotated support images is sampled, augmented by the same stack, and likewise encoded into embedding vectors. Finally, (soft) pseudo-labels are derived for each view by comparing the views' embeddings with the embeddings of the support set samples using a similarity function *sim*. As an optimization task, the difference between these pseudo-labels is minimized so that the encoder network becomes stable against different augmentations and learns important features by exploiting the annotated support set.

On a formal level, for an input $x \in D_{\text{unl}}$, the pair of views $(v_1, v_2)$ and the corresponding embeddings $(z_1, z_2)$ are generated.[4] Similarly for a support set batch $(x_1^{(s)}, y_1^{(s)})$, …, $(x_{n_s}^{(s)}, y_{n_s}^{(s)}) \in D_{\text{sup}}$, the embeddings $z_1^{(s)}$, …$z_{n_s}^{(s)}$ are generated where $y_i^{(s)} \in \mathbb{R}^K$ is the one-hot encoded label of $x_i^{(s)}$ with $K$ being the number of classes. The pseudo-labels $(p_1, p_2)$ for $(v_1, v_2)$ are yield by

$$p_i = \sum_{j=1}^{n_s} \frac{sim\left(z_i, z_j^{(s)}\right)}{\sum_{k=1}^{n_s} sim\left(z_i, z_k^{(s)}\right)} y_j^{(s)} \tag{1}$$

with $i = 1, 2$. We used the exponential temperature scaled cosine similarity as $sim(z_i, z_j) = \exp\left(z_i^T z_j / \|z_i\| \, \|z_j\| \, \tau\right)$ with $\tau$ denoting the temperature parameter, which is 0.1 in all experiments. Since by construction $p_1, p_2 \in [0,1]^K$ and the sum of their elements is normalized to 1, they are interpreted as a class probability distribution, providing (soft) pseudo-labels, for the views $(v_1, v_2)$. At last, $p_1, p_2$ are compared by cross-entropy $H$, which constitutes the cost function of PAWS. In practice, the 2 training batches are independently sampled from $D_{\text{unl}}$ and $D_{\text{sup}}$. To optimize the function of the supporting batch, it should be ensured that each class is equally represented in the support set.

As suggested in Assran et al,[4] we modified the CNN encoder $f$ by adding 3 linear layers, while the first 2 layers included batch normalization and ReLU as an activation function. The input and hidden dimensions of the MLP were chosen according to the CNN architecture (128 for the Wide_ResNet28w2, 2048 for ResNet50, and 512 for ResNet18), and we fixed the output dimension to 128.

We employed PAWS without methodological adaptations on the histopathological data. PAWS models were trained using a LAMB optimizer for about 10 epochs, concerning the size of $D_{unl}$. Since the approach outputs soft pseudo-labels, we could compute classification metrics for a validation set to monitor the learning process as for standard supervised learning approaches. We trained multiple models for each architecture–method–data pair exploring distinct sizes of the support set $D_{sup}$. Given a training set $D$ of size $N$, we consider support sets of size $0.08N$, $0.008N$, and $0.002N$. The smallest support set size resulted in about 20 labeled images per class for Kather, which is extremely low considering modern deep learning approaches. Table 1 provides a detailed overview of data split sizes.

*SimCLR*

SimCLR is a self-supervised learning method based on the contrastive approach. The fundamental training objective is to maximize the distance of encodings resulting from fundamentally unequal inputs. This process is controlled by minimizing the embedding similarity of views generated from the same input, while simultaneously maximizing the distance to all other views in a training batch. Again, views are generated by an augmentation stack containing random operations.

On a formal level from the unlabeled dataset $D$, a batch $x_1$, …$x_n \in D$ is drawn and for each sample $x_i$, 2 views $(v_{2i-1}, v_{2i})$ (forming a positive pair) are generated, resulting in $2n$ views $v_1$, …, $v_{2n}$. The other $2n - 2$ views act as negative (contrastive) examples for each positive pair of views. We created the views with a reduced augmentation stack as the original SimCLR paper proposed using a strong color distortion and random crop only (Appendix A.1). Each view $v_i$ is processed through an encoder $f$ and an additional projection head $h$ to create a corresponding encoding $\tilde{z}_i$, i.e., $h(f(v_i)) = \tilde{z}_i$.[5]

[4] The notation was adapted for consistency. In the original formulation the views were denoted as $v$ and $v_+$ with the corresponding embeddings $z$, $z_+$ and pseudo-labels $p$ and $p_+$.

[5] We refer to the output of the full network $\tilde{z}_i$ as encoding here as we want to distinguish it from the outputs of encoders, which we usually refer to as embedding.

The loss function is built around encoding $\tilde{z}_i$. Therefore, the loss calculates the similarity of encoding $\tilde{z}_i$ and all other encodings $\tilde{z}_j$, which result from views of different inputs, by a function *sim* and is given by

$$L = \frac{1}{2n}\sum_{k=1}^{n}[l_{2k-1,2k} + l_{2k,2k-1}]$$
$$\text{with } l_{i,j} = -\log \frac{\exp(sim(\tilde{z}_i, \tilde{z}_j)/\tau)}{\sum_{\substack{k=1 \\ k\neq i}}^{2n} \exp(sim(\tilde{z}_i, \tilde{z}_k)/\tau)}. \quad (2)$$

Here, $\tau$ denotes a temperature parameter. We fixed $\tau$ to 0.5 for all of our experiments and used the cosine similarity as *sim*. By the definition of the loss, the overall learning objective of SimCLR is to maximize the similarity between positive pairs while minimizing the similarity to all negative examples.

We implemented adaptations as suggested by Chen et al (p. 6).[42] The encoder $f$ consists of a base CNN extended by an additional linear layer with variable output dimension, batch normalization, and ReLU activation. Likewise, a deeper projection head $h$ was employed by adding 2 additional linear layers, including batch normalization and ReLU. The output dimension value of $f$ and the hidden dimensions of $h$ resulted from a hyperparameter optimization, but the output dimension of $h$ was fixed to 128.

Each SimCLR model was trained using a LAMB optimizer and a 1-cycle cosine decaying learning rate. We investigated other optimizers, but LAMB led to the best results. As large batch sizes are crucial for the SimCLR method, we trained with a batch size of 512. We utilized Adaptive-ASHA to obtain the final parameters of the network dimensions and optimizer settings. We appended an overview in Appendix A.2. A k-nearest-neighbor classifier was regularly calculated on a validation set in the embedding space during the training process. The metric did not influence the training but functioned as a proxy variable to observe the learning process.

### SimSiam

SimSiam uses the self-supervised learning paradigm implementing a non-contrastive approach, i.e., it does not repel the representations of negative pairs in the learning process.

The method implements the learning paradigm by comparing the output vectors of 2 views of the same input. Again, the views are created by an augmentation stack, including random operations. For each view, an embedding (encoder output), an encoding (output of an additional MLP), and a similarity of these 2 vectors are computed. However, the similarity is calculated between the embedding of one view and the encoding of another.

Formally, for an image $x$ from an unlabeled dataset $D$, we created a pair of views $(v_1, v_2)$. For each view $v_i$, we computed the embedding $z_i = f(v_i)$ and the encoding $\tilde{z}_i = h(f(v_i))$. In this study, the encoder $f$ consisted of a deep CNN with an additional 3-layer MLP network. The last batch-norm layer of the MLP was dropped to match the network structure of PAWS. The only difference is the output dimension of the encoder, i.e., the embedding size. The projection head $h$[6] was left unchanged, a 2-layer MLP with a bottleneck structure and a hidden dimension of one-quarter of the embedding size as recommended in Chen and He.[6]

The optimization goal is to maximize the similarity, measured by a function *sim*, between the embeddings and the encodings as represented in the loss function

$$L = -\frac{1}{2}[sim(\tilde{z}_1, z_2) + sim(z_1, \tilde{z}_2)], \quad (3)$$

The overall minus sign converts the maximization of the similarity into a minimization of the given loss function, and the *cosine similarity* was used as the similarity measure *sim*. An essential aspect of the method is that only the gradients of the encodings $\tilde{z}_i$ update the network where the embedding $z_j$ are considered constant during the update step, see Fig. 1.

In the training process, we studied the standard deviation of the $\ell_2$-normalized outputs to check if the embeddings were collapsing to a constant vector and regularly calculated the accuracy of a k-nearest-neighbor classifier as a proxy metric to monitor the progress. We employed an SGD optimizer, scaled the learning rate linearly with $lr * batch\_size/256$ according to the original implementation, and adjusted with a 1-cycle cosine decay schedule. We performed a hyperparameter optimization using grid search here instead of ASHA. The final network parameters and optimizer settings are reported in Appendix A.2.

### Finetune protocol

To increase the comparability of the pretrained model performance across the methods, we used a fixed protocol unifying the training settings of every finetune process, i.e., the downstream tasks. We recognize that, in general, this approach may result in weaker performance on the task than is individually possible for each classifier, especially if compared to published state-of-the-art results. However, disentangling and inter-comparing the effect of the pretrained models is essentially impossible if each experiment is individually optimized.

In contrast, the fixed finetune process is a reasonable basis for a fair comparison, provided that the results are compared relatively between the evaluated methods. In addition, using this approach, the focus is clearly on the pretrained model in a controlled environment. Therefore, we consider this strategy justified.

Each downstream task was trained using the following training settings and network modifications. Based on the experiment, the model's head $h$ was replaced by a reinitialized MLP head consisting of either 1 linear layer (logistic regression) or 3 blocks, each including a linear layer, a batch-norm layer, and ReLU as an activation function. The corresponding downstream task at hand defined the output dimension. The input dimension of the head was chosen according to the embedding dimension of the encoder. In the case of PAWS, where $h$ is missing, the head was attached to the encoder instead. The MLP parts of the encoders, described in the related method sections above, are usually kept if not stated otherwise. Since this approach resulted in a stronger, more complex prediction head for SimSiam and SimCLR, we conducted an ablation study to investigate this further, where the MLP part is reduced to the initially proposed one. We report the results in Appendix C.3.

To ensure the performance comparability of the semi- and self-supervised pretraining, we trained a corresponding model in a fully supervised fashion for each encoder finetuning. In this case, the entire model was randomly reinitialized using the proposed strategy of He et al.[38] The training was performed with the same settings as the finetuning of the related encoder. Due to this process, we were able to calculate a benefit for each encoder that is not biased by the slight architectural differences of the encoder and thus ensures comparability.

