EXPLORING SELF-SUPERVISED LEARNING WITH GEOMETRIC TRANSFORMATIONS

tanja de vries

Faculty of Science and Engineering
University of Groningen

July 7, 2023

Tanja de Vries: *Exploring Self-Supervised Learning with Geometric Transformations,* © July 7, 2023

SUPERVISORS:
Ioannis Giotis (Daily supervisor from ZiuZ Visual Intelligence)
Michael Biehl
Estefania Talavera Martinez

LOCATION:
Groningen

DATE:
July 7, 2023

ABSTRACT

We made a self-supervised model with a pretext-task based on geometric transformations. Our model is based on RotNet, a model that predicts image rotations by Gidaris et al. [GSK18]. We implemented our model in TensorFlow, and were able to reproduce their results for rotation. The rotation model cannot be used for all datasets, for example datasets with a lot of top-down images or images of round objects. Therefore, we modified our network to predict the scale of an image. In order for our scale model to work, the used dataset needs to have a defined scale, that is, all images are taken from the same distance. We see the patterns that show the pretext task learns some useful features. However, we also see that for most experiments the gap with supervised is bigger for our scale model than for our rotation model. Most promising is that our scale model closes the gap with supervised learning during our experiment with a low amount of labelled data.

CONTENTS

1 INTRODUCTION

In recent years, supervised deep neural networks, especially convolutional neural networks [LeC+98], have shown to obtain impressive results on image classification (e.g. [He+16]). However, it can be difficult to construct a good dataset, the images need to be of decent quality and there should not be too many outliers. Moreover, a lot of human-annotated labels are needed in order for these networks to perform well. Especially in the medical field, where experts are needed to label the data, it can be very expensive to collect enough labels for a supervised network to perform well.

Research is done to improve the unsupervised methods, in order to obtain good results without labels. However, we would like to benefit from supervised learning with smaller datasets too. A well-known method to help with this is transfer learning [PY10]. If this method is used with supervised learning, the model is first trained on a large, already labelled, dataset. For example, ImageNet [Den+09] is often used for this purpose. The obtained knowledge is then transferred to a new network that will be trained on the small dataset. Because a lot of the more basic information is learned on the big dataset, the model can use the limited data from the small dataset to learn the details specific to the final task. Due to the nature of neural networks, transfer learning will work best when the two datasets are similar.

A more novel method is self-supervised learning, which uses supervised networks on unlabelled datasets [JT20]. The idea of self-supervised learning is older, but due to large unsupervised datasets being available it became a popular topic for research around 2018 [Gui+23]. The first self-supervised methods often had as motivation to learn invariances [Ran+07], or a model that can be used for a lot of different tasks without a lot of human time for designing [Ahm+08]. Self-supervised networks learn a pretext-task for which the labels can be automatically generated. For example, a network by Larsson et al. uses the original pictures as labels and the black and white version as input [LMS17]. The network has to learn how to colourize the image, and in order to do this the network will learn to detect what objects are in the image. Some other pretext tasks include solving jigsaw puzzles [NF16], or predicting the applied data augmentation [JF18]. Although self-supervised learning can be used as an unsupervised method, it is often combined with transfer learning and a labelled dataset. It can also be used with a dataset that has human-annotated labels for only a small part of the data. The pretext-task is then trained on the whole dataset, and the final part is trained on the labelled part of the dataset.

1.1 SCOPE OF THESIS

Our research is based on a network called RotNet by Gidaris et al. [GSK18]. Their model is self-supervised, and learns as pretext task to predict the applied rotation to an image. The intuition behind the model is that it has to learn what is depicted on the image in order to determine whether this image is rotated. They have promising results, however the model cannot be used for datasets that do not have a natural 'upright' position. For example top-down images and images of round objects. Gidaris et al. mention their model could be used with other geometric transformations, but they also explain why they think rotation works best. In this thesis we re-implement RotNet and expand upon it to use scale instead of rotation.

In Chapter 2 we will explore self-supervised learning in more detail. In Chapter 3 we look at the theory behind RotNet and explain how we expand on their research by using scale as well. In Chapter 4 we show that we are able to reproduce the results from RotNet with our own model. Moreover we show the results of our scale experiments and explain the conditions for a dataset that can be used.