We studied the encoders on tiny subsets of the training data, Section Downstream task data reduction. Since some datasets contained only a few hundred samples per class in such experiments, a fixed batch size of 32 was used for every training. In the extreme case of a 0.2% subset split of Kather, 1 epoch is about 180 samples only (20 images per class), for example. The training length, the number of batches trained, was chosen to represent 20 epochs on each dataset. In addition, an encoder warmup for roughly 4 epochs was implemented, i.e., parameters of the pretrained model were not updated during that period. The models were trained using an SGD optimizer with a learning rate of 0.005, a momentum value of 0.9, and a weight decay value of 0.0001. The learning rate was adjusted with a cosine decay. We applied a small augmentation stack to the training set suited for histopathological data (Appendix A.1). Despite rescaling to match the input size and data normalization, no test augmentation was employed. As an exception to the protocol, if a training utilized the entire

[6] Notation was adapted for reasons of consistency. In the original publications of SimSiam, the mapping $h$ is denoted as a prediction head, and the modification of the encoder MLP as projection head.

dataset, we trained with a batch size of 256 instead of 32 and adjusted the total batches trained to be equal to 20 epochs.

If not stated otherwise, we used a fixed set of seeds for each experiment's method/dataset combination to ensure equally random initializations again. Model evaluations are performed on fully trained models only to achieve consistent comparability across experiments, i.e., intermediate model states were discarded even if they performed better on the validation data during training.

## Experiments

We investigated semi- and self-supervised trained models in various experimental setups to gain more insight into the benefit of pretraining in the histopathological environment. In particular, we explored the stability of the training process, which is an essential factor for the reproducibility and applicability of a classifier in the medical field. In addition, we analyzed the encoder performance under the reduction of training data on different downstream tasks and compared them to a fully supervised approach. Finally, we examined the transferability properties of the encoders learned by the different methods upon in-domain shifts.

### *Encoder initialization sensitivity*

Within this experiment, we analyzed the classifiers for possible performance fluctuations. In critical domains like medical applications of deep learning algorithms, it is essential to be consistent in the results, and also for scientific work, traceability and reproducibility are crucial aspects. We trained 5 encoders for each method on the Kather dataset using different experiment seeds to study the performance fluctuations. Seeds were randomly sampled. After, we finetuned each encoder on the Kather downstream task 5 times, reusing the same seeds, resulting in 25 runs per configuration.

By fixing the seeds, it is ensured that for the encoder training, all CNNs were initialized consistently for all methods, and thus provided a fair and comparable starting ground. Accordingly, on the downstream task, the seeds ensured an equal initialization of the prediction heads and training settings (data sampling, batch sampling, etc.) on a method level.

In this setting, we limited prediction heads to logistic regression since it provides the most information about the performance of each encoder. However, we trained each configuration with and without updating the encoder weights, i.e., freezing the encoder during finetuning. If the encoder weights were updated, we trained only the prediction head for about 4 epochs before updating the entire model weights (encoder warmup). In addition, the finetuning was performed on a fixed 8% data split set to save computational resources. In this experiment, a ResNet50 was studied as a default architecture representative.

### *Downstream task data reduction*

One purported benefit of training networks based on an encoder is to achieve a competitive performance using less annotated data compared with solely supervised training. We finetuned encoders on subsets to examine this aspect, simulating the scenario of limited annotations being available.

Besides using the entire training set, we randomly sampled 3 notably smaller subsets for every dataset. In the initial investigation of Kather, we could not observe any significant performance changes before reducing the data to 8% of its original size. Hence, we used this breakpoint to create a first subset. Going further, we limited the samples to 0.8% and 0.2% of the corresponding datasets for the other splits. The resulting number of training data and average images per class are given in Table 1. We ensured the subsets were fixed across all experiment runs to maintain comparability.

We trained PCam and Kather encoders with each method for the selected ResNet architectures on the entire datasets. Note that PAWS models have seen labeled data in the process, potentially introducing a bias if the encoder has seen more annotated samples than in finetuning. To counteract this risk, we calculated multiple encoders for PAWS with different sizes of the support set $D_{sup}$ matching the split sizes. If the finetuning is performed on an entire training dataset, we used the 8% split encoder version.

Each encoder was then finetuned on the downstream task of every dataset using the 4 training data splits with several configurations. We explored the regression head and 3-layer MLP option as a prediction head. Furthermore, we trained variants in which the encoder weights were either updated or frozen, i.e., finetuning the prediciton head only. We used the same warmup strategy as before (4.1) in the configurations where the encoder was updated. Across all datasets, network variations, and configuration settings, we trained 288 classifiers per method. Since we calculated a fully supervised complement for each encoder finetuning, the number effectively doubles.

We monitored the accuracy, the f1-score, and the expected calibration error (ECE) metric to measure the performance on the downstream tasks. In addition, we calculated a benefit for each of those metrics except the ECE. We define the benefit as the difference between the performance of a finetuned encoder and the solely supervised trained counterpart. Accordingly, the benefit reflects the absolute difference (gain or loss) in each experimental setting and allows us to compare the performance gain of different methods.

### *Histopathological features and transferability*

An essential assumption is that self-supervised methods learn underlying fundamental patterns from non-annotated data.

We investigate this assumption on a macro-level by reexamining the data generated by the experiments conducted in 4.2. Hereby, the experiment runs finetuning an encoder to classification task unrelated to its training data were significant. Hence, their performance metrics provide information about the transferability of the feature space, i.e., to what extent embeddings learned on a specific tissue can be valuable for tasks using other tissue types.

Therefore, we examined specific encoders with basic explainable AI methods to gain a deeper insight into the feature space. Grad-Cam[43] and Guided Backpropagation[44] maps were created for randomly selected images of the Kather test set. In consultation with pathologists, we evaluated maps to determine which morphological structures were relevant for each classifier. However, since we have performed this evaluation exclusively for a few examples, we include the results, even though promising, in Appendix C.2. Finally, we computed t-SNE mappings of these encoders to explore how the datasets were already clustered in feature space, also reported in the B.1.

## Results and discussion

### *General observations*

Before discussing the experimental results, we report our experience gained in applying the methods in the histopathological domain.

The PAWS approach was most sensitive to the choice of training parameters. To identify suitable parameters, a hyperparameter optimization was needed for every support set split. The ASHA algorithm, covering a larger search space than Grid Seach with equal computational resources, worked best with PAWS. In the initialization sensitivity experiment, even with optimized parameters, 2 of 5 encoder trainings collapsed by changing the random seed only, i.e., different initial weights. Thus, we had to change the random seeds twice to train all models. However, overall, we observed that PAWS adapted sufficiently to the domain. But among the tested methods, it was the most computationally intensive.

Discovering a parameter setting for the SimSiam training was initially challenging. When using sophisticated hyperparameter search algorithms such as ASHA, we observed that this approach leads to corrupted training processes regardless of the chosen thresholds. The loss drops close to the minimum within a few iterations, but the proxy metrics did not reflect any improvement. Also, monitoring the standard deviation of the normalized outputs did not help with this problem either. We observed cases

where ASHA would identify parameter settings leading to non-collapsed encoders with no meaningful representation. Thus, we conclude that deficient representation at local minima could be a general issue (even for the class of NCSSL methods) and a subject of further research.

Finally, we used Grid Search for SimSiam with a significantly narrowed search space, leading to appropriate parameter settings. However, the determined parameters were stable across the experiments regardless of changing the architecture or dataset. During our exploration of the role of the augmentation stack, we learned that it is another critical factor for learning meaningful embeddings, particularly strong color manipulations were essential.

SimCLR was the only method chosen where we have not encountered any reportable difficulties when applying it to the histopathological domain. However, large batch sizes are required for optimal performance, so the appropriate computational resources are a prerequisite.

*Encoder initialization sensitivity*

We investigated how much the weight initialization affected the encoder training and its performance. Therefore, encoders were trained and finetuned on Kather with a fixed setting but a changing random seed. This yielded a total of 25 data points for each method. The experiment was repeated without updating the encoder weights in the finetune process, i.e., finetuning the prediction head only. We report each method's mean accuracy and f1-score and their respective standard deviations in Table 2.

The results of SimCLR slightly outperform those of SimSiam in the frozen setting. The standard deviation values are consistently low regardless of metric and dataset. Indeed, comparing these methods, this is expected since the contrastive approach is a more stable learning strategy and likely produces a more distinct clustering in the feature space. However, interestingly, this performance advantage is lost when finetuning the encoder weights. As a result, the performance of SimCLR barely changes, while SimSiam improves both metrics and their standard deviations and exceeds the other methods.

In contrast, the metrics of PAWS are predominantly lower, and their standard deviations are consistently above those of the other methods. Especially in the frozen setting, the performance differences on the test data are substantial with a gap of at least 8.17% (f1-score of SimSiam) and standard deviations above 2.15. We conclude that the PAWS encoders learned much weaker embeddings, which was unexpected since it partially exploits annotated data. A probable explanation is the choice of the support set $D_{sup}$. When exploring the methods, we observed that the choice of $D_{sup}$ strongly influences the performance of PAWS. Finetuning the encoder significantly closes the performance gap and elevates the results of PAWS to the ones of SimCLR.

Next to the apparent performance differences, we also noticed that the standard deviation of, for instance, the test set accuracy ranges from 0.29% to 2.60%, hence changes by an order of magnitude when comparing the different methods. We investigated the accuracy standard deviation further to explore the origin of the corresponding values. Hence, we computed for all 5 encoders of each method, i.e., fixing the pretraining seed, the standard deviation of their 5 finetune runs. Averaging these encoder-wise standard deviations for each method gives us an estimate of the fluctuations resulting from the finetuning. Vise versa, we received the pretraining estimate by averaging the finetune-wise standard deviations obtained from the 5 distinct encoders sharing the same finetune seed. We restrict ourselves to the frozen encoder setting and report the estimates in Table 3.

**Table 3**
Summary of the mean standard deviation (std) of the resulting accuracy on the validation (val acc) and test set (test acc), computed for the varying seeds in the pretraining and finetuning where the encoder weights were not updated during finetuning.

| Method | Val acc std | | Test acc std | |
|---|---|---|---|---|
| | Finetune | Pretrain | Finetune | Pretrain |
| PAWS | 0.70% | 2.44% | 1.06% | 2.34% |
| SimCLR | 0.18% | 0.25% | 0.26% | 0.21% |
| SimSiam | 0.18% | 1.57% | 0.18% | 0.41% |

In cases with a significant standard deviation, we find that the major contribution resulted from the pretraining. Thus, genuinely different encoders resulted from the different initializations of the pretraining with the semi- or self-supervised method. In contrast, the variations resulting from the downstream task are mostly minor, which post-validates our finetuning scheme as relatively stable. The only exception is the PAWS method, where the finetuning also led to notable variations. This observation likely resulted from the weaker embeddings learned in the pretraining such that more substantial performance variations appeared in the finetuning process.

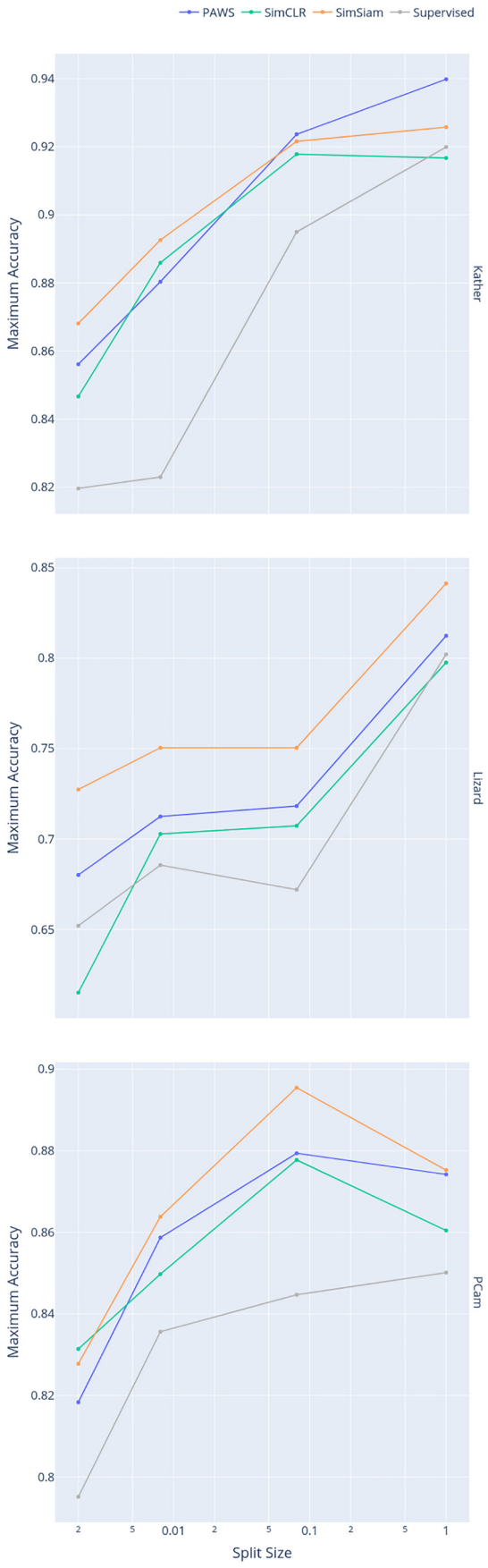
In summary, PAWS generated the weakest encoders, showing the most considerable performance fluctuations. Moreover, these fluctuations persisted even when such an encoder was finetuned in this experiment. The other methods yield a stable performance all over (SimCLR) or at least after finetuning (SimSiam). However, the findings were somewhat surprising since we expected methods that receive more information in pretraining to be more powerful. Hence, PAWS should be the most reliable approach, followed by SimCLR and SimSiam. Especially with SimSiam, we were concerned that the naive NCSSL approach would have difficulties separating the "similar-looking" histopathological data. On the contrary, it is noteworthy that the method achieved almost comparable results to currently published ones[24,30,45], even though the training conditions were not optimal due to the unified finetuning protocol and the usage of merely 8% of the training data. However, observation indicates that SimCLR is the most stable method concerning the investigated different training initializations.

*Downstream task data reduction*

We studied the performance development of the encoders depending on the size of the training dataset. Therefore, PCam and Kather encoders were trained using the proposed methods with different ResNet architectures. Each encoder was finetuned on every dataset classification task using multiple data splits (100%, 8%, 0.8%, and 0.2%) and exploring several training configurations by changing the prediction head complexity and freezing the encoder weights. Furthermore, for each encoder finetuning, a randomly initialized network counterpart was trained complementarily. Since the calculated performance metrics, f1-score and accuracy, demonstrated analogous results, we limit ourselves to discussing the accuracy here. We append all representations using the f1-score in Appendix B.2.

**Table 2**
Accuracies and f1-scores averaged over 25 results on the Kather classification task. Their standard deviation is noted in brackets. Using a logistic regression prediction head, several ResNet50 encoders were finetuned on the 8% data split. The training runs differed only by the initialized weights.

| Method | Acc (val) | Acc (test) | f1 (val) | f1 (test) |
|---|---|---|---|---|
| PAWS (freezed) | 82.80% (2.52) | 79.25% (2.60) | 81.87% (2.68) | 75.28% (2.15) |
| SimCLR (freezed) | 92.67% (0.28) | 90.45% (0.29) | 92.81% (0.28) | 87.02% (0.44) |
| SimSiam (freezed) | 85.13% (1.57) | 89.42% (0.44) | 84.20% (2.09) | 83.45% (0.97) |
| PAWS (finetuned) | 91.14% (1.67) | 90.52% (1.15) | 91.21% (1.72) | 87.23% (1.16) |
| SimCLR (finetuned) | 94.10% (0.45) | 90.57% (0.70) | 94.19% (0.45) | 87.01% (0.96) |
| SimSiam (finetuned) | 95.47% (0.27) | 91.47% (0.52) | 95.54% (0.26) | 88.53% (0.60) |




We report each method's maximum accuracies on different dataset splits in Fig. 2. Note, as we analyzed the completely finetuned networks here, each maximum was picked from a pool of 12 training configurations in which the CNN architecture, MLP complexity, and pretraining dataset were altered. Since complementary supervised runs exist for every method, such maxima were selected from 36 data points. On Kather, the performance consistently decreases if the available training data is reduced. A widening gap between the solely supervised trained models and finetuned encoders is discernible and suggests that the smaller the annotated dataset becomes, the greater the relative advantage of pretraining. The SimCLR performance on the 100% split is an exception to this general behavior, which is not reflected in the f1-score. However, both observations align with our anticipations of the experimental setup and are consistent with recently published results.[30] Overall, the methods yield minor differences in maximum accuracy using lower training splits, and none is consistently superior.

We observe this somewhat differently with the Lizard data. Simsiam performed consistently better than the other methods, while SimCLR did not consistently exceed the supervised runs. Further, an apparent widening of the performance gap between the finetuned encoders and the purely supervised runs is missing. For SimSiam and PAWS, it seems relatively constant. One likely explanation is that most encoder embeddings were irrelevant for the task or tissue since Lizard was absent in the pretraining. Note that this indicates some encoder transferability limitations, which we will investigate further in the following section. The PCam results rank between the observations for Kather and Lizard. The training data reduction aligns mainly with a performance drop but not a widening performance gap. One oddity is that the peak performance was generally achieved at a training split of 8% for each encoder finetuning, rather than for the entire dataset. This observation may indicate that the finetuning could not fully converge on the large PCam dataset within the finetune protocol. Here, too, SimSiam predominantly outperforms the other methods slightly.

By comparing the maximum accuracies in Fig. 2, we simulated a real-world case of running multiple experiments with various architectures and selecting the best performing. However, we compared data resulting from multiple configurations and thus discarded information about performance differences of encoders in single experiment runs. To investigate this further, we calculated the accuracy benefit for the same data as used before, i.e., runs with finetuned encoder weights. As a reminder, we defined the benefit as the performance difference of a specific metric between a finetuned encoder and its purely supervised trained counterpart. Hence, the benefit estimates the gain or loss expected through an encoder. We summarize the accuracy benefits in Fig. 3.

A prominent observation from the benefit evaluation is that SimSiam outperforms the other methods, especially on Kather and Lizard, where the benefit disparity was significant. It was the only method close to purely positive benefits. Yet, on Kather, each method had a considerable benefit, and, like before, we observed the expected behavior, i.e., an increasing benefit with decreasing training data, only here. In contrast, substantial positive benefits were also evident on Lizard, but not following the same pattern and also with apparent method differences. For example, the smallest median of SimSiam was close to 0.1, while SimCLR was barely positive. Ignoring the data points for the entire training set, the benefits consistently decreased with the data reduction. That the performance relies primarily on the amount of data supports the findings in Fig. 2 and strengthens the assumption that embedding transferability is an issue between the tissue types. This effect was also observable in PCam, which is curious since it was used in pertaining. Further, the benefit variance

**Fig. 2.** Top accuracy of each training data split (0.2%, 0.8%, 8%, and 100%) divided by dataset. As the training data is reduced, the classifier performance generally decreases; an exception is the 100% PCam split. The finetuned encoders mostly attain higher accuracies than purely supervised training.

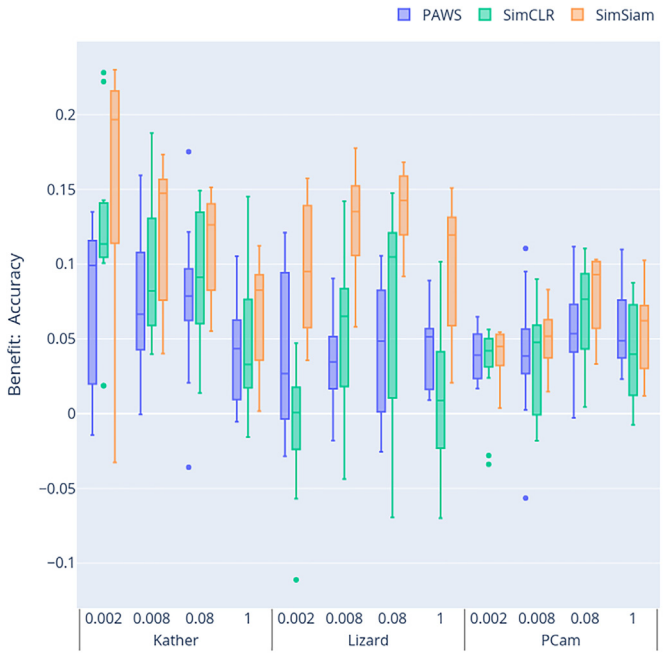



**Fig. 3.** Test set accuracy benefits for each dataset and split size for runs finetuned encoder weights. Overall, the gain with SimSiam compared to PAWS and SimCLR is significant for Kather and Lizard.

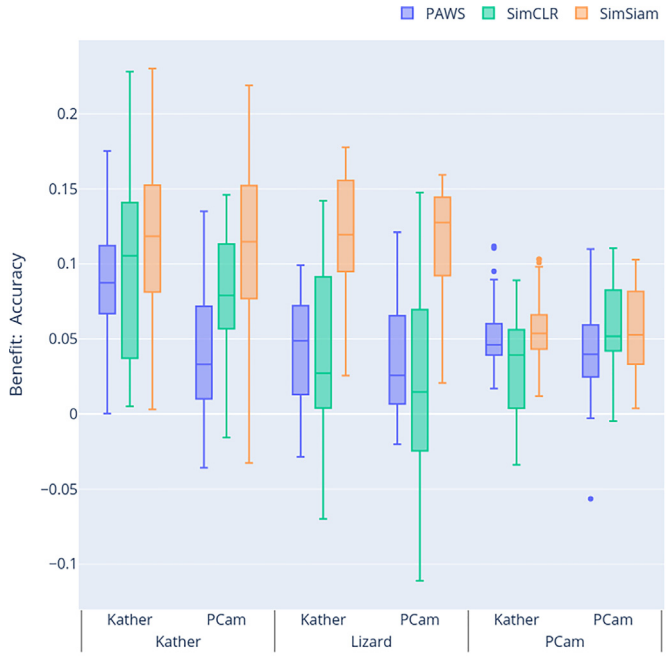



**Fig. 4.** Benefits distribution of finetuned encoders depending on training and downstream task data. The benefit represents the accuracy gained (the mean is noted in the setting label) by comparing the encoder's performance with its supervised counterpart. SimSiam generally outperforms the other methods.

was, in general, significantly more minor compared to the other datasets. Therefore, the finetuned PCam and Kather encoder yielded relatively stable results. We suppose that the Kather encoder embeddings benefit PCam but not Lizard. Overall, the benefits investigation confirms some observations made in Fig. 2 on the individual experiment level.