2 BACKGROUND

In this chapter we discuss the state of the art of self-supervised learning. First a general introduction is given and then an in-depth explanation is given for some methods, based on the categories these methods can be divided into. We have used the extensive survey on self-supervised learning from Jing and Tian as a guideline [JT20]. They divide the self-supervising networks into four categories: generation-based, context-based, free semantic label-based, and cross modal-based. However, since we focus solely on images in combination with image classification as the downstream task, we will only look into generation- and context-based methods. We refer to their paper for a more extensive overview on self-supervised learning. Note that some other papers use different categories.

Before we look at the state of the art of self-supervised learning, we introduce some of the terms used around self-supervised learning. In order to use supervised learning we do need labels, however instead of *human-annotated labels*, self-supervised learning uses *pseudo labels* that can be automatically generated. The network is then trained on these pseudo labels, and this is called the *pretext task* or *proxy task*. A good pretext task lets the network capture visual features which will be relevant for the *downstream task*. In our case, the downstream task is always image classification.

As mentioned in the introduction chapter, our motivation to look into self-supervised learning is based on the enormous amounts of human-annotated labels that are needed for supervised learning. We want to make use of the techniques and good results found in supervised learning without the drawback of needing all these human-annotated labels. A good pretext-task is chosen in such a way that the network learns useful features while learning to solve the pretext task. The network is basically tricked into learning something else while training on the pretext task. Many pretext tasks have been shown to give good results.

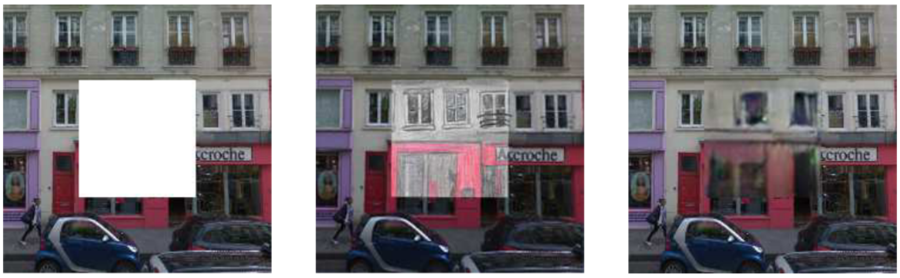
2.1 GENERATION-BASED METHODS

Generation-based methods learn from pretext-tasks that involve image generation. These models mostly consist of two cases, for both the pseudo label is usually the original image. In the first case, something is removed from the image and the model has to generate the missing parts. In the second case, something extra is generated and the image has to recognise what parts are not from the original image.

The intuition behind these methods is that the model has to learn to interpret the objects in the image in order to fill in the missing pieces. Hence, it could learn useful features for the downstream task.

A Generative Adversarial Network (GAN) framework was proposed by Goodfellow et al. [Goo+14]. This framework consists of two models that are simultaneously trained; a generator and a discriminator. The generator captures the data distribution and generates new images similar to the images in the dataset. The discriminator then classifies the image to be either generated or from the dataset. The two networks are thus opposing each other, hence the name adversarial. One application is for example StyleGAN [Kar+20], where the generator is used for the popular website www.thispersondoesnotexist.com to generate non-existing faces. When GANs are used for self-supervised learning, often the discriminator is used as basis for the downstream task. Some examples are crack detection by Zhang et al. [ZZC20], or artefacts detection by Jenni et al. [JF18]. A survey by Qi et al. focuses on self-supervised learning with adversarial pre-training [QS22].

Another generation-based method that uses adversarial learning is image inpainting by Pathak et al. [Pat+16]. In this method a part of the image is removed and has to be generated by the network. The adversarial loss is combined with a reconstruction loss that compares the generated image to the ground-truth, the removed part of the original image. In Figure 1 an example is shown where the centre of the image is removed. In Figure 1b we see that a human artist can interpret the surrounding and hence fill in the missing part. In the network output we see that the network also interprets the entire image, it does not just give an interpolation. This can be seen specifically at the row of windows, the network has added an additional window in the middle, which does not touch any of the edges of the missing part. This makes the method suitable as pretext-task, because it learns to interpret the whole image.



(a) *Network input* (b) *