To disentangle the influence of the different finetuning configurations, we analyzed the benefit distributions separated by network architecture, prediction head, and finetuning of the encoder weights. We append a detailed discussion in Appendix C.1, but the findings indicated that the usefulness of pretraining increases with the network complexity, i.e., architectures with more parameters benefit more. Similarly, SimSiam mostly exceeded the other methods in settings where the encoder weights were finetuned, consistent with the previous observations. However, significant instabilities were observed for SimSiam if the encoder was fixed during the finetuning. A closer examination of the apparent oddity revealed that it is an artifact induced by the unified finetune protocol (Appendix C.3). Further investigating runs with fixed encoders, we find that SimCLR produces the most stable, beneficial embeddings, which aligns with previous observations.

In summary, the expected behavior for reducing the training data available was only observable for Kather. This may be because the finetune protocol was initially adjusted to this dataset. But when averaging over the experimental runs, the results showed that building upon a domain encoder was generally advantageous. This strengthens the assumption that all methods generated encoders with beneficial embeddings for the data domain, while SimSiam performed best in the comparison group.

*Basic histopathological features*

To investigate the in-domain transferability of the encoder embeddings, we reexamined the results generated in 4.2. We studied the benefit distribution of the finetuned encoders based on their training data on every downstream task. Fig. 4 summarizes the findings.

Analyzing the Kather finetuning results, we observed that all methods performed consistently better if their encoders were trained on this dataset. We assume an encoder learns the underlying patterns in its pertaining data. Therefore, the observation was rather logical and supports this assumption. The benefits generally dropped when the finetuning was based on the PCam encoder. However, this was anticipated since the dataset is built upon another tissue type. Though, the methods decreased to a significantly different extent, which was indeed unexpected. Thus, we suppose the encoder embeddings differ considerably, with some learning more features transferable in-domain than others. Method-wise, SimSiam yielded relatively stable results across both encoders while consistently surpassing the other methods. Followed by SimCLR, which was almost on par with SimSiam using the Kather encoder. Even though the median and average benefit of PAWS were positive, the method performed weaker in comparison. However, concluding whether such embeddings represent fundamental histopathological patterns or universal features is challenging. Therefore, we used basic XAI techniques to examine the models on some Kather samples as discussed in Appendix C.2. We found qualitative evidence that the classifiers targeted pathologically relevant tissue structures. Although SimCLR produced the most consistent maps, this was genuine for all methods. Indeed, this conveys the presence of domain-specific embeddings.

Examining the PCam downstream task, we observed a somewhat equivalent behavior, with SimSiam exceeding SimCLR, which in turn outperformed PAWS. Interestingly, the median and the average benefit were higher for PAWS using an encoder trained on Kather. Nevertheless, the differences were minor since the second and third quantiles strongly overlap, and the higher average benefit could result from some positive outliers. Overall, the differences were minor between the methods regarding their general performance and the influence of the encoder training data. This finding contrasts with the one made before on the Kather downstream. We suspect this is a direct consequence of the datasets and can be explained accordingly. The Kather dataset is built upon colorectal cancer and normal tissue. The evenly distributed 9 classes represent almost distinguishable tissue structures from which the classification task is derived. This structural distinction is not reflected in the design of PCam, which consists of randomly selected samples of sentinel lymph node sections that may or may not contain metastatic tissue. Thus, the apparent benefit decrease using a PCam encoder on Kather is explainable. In contrast, the Kather dataset may include helpful information for the PCam task, i.e., cancer and normal tissue, even if from another tissue type.

We chose the Lizard dataset since the intersection of the tissue sections included in the dataset with PCam, and Kather is small. In addition, its classification task, detection of cell nuclei, is, in fact, different from that of the other datasets. Therefore, Lizard represents a proxy metric for learning fundamental histopathological features or at least valuable universal features that work across tissues. Under this assumption, we observed that the feature transfer was not optimal for PAWS or SimCLR, regardless of the encoder training data. SimSiam yielded remarkably better performance, significantly surpassing the other methods. This observation aligns with the findings of Fig. 2 and Fig. 3. However, across all experimental setups, the general performance achieved and benefits gained on the Lizard dataset were mainly inferior to the other datasets. This is strongly supported when studying the experiment runs with a frozen encoder (e.g., Fig. C.2) in more detail. Note that the influence of the initially learned embeddings is maximized in such runs. The benefit distribution was either decreasing or relatively constant, depending on the method, compared to finetuned encoder runs. Hence, the learned embedding was as valuable as those resulting from a freshly initialized network for Lizard. The difference between Lizard and the pretraining datasets seems too pronounced. Accordingly, training encoders on specific datasets seem insufficient to represent the entire histopathological domain.

In summary, we found evidence that the encoders produced embeddings transferable in-domain but with limitations only. The observations suggest that the embeddings lose value while the downstream task data shifts from the pretraining data. Hence, training with a diverse dataset, including various tissue types and structures, is likely a requirement to produce encoder models applicable across the domain. We identify the training of such models as a further research topic, including a closer investigation of the embeddings.

## Conclusions

We have compared a cross-selection of state-of-art semi- and self-supervised learning methods in the histopathological domain. Therefore, we trained encoders with PAWS, SimCLR, and SimSiam on public datasets and evaluated them within a unified framework on several domain tasks.

The findings suggest that pretraining generally has a positive effect, measured by the average benefit and absolute performance, compared to a purely supervised training approach. This effect is more significant for complex network architectures and persists in insufficient training data regimes. Thus, less annotated data is needed to achieve comparable performance if building upon a domain encoder. Furthermore, there is evidence that such models learn underlying domain-specific features applicable across tissue types. Regarding this, initial explorations of encoder models using GradCam are promising, revealing that classifiers targeted structures of interest from a pathological perspective. Yet, according to observations, feature transferability is limited and dependent on encoder training and downstream task data divergence. Accordingly, we identify the relation between training data and resulting feature space as a critical topic of further research to build a general domain encoder. We recommend building a neural network classifier on an in-domain encoder model, especially if less annotated data is available. Therefore, we notice a necessity to expand the sharing of domain models, i.e., a public histopathological model zoo.

As a semi-supervised approach, PAWS is the only method that uses partially labeled data in the training process. This additional information showed no advantage in this work since PAWS could keep up on average but mainly achieved significantly lower values than the other methods. Also, we experienced PAWS results as the most sensitive regarding hyperparameter settings and network initialization, which is potentially unsuitable in the medical field. Across the methods, PAWS has the highest computational resource consumption.

SimCLR is also a relatively computationally expensive method. It yielded the most consistent performance concerning all experimental settings, a valuable property in this domain. Moreover, it mainly achieved comparable results in its absolute metrics and benefit distributions, close to the best-performing method. SimCLR was straightforwardly applicable to the new data domain, i.e., identifying appropriate training settings was uncomplicated.

SimSiam demonstrated the lowest sensitivity regarding weight initialization in the finetuned encoder setting. In comparison, it achieved top evaluation metrics and gained the most relative benefit overall. Additionally, the method consumes, by far, the least computational resources. However, it was the most challenging method to train a performant encoder.

Note that a unified evaluation strategy limited the performance of every method. Though, it enhanced the inter-comparability of the encoders and, therefore, the investigated learning strategies, which is the primary aspect of this work. We assume each method will improve relatively if optimized for a downstream task. Under this assumption and based on the experimental result, we recommend trying SimSiam as a promising alternative or addition to the more established SimCLR, which seems the primary choice for consistent results in the histopathological domain.

## Funding

This work was supported by the Federal Ministry of Education and Research of Germany (BMBF, https://www.bmbf.de/) in the project deep.Health [project number 13FH770IX6]. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

## Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

We thank Tim-Rasmus Kiehl and Sebastian Lohmann from Charite - Universitätsmedizin Berlin for assistance with the medical interpretation of the results and for providing the perspective of pathologists.

We thank our colleagues from CBMI, Jonas Annuscheit and Patrick Baumann, for their input and discussions about our work.

## Appendix A. Encoder training information

### A.1. Augmentation stacks

All methods utilized augmentation strategies to create distinct views of a given image. In addition, to the augmentation stacks published with the associated methods, we implemented custom augmentation stacks suited for the data domain in a supervised setting, based on the findings of Tellez et al,[41] and Annuscheit et al.[46] Fig. A.1 illustrates transformations of a datapoint from the Kather dataset using all different augmentation stacks.

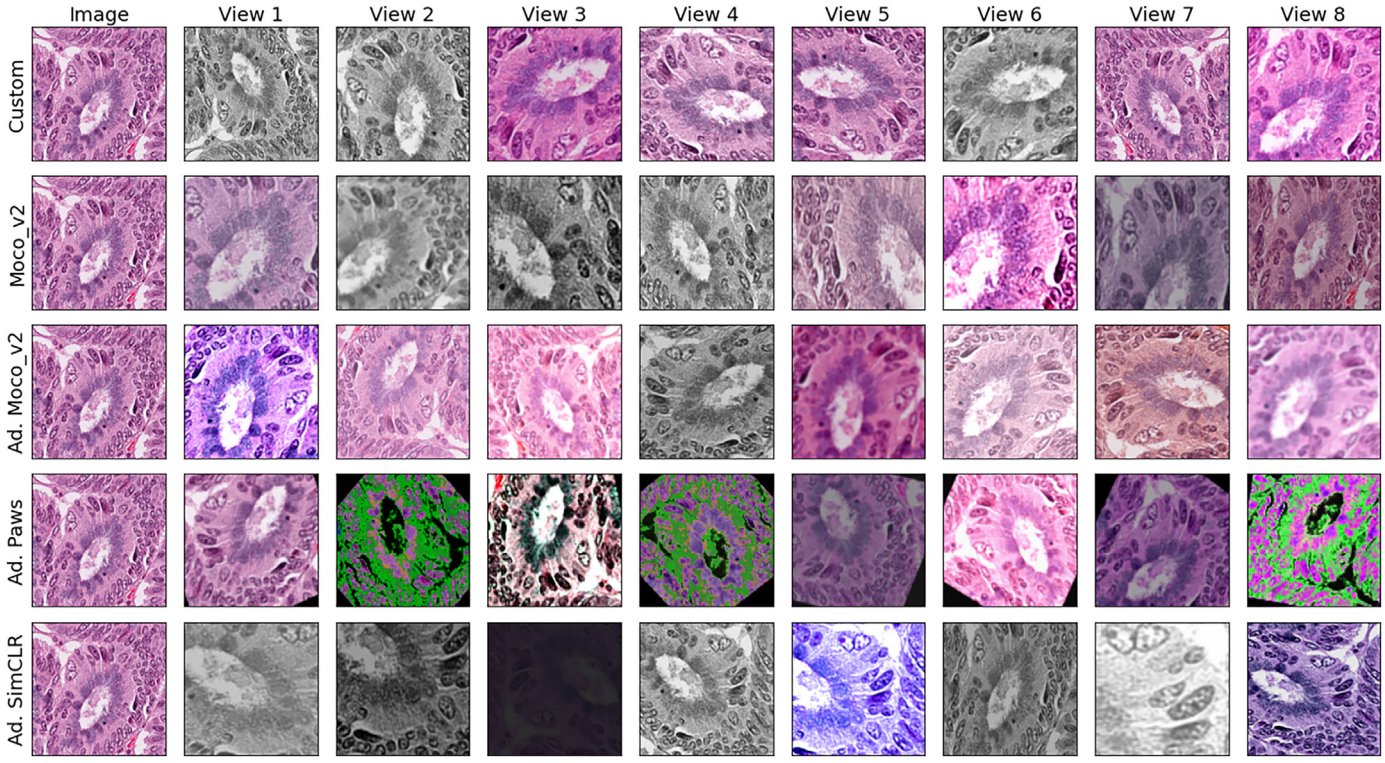



**Fig. A.1.** A Kather sample transformed with the different augmentation stacks used.

First, we trained SimSiam models with the simpler of our adapted histopathological stacks. The stack utilized only a slight color jitter because we expected too much distributional shift and problems in the learning process. Usually, color differences in the input data are a problem when modeling a classifier for histological tasks, e.g., stain normalization problems. Furthermore, gray scaling of the image was applied as a color transformation. As geometric manipulations random cropping, rotating (360°), and flipping (horizontal/vertical) are parts of the augmentation stack. The size of an image is correctly adjusted before and after the rotation transformation to avoid black borders. Each transformation is applied with a probability $p$ with $p \in [0.2; 0.8]$.

We observed that models trained with the stack struggled to learn meaningful representations even though they did not collapse during the process. Hence, we attempted the moco_v2 stack initially used for SimSiam training, leading to considerably better results. However, after further investigations, we found that the parameterization of the color jitter caused the problem. Therefore, we modified our stack to use the moco_v2's heavier color jitter parameterization and also added the Gaussian blur to this method. This new custom stack, adapted moco_v2, achieves similar to better results with the same training settings of SimSiam and SimTriplet learning approaches. For the other learning methods, we also adapted their augmentation stacks to our observations by adding geometric transformations or using different parameterizations but keeping the color manipulations of the respective method.

### A.2. Hyperparameter settings

Table Appendix A.2 reports the final hyperparameter used for the encoder training. In addition to the noted optimizer parameter, each one was wrapped in a 1-cycle cosine scheduling to decrease the learning rate. For the PAWS encoder trained on the Kather dataset, the same setting was used across the different split sizes.

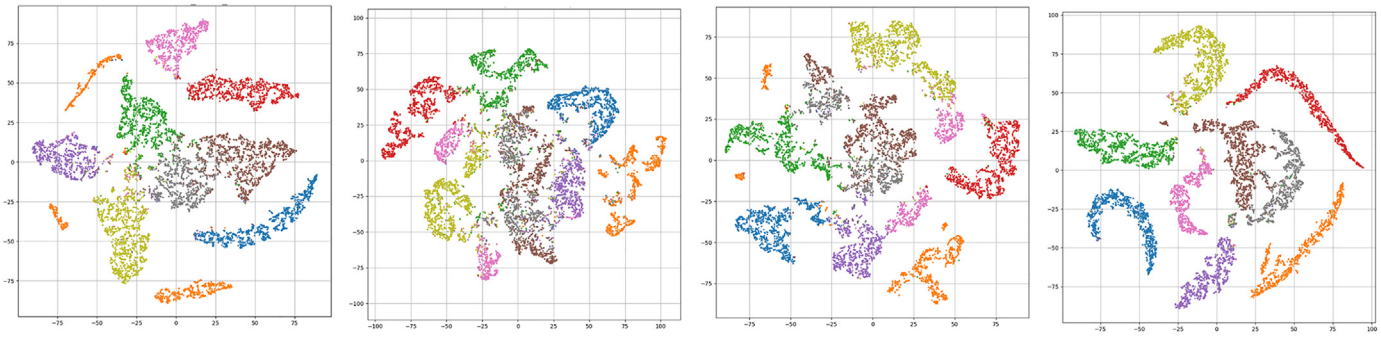
**Table A.1**
Summary of final hyperparameter settings used to train the different encoders. PAWS networks are divided by the size of the support set $D_{sup}$ used. For brevity some column headings are shortened: 'Size' represents the batch size used; the optimizer column includes values for learning rate, momentum, and weight decay; column 'Emb./MLP' reports the values used for the encoder embedding size and the MLP hidden layer size.

| Method | Data | Network | Epoch/Batch | Size | Optimizer | Emb./MLP |
|---|---|---|---|---|---|---|
| PAWS | PCam | R18/.08 | 10/40940 | 64 | LARS (0.467, 0.647, 72e-6) | 128/– |
| | | R18/.008 | 10/40940 | 64 | LARS (1.17, 0.364, 29e-6) | 128/– |
| | | R18/.002 | 10/40940 | 64 | LARS (2.41, 0.317, 31e-7) | 128/– |
| | | R50/.08 | 10/40940 | 64 | LARS (0.88, 0.48, 25e-6) | 128/– |
| | | R50/.008 | 10/40940 | 64 | LARS (1.39, 0.57, 18e-6) | 128/– |
| | | R50/.002 | 10/40940 | 64 | LARS (1.57,0.6, 19e-6) | 128/– |
| | | W28/.08 | 10/40940 | 64 | LARS (2.231, 0.364, 87e-6) | 128/– |
| | | W28/.008 | 10/40940 | 64 | LARS (2.379, 0.49, 82e-6) | 128/– |
| | | W28/.002 | 10/40940 | 64 | LARS (0.784, 0.81, 32e-6) | 128/– |
| | Kather | R18 | 10/28120 | 32 | LARS (1.772, 0.499, 28e-6) | 128/– |
| | | R50 | 10/28120 | 32 | LARS (2.0, 0.5, 3e-4) | 128/– |
| | | Wide | 10/28120 | 32 | LARS (1.5, 0.6, 5e-4) | 128/– |
| SimCLR | PCam | R18 | 23/12000 | 512 | LAMB (6e-3, -, 1e-5) | 1024/64 |
| | | R50 | 23/12000 | 512 | LAMB (6e-3, -, 1e-5) | 2048/256 |
| | | Wide | 23/12000 | 512 | LAMB (17e-4, -, 1e-5) | 2048/128 |
| | Kather | R18 | 77/15000 | 512 | LAMB (7e-4, -, 1e-5) | 2048/256 |
| | | R50 | 77/15000 | 512 | LAMB (3e-3, -, 1e-5) | 1024/256 |
| | | Wide | 77/15000 | 512 | LAMB (3e-3, -, 1e-5) | 1024/64 |
| SimSiam | PCam | R18 | 50/51200 | 256 | SGD (0.5, 0.5, 1e-4) | 512/128 |
| | | R50 | 50/51200 | 256 | SGD (0.5, 0.5, 1e-4) | 2048/512 |
| | | Wide | 50/204800 | 64 | SGD (0.5, 0.5, 1e-4) | 512/128 |
| | Kather | R18 | 115/45000 | 256 | SGD (1.0, 0.5, 1e-3) | 512/128 |
| | | R50 | 29/45000 | 64 | SGD (0.5, 0.9, 1e-4) | 2048/512 |
| | | Wide | 29/45000 | 64 | SGD (0.5, 0.9, 1e-4) | 512/128 |

## Appendix B. Additional results

### *B.1. t-SNE examplesv*

Fig. B.1 shows t-SNE plots of the Kather validation dataset after an encoder training using a ResNet50. To what extent the investigated semi- and self-supervised methods can separate the validation data after training is remarkable. The plots indicate that the separation of some classes is difficult. Especially for the classes MUS (smooth muscle tissue, brown) and STR (cancer-associated stroma, gray), the separate clustering seems to be challenging. Both SimTriplet and PAWS achieve the sharpest separation. This results probably from the direct label information of the PAWS method and the close-patch strategy implementation of the SimTriplet method used here, which gives indirect information about the class affiliation. Also, the background class (BACK, orange), which combines quite diverse structures, appears problematic to unite in one cluster. With this mapping, SimTriplet and SimCLR manage to assign most data points to a joint region.



(a) PAWS (b) SimCLR (c) SimSiam (d) SimTriplet

**Fig. B.1.** Example ResNet50 encoder t-SNE representations of each method on the Kather validation dataset. The representations suggest that the MUS (brown) and STR (gray) classes are the most difficult to separate in the learning process. The most structured clusters create the SimTriplet method (d) on the dataset.

### *B.2. F1-score and ECE results*

We append the corresponding f1-score representations (Figs. B.2, B.3 and B.4), of the accuracy ones reported in the main text. Although performance differences were observed for each method when comparing f1 score/accuracy metrics, they were small and non-significant. In particular, there was no difference in terms of methodological relations. This effect could be expected since the Kather, and PCam datasets are nearly balanced. Only the Lizard dataset contains some imbalances, which could have led to major differences in those metrics. However, these were also not observed, compare Fig. 4 with Fig. B.4 and Fig. 3 with Fig. B.3.

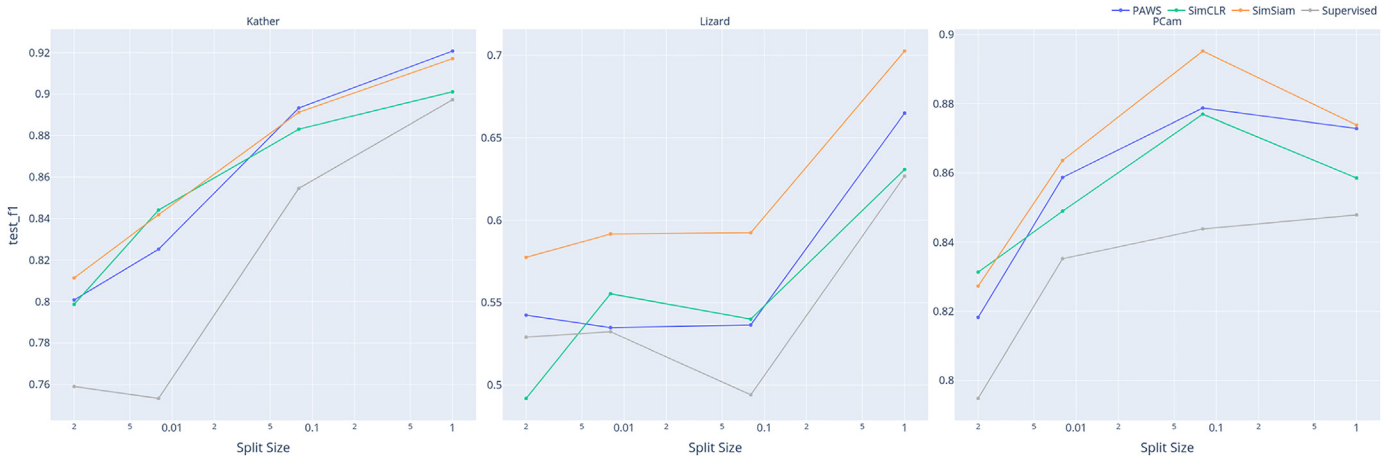



**Fig. B.2.** Top F1-score of each training data split (0.2%, 0.8%, 8%, and 100%) divided by dataset. Corresponding representation to 2.


**Fig. B.3.** Test set f1-score benefits for each dataset and split size for runs finetuned encoder weights. Corresponding representation to 3.

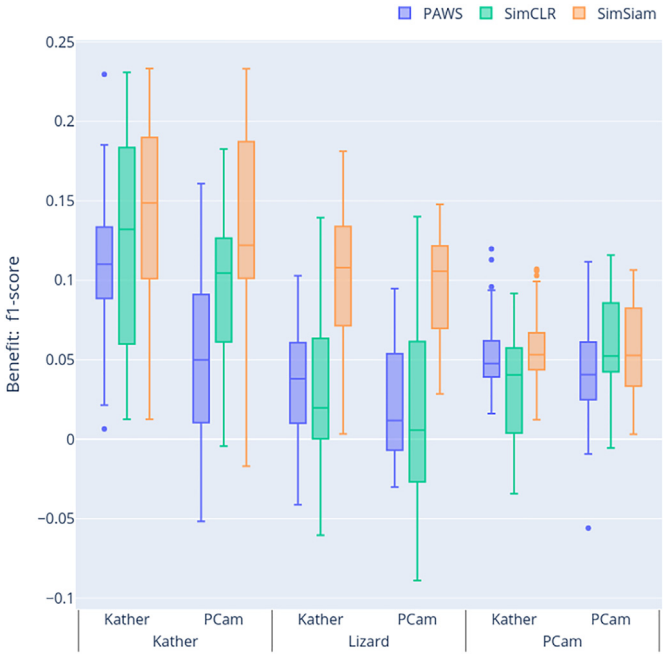



**Fig. B.4.** F1-score benefit distribution of finetuned encoders depending on training and downstream task data. The benefit represents the accuracy gained (the mean is noted in the setting label) by comparing the encoder's performance with its supervised counterpart. Corresponding representation to 4.

In addition, Fig. B.5 reports the distribution of the ECE (expected calibration error) across the different architectural settings. The models are usually relatively calibrated for experiments that finetune the encoder, but some outliers exist. The medians are between 0.05 and 0.1, which is reasonable for networks with that many parameters.

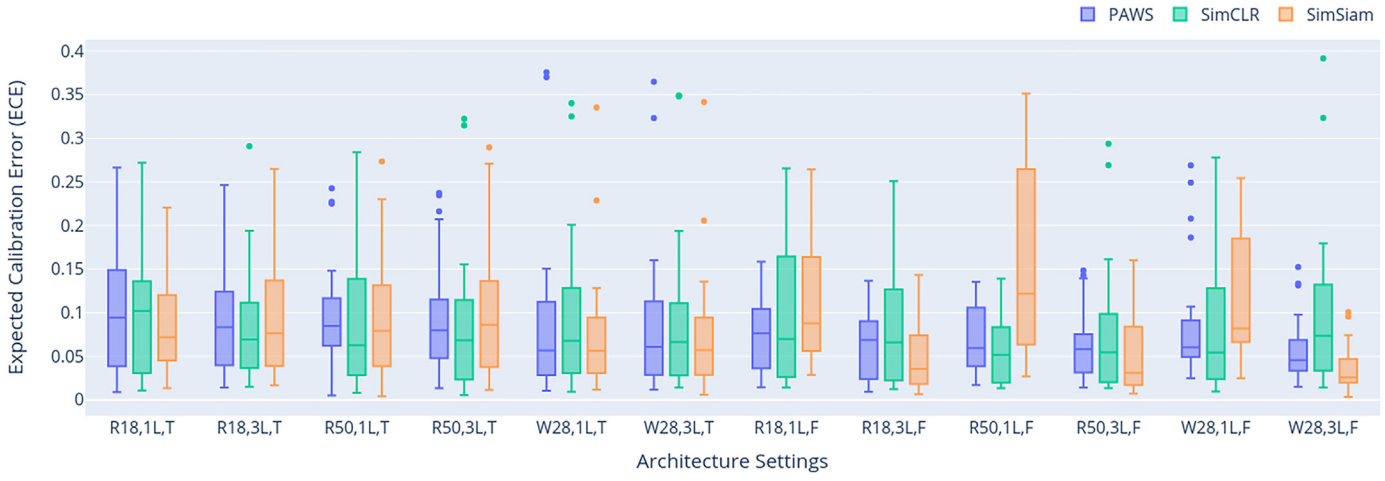



**Fig. B.5.**

## Appendix C. Additional analysis

### C.1. Benefit of architectural settings and frozen encoder

In Sections Encoder initialization sensitivity and Downstream task data reduction, we have investigated the experiments finetuning encoders across the different datasets since this view is comparable to real-world training procedures and, thus, reflects practical usage. However, such finetunings blur the effective contribution of the encoder to the results, and inferences about learned domain-specific features are impaired. To disentangle some of those effects, we analyzed the benefits regarding the finetuning configurations more in-depth. Therefore, we calculated the benefit distribution separated by network architecture, prediction head, and finetuning of the encoder weights and summarized the findings in Fig. C.1.

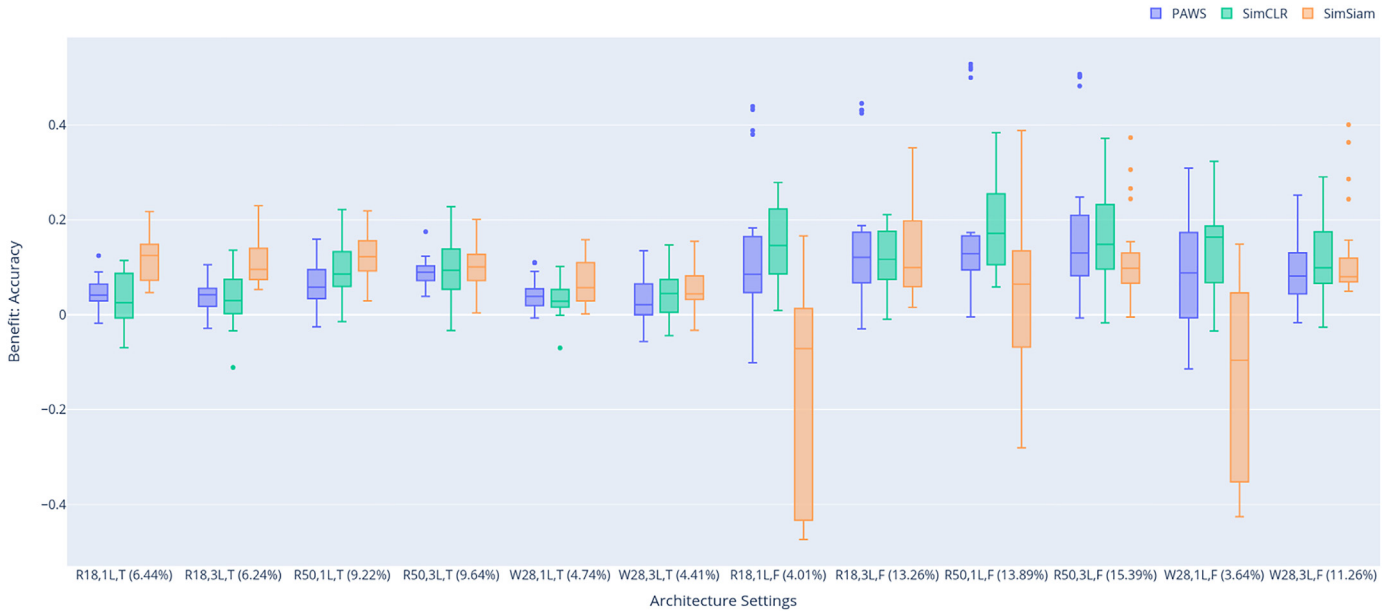



**Fig. C.1.** Benefit distribution over different network configurations. The benefit represents the accuracy gained/lost using a pretrained model instead of a randomly initialized network. The mean accuracy gain is noted in the label of each setting. For brevity, the labels are shortened (R18: ResNet18, R50: ResNet50, W28w2: Wide_ResNet28w2, 1L: logistic regression, 3L: 3-layer MLP, T: encoder weights finetuned, F: encoder weights frozen).

We found only minor influences of the network settings in case the encoder weights were finetuned (indicated by a 'T' in the figure label). There is almost no difference when using logistic regression or a 3-layer MLP prediction head. However, we observed more minor benefits for all methods when using the Wide_ResNet28w2 and for PAWS/SimCLR using the ResNet18. The most consistent positive benefit occurs for the ResNet50 across all methods, although SimSiam showed a similar positive benefit for the ResNet18. We expected an increased benefit for the ResNet50 since networks with increasing complexity also require more data to tune their many parameters appropriately. Therefore, pretraining should be more helpful for more complex architectures.

For finetunings with fixed encoder weights (indicated by an 'F' in the figure label), we observed evident changes. First, some SimSiam configurations were immediately conspicuous. The methods showed tremendous benefit fluctuations in the case of a logistic regression head. The median was negative in 2 of these settings, suggesting that the encoder impaired the classification performance. Even if this is possible, e.g., a (nearly) collapsed feature space, this observation appeared curiously based on the previous results. Therefore, we investigated these runs in more detail and found that adapting the learning rate was sufficient to improve these seemingly corrupted runs. Further, by dropping the encoder MLP part, this issue could be fixed, see Appendix C.3. The MLP encoder part represented an adaption of SimSiam to match the architecture baseline of PAWS. Hence, this observation was not caused by methodological issues but represents an artifact resulting from the unified finetune protocol. Notably, this clearly showed that only the CNN embeddings, as intended by the method, should be used as feature extractors to create performant embedding.

Besides the artifacts, we found the benefits generally increased but registered larger fluctuations in the results. Especially PAWS and SimCLR significantly improves in particular. On the one hand, this seems reasonable since it cannot be assumed that a randomly initialized encoder, like in the supervised runs, will produce linearly separable features without finetuning its weights, and the performance gap measured by the benefit, therefore, should be increased in such settings. On the other hand, the findings could also indicate that the encoder captured beneficial domain properties, resulting in a gain. More detailedly, we also notice that the average benefit increases and its fluctuations decrease if the logistic regression is replaced with a 3-layer MLP here. Again, the ResNet50 gained the most average benefit in each setting, supporting the previous observations that pretraining is more profitable for complex networks.

In Fig. C.2, we reported the corresponding representation to the finetuned encoder runs (Fig. 4), i.e., same settings but without updating the encoder weights during finetuning. SimSiam's results here suffered from the same corrupt experiment runs, leading to tremendous fluctuations. However, we noticed that the benefit median generally increases, even for SimSiam, including the faulty runs. Again, this is expected compared to a supervised run but also indicates the presence of valuable features. Especially if the difference is substantial, like in some observations for PAWS or SimCLR. There was a notable distinction between the encoder of those two methods, i.e., in the case of the Kather-encoder, PAWS benefited more, while for the PCam-encoder, SimCLR surpassed the other methods. SimCLR repeatedly showed minor differences between the different encoders and overall fluctuations, supporting the finding that this method is the most consistent.

The drop behavior between the encoder training set and the downstream task did not differ in the case of the frozen encoder from the previous observations in Section Downstream task data reduction. Hence, it supports the conclusion regarding the transferability of the encoder embeddings, which is an important one.


**Fig. C.2.** Accuracy benefit distribution of frozen encoders depending on training and downstream task data. The benefit represents the accuracy gained (the mean is noted in the setting label) by comparing the encoder's performance with its supervised counterpart. Corresponding representation to 4.

### *C.2. Encoder GradCam and Guided Backprop illustrations*

We investigated the topic qualitatively by examining a part of the resulting models from Section Downstream task data reduction using eXplainable AI methods, i.e., Grad-Cam[43] and Guided Backpropagation[44] on randomly selected examples from all downstream test datasets. Here, we only used encoders with the Wide_ResNet28w2 for the experiment because it produced the largest feature map (14x14) of the 3 architectures before entering the global average pooling layer, using an input size of 96x96 pixels. The feature map size is crucial for the quality of resulting activation maps of the applied technics and consequently allows a more significant analysis. Furthermore, to maximize the impact of the encoder, we only evaluated models that used logistic regression as the prediction head and only considered models that were finetuned on an 8% split of the corresponding training dataset with a frozen encoder. Besides the resulting activation maps, we rendered the normalized version of the original image with a threshold of 0.6 to highlight the relevant content. Parts below the threshold are whitened, and higher activated areas are more color-intense, see Fig. C.3.

We observed a fragmented selection from the purely supervised models. On the *normal colon mucosa* image (Fig. C.3.a), the activation reflected some structures of the original image, which was nevertheless remarkable for a randomly initialized network with a regression head only. However, the higher activations in C.3.e were somewhat arbitrary from a pathologist's point of view.[47] Crypts can be recognized (besides the shape) by the goblet cells. Typically, the nucleus is close to the basement membrane and the cytoplasm of the goblet cells towards the intestine. Thus, only marking the cell nuclei is problematic because these would still have to be distinguished from other cell nuclei, e.g., those of basal cells. However, selecting the cytoplasm of the goblet cells is more meaningful. From a pathological point of view, marking both aspects is the most significant. We observed all gradations in the activation maps of the pretrained models. Almost all models pretrained on the Kather dataset considerably recognized the cytoplasm of the goblet cells. SimSiam (Fig. C.3.d) also marked the cell nucleus structure, albeit slightly. For the 2 encoders pretrained on Kather and PCam, respectively, SimCLR showed the best generalization performance. Both variants identified parts of the cytoplasm and partially marked the same image content, comparing Fig. C.3.c and C.3.g.

The second example (Fig. C.3.i) represented *adipose tissue*. Detecting fatty tissue can be tricky. The fat cells can quickly be perceived as background and useless information due to larger uniform areas. The supervised approach (Fig. C.3.m) targeted some nuclei with no discernible structure. The surrounding area had a much lower activation, and no explicit marking of relevant adipose tissue was noticeable. In comparison, all pretrained models consider this tissue type to varying degrees. One strategy is to mark the cell membrane and the adjacent adipose tissue. Good results were achieved here by several methods. However, SimCLR had consistent results for both encoders again, comparing images C.3.k and C.3.o. The PAWS encoder, trained on the PCam dataset, followed a different approach and targeted the fat tissue directly. The PAWS results also contain a part of the cell membrane and other unimportant tissue in the context, see Fig. C.3.n.

Even though it is a qualitative exploration, the results indicate that the evaluated learning approaches produce models already adapted to the histopathological data domain. Therefore, such models can generate reasonable representations from a pathologist's view to a certain degree. The observation also suggests that encoders trained on a particular tissue type can produce underlying histopathological features usable for another tissue type, i.e., downstream task. Both encoders occasionally selected the same image content or used the same histopathological features across the encoders, regardless of the chosen method.

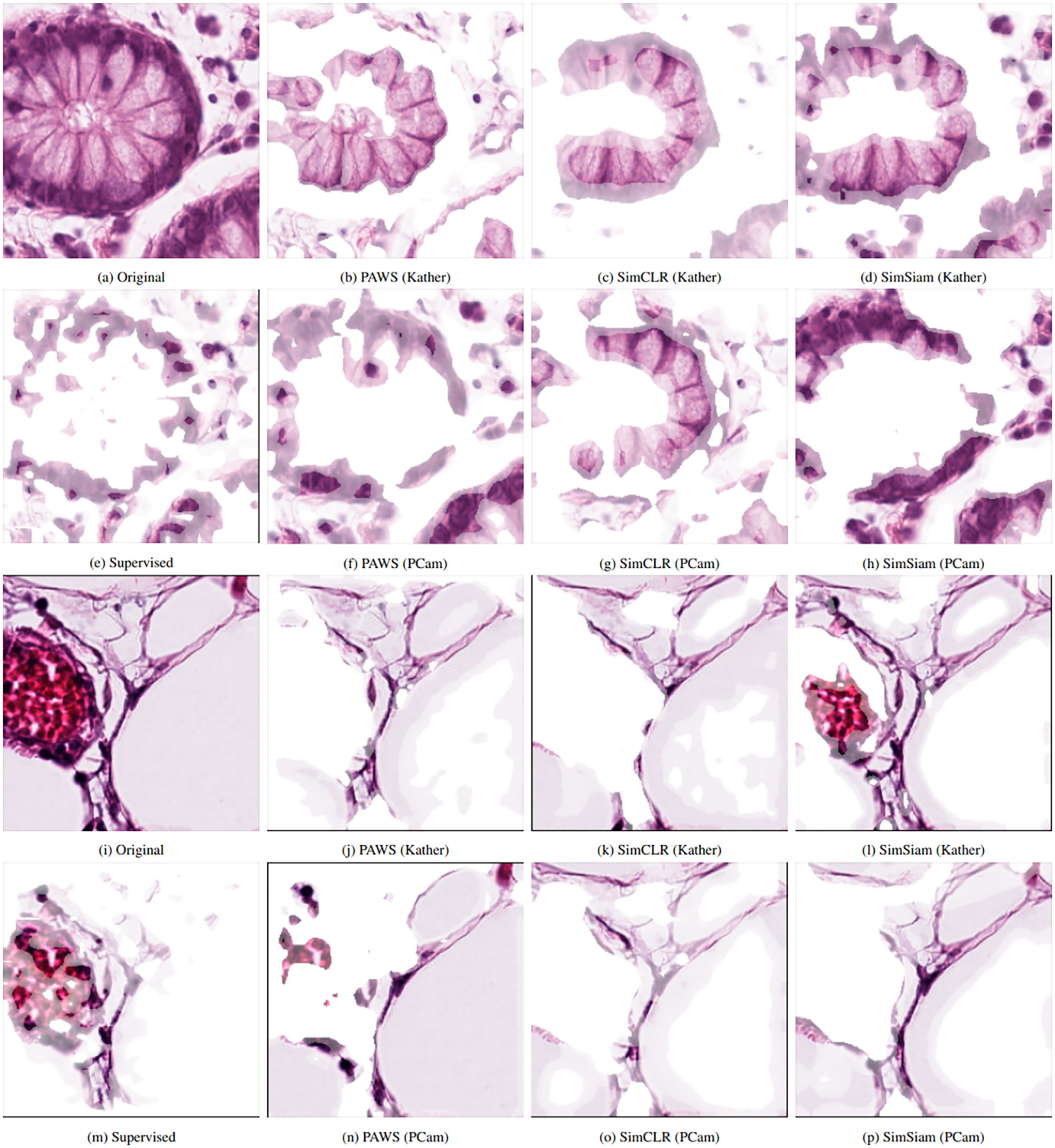



**Fig. C.3.** Exploration of tissue samples with the Grad-Cam method. Images (a) and (i) illustrate the original samples from the Kather data. All other images merge the original image with the activation map produced by Grad-Cam on a particular encoder, revealing the relevant parts for the classification decision. The encoder training data is noted in brackets after the method. Pretrained models make use of domain-specific structures but to a different degree, e.g., selecting goblet cells (b)–(h) or targeting fat tissue (j)–(p).

Overall, the experience applying both methods was that GradCam marked medium-sized image regions of interest based on feature activation maps from the CNNs' last convolutional layer, while Guided Backprop marked relatively small local pixel regions, given the nature of the method. Therefore, GradCam worked more closely with a pathologist's workflow when making classification decisions and showed a decision-making process based on contextual regional information. For Guided Backprop, it seemed that instead of medium-sized regions, strongly H&E colored pixels are marked, which might not always be of interest for a specific classification. Especially for fully supervised experiments, meaning experiments with no feature-based pretraining, this appeared to be the case. Even some Guided Backprop maps that looked like noise were no exception in this setup. However, this problem softens for semi-supervised experiments, and Guided Backprop showed far better maps that looked reasonably congruent to GradCam maps, underlying the importance of feature-based pretraining. We suggest that, when using XAI methods to analyze the feature space of encoder models, multiple strategies should be used and discussed with domain knowledge experts.

### *C.3. MLP ablabation study*

In the original publications of SimCLR and SimSiam, only the base network of the encoder $f$, i.e., without the modified MLP part, is used for further downstream training. To preserve the comparison with PAWS, this was changed in the study. Hence, we also performed all experiment runs of the data reduction experiment with the intended implementation of the methods to examine the influence of these adaptations. We limit ourselves here to report one representation,[7] which can be used to summarize the results of the ablation study. Fig. C.4 illustrates the distribution of accuracy benefits for the different training configurations (corresponding to Fig. C.1).


**Fig. C.4.** Accuracy benefit distribution over different network configurations of SimCLR and SimSiam omitting the additional MLP. The mean accuracy gain is noted in the label of each setting. For brevity, the labels are shortened (R18: ResNet18, R50: ResNet50, W28w2: Wide ResNet28w2, 1L: logistic regression, 3L: 3-layer MLP, T: encoder weights finetuned, F: encoder weights frozen).

In the case of the finetuned encoder runs, we observed that the performance is relatively constant when dropping the MLP. The additional parameters were, therefore, not very profitable in this scenario, which corresponds more to the realistic use case. Even if the MLP learned helpful information in the pretraining. Even though the MLP had contained helpful representations learned in pretraining, this could be negated solely based on the encoder CNN feature and a simple prediction head.

However, the frozen encoder settings were more interesting in this context. We found that dropping the MLP significantly influenced the performance of SimSiam here. In all settings with logistic regression, this leads to an apparent reduction of corrupted runs. Indeed, omitting the encoder MLP when using the ResNet50 architecture fixed the problem completely. In contrast, using a 3-layer prediction head, no significant performance change was observable. We suppose that during pretraining, the MLP learned a poor mapping, which a single linear transformation could not correct in the finetune process, which resulted in corrupted runs. Additional experimental runs without the added MLP were executed to investigate this further. We repeated all ResNet18 and ResNet50 architecture experiments with one exception to the finetune protocol, which was an increased learning rate. This modification, furthermore, evoked a significant performance gain in the case of the ResNet18 but only barely changed the results of the ResNet50, supporting the idea of harmful mapping. Overall, these are strong indications that this problem was induced by the design of the pretraining and finetuning protocol, not the method itself. Thus, we recommend using SimSiam as initially proposed, i.e., using only the CNN structure as a feature extractor.

[7] However, all presented diagrams and the corresponding f1-score charts can be generated from the raw experimental data using the reproducibility script found in the git repository.

## References


1. Russakovsky O, Deng J, Su H, et al. ImageNet large scale visual recognition challenge. Int J Comput Vision (IJCV) 2015;115:211–252. https://doi.org/10.1007/s11263-015-0816-y.
2. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Adv Neural Inform Process Syst 2012;25.
3. Azizi S, Mustafa B, Ryan F, et al. Big self-supervised models advance medical image classification. Proceedings of the IEEE/CVF International Conference on Computer Vision; 2021. p. 3478–3488.
4. Mahmoud ASSRAN, et al. Semi-supervised learning of visual features by non-parametrically predicting view assignments with support samples. Proceedings of the IEEE/CVF International Conference on Computer Vision. In. 2021. p. 8443–8452.
5. Chen T, Kornblith S, Norouzi M, Hinton G. A simple framework for contrastive learning of visual representations. International Conference on Machine Learning. PMLR; 2020. p. 1597–1607.
6. Chen X, He K. Exploring simple siamese representation learning. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2021. p. 15750–15758.
7. Veeling BS, Linmans J, Winkens J, Cohen T, Welling M. Rotation equivariant cnns for digital pathology. International Conference on Medical image computing and computer-assisted intervention. Springer; 2018. p. 210–218.
8. Kather JN, Halama N, Marx A. *100,000 histological images of human colorectal cancer and healthy tissue*. 2018. https://doi.org/10.5281/zenodo.1214456.
9. Graham S, Jahanifar M, Azam A, et al. Lizard: a large-scale dataset for colonic nuclear instance segmentation and classification. Proceedings of the IEEE/CVF International Conference on Computer Vision; 2021. p. 684–693.
10. McAlpine ED, Michelow P, Celik T. The utility of unsupervised machine learning in anatomic pathology. Am J Clin Pathol 2021;157:5-14. https://doi.org/10.1093/ajcp/aqab085.
11. Raina R, Battle A, Lee H, Packer B, Ng AY. Self-taught learning: transfer learning from unlabeled data. Proceedings of the 24th International Conference on Machine Learning; 2007. p. 759–766.
12. Van den Oord A, Li Y, Vinyals O. Representation learning with contrastive predictive coding. arXiv e-prints 2018.arXiv–1807.
13. Chen X, Fan H, Girshick RB, He K. Improved baselines with momentum contrastive learning. ArXiv 2020.abs/2003.04297.
14. Saillard C, Dehaene O, Marchand T, et al. Self supervised learning improves dmmr/msi detection from histology slides across multiple cancers. arXiv preprint 2021. arXiv:2109.05819.
15. Kather JN. Histological images for MSI vs. MSS classification in gastrointestinal cancer. FFPE Samples 2019. https://doi.org/10.5281/zenodo.2530835.
16. Baykaner K, Xu M, Bordeaux L, et al. Image model embeddings for digital pathology and drug development via self-supervised learning. bioRxiv 2021.
17. McInnes Leland, John Healy, Saul Nathaniel, Lukas Großberger. UMAP: Uniform Manifold Approximation and Projection. Journal of Open Source Software 2018;3(29):861. https://doi.org/10.21105/joss.00861.
18. Lu Ming Y, Chen Richard J, Mahmood Faisal. Semi-supervised breast cancer histology classification using deep multiple instance learning and contrast predictive coding (Conference Presentation). In: *Tomaszewski John E, Ward Aaron D, eds.* Proc. SPIE 11320, Medical Imaging 2020: Digital Pathology; 2020, 11320. https://doi.org/10.1117/12.2549627.
19. Aresta G, Araújo T, Kwok S, et al. Bach: grand challenge on breast cancer histology images. Med Image Anal 2019;56:122–139.
20. Stacke K, Lundström C, Unger J, Eilertsen G. Evaluation of contrastive predictive coding for histopathology applications. Machine Learning for Health. PMLR; 2020. p. 328–340.
21. Chen Richard J, Lu Ming Y, Wang Jingwen, Williamson Drew FK, Rodig Scott J, Lindeman Neal I, Mahmood Faisal. Pathomic fusion: an integrated framework for fusing histopathology and genomic features for cancer diagnosis and prognosis. IEEE Transactions on Medical Imaging, IEEE 2020;41(4):757–770.
22. Dehaene O, Camara A, Moindrot O, de Lavergne A, Courtiol P. Self-supervision closes the gap between weak and strong supervision in histology. arXiv preprint 2020.arXiv:2012.03583.
23. Koohbanani NA, Unnikrishnan B, Khurram SA, Krishnaswamy P, Rajpoot N. Self-path: self-supervision for classification of pathology images with limited annotations. IEEE Trans Med Imaging 2021;40:2845–2856.
24. Srinidhi CL, Kim SW, Chen F-D, Martel AL. Self-supervised driven consistency training for annotation efficient histopathology image analysis. Med Image Anal 2022;75, 102256.
25. Grill J-B, Strub F, Altché F, et al. Bootstrap your own latent-a new approach to self-supervised learning. Adv Neural Inform Process Syst 2020;33:21271–21284.
26. Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R. Signature verification using a" siamese" time delay neural network. Adv Neural Inform Process Syst 1993;6.
27. Li T, Feng M, Wang Y, Xu K. Whole slide images based cervical cancer classification using self-supervised learning and multiple instance learning. 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE); 2021. p. 192–195. https://doi.org/10.1109/ICBAIE52039.2021.9389824.
28. Yang P, Hong Z, Yin X, Zhu C, Jiang R. Self-supervised visual representation learning for histopathological images. International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer; 2021. p. 47–57.
29. Zhang J, Hua Z, Yan K, et al. Joint fully convolutional and graph convolutional networks for weakly-supervised segmentation of pathology images. Med Image Anal 2021;73, 102183.
30. Ciga O, Xu T, Martel AL. Self supervised contrastive learning for digital histopathology. Mach Learn Appl 2022;7, 100198. URL: https://www.sciencedirect.com/science/article/pii/S2666827021000992, https://doi.org/10.1016/j.mlwa.2021.100198.
31. Wang H, Zheng H, Chen J, Yang L, Zhang Y, Chen DZ. Unlabeled data guided semi-supervised histopathology image segmentation. 2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). IEEE; 2020. p. 815–820.
32. Wei J, Suriawinata A, Vaickus L, et al. Generative image translation for data augmentation in colorectal histopathology images. Proc Mach Learn Res 2019;116:10.
33. Wei J, Suriawinata A, Vaickus L, et al. Generative image translation for data augmentation in colorectal histopathology images. In: *Dalca AV, McDermott MB, Alsentzer E, et al, eds.* Proceedings of the Machine Learning for Health NeurIPS Workshop. Volume 116 of Proceedings of Machine Learning Research. PMLR; 2020. p. 10–24. https://proceedings.mlr.press/v116/wei20a.html.
34. de Vulpian A, di Proietto V, Roy G, Hadj SB, Fick RR. A semi-supervised deep learning approach for multi-stain foreground segmentation in digital pathology. Medical Imaging with Deep Learning; 2022. https://openreview.net/forum?id=6uw53DAsjNG.
35. Liu Kun, Liu Zhuolin, Liu Sidong. Semi-supervised breast histopathological image classification with self-training based on non-linear distance metric. IET Image Processing 2022;16(12):3164–3176.
36. Sikaroudi M, Safarpoor A, Ghojogh B, Shafiei S, Crowley M, Tizhoosh HR. Supervision and source domain impact on representation learning: A histopathology case study. 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC). IEEE; 2020. p. 1400–1403.
37. Bejnordi BE, Veta M, Van Diest PJ, et al. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. Jama 2017;318:2199–2210.
38. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2016. p. 770–778.
39. Li L, Jamieson K, Rostamizadeh A, et al. A system for massively parallel hyperparameter tuning. Proc Mach Learn Syst 2020;2:230–246.
40. van der Maaten L, Hinton GE. Visualizing data using t-sne. J Mach Learn Res 2008;9:2579–2605.
41. Tellez D, Litjens G, Bándi P, et al. Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology. Med Image Anal 2019;58, 101544.
42. Chen T, Kornblith S, Swersky K, Norouzi M, Hinton GE. Big self-supervised models are strong semi-supervised learners. Adv Neural Inform Process Syst 2020;33:22243–22255.
43. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-cam: visual explanations from deep networks via gradient-based localization. Proceedings of the IEEE International Conference on Computer Vision; 2017. p. 618–626.
44. Springenberg Jost Tobias, Dosovitskiy Alexey, Brox Thomas, Riedmiller Martin A. Striving for Simplicity: The All Convolutional Net. CoRR 2014.abs/1412.6806.
45. Kather JN, Krisam J, Charoentong P, et al. Predicting survival from colorectal cancer histology slides using deep learning: a retrospective multicenter study. PLoS Med 2019;16, e1002730.
46. Annuscheit J, Voigt B, Fischer O, et al. Systematic investigation of basic data augmentation strategies on histopathology images. Artificial Intellgence – Application in Life Sciences and Beyond. The Upper Rhine Artificial Intelligence Symposium UR-AI 2021, arXiv; 2021. p. 39–48. https://doi.org/10.48550/ARXIV.2112.05657.
47. Kiehl T-R. Private Communication 2022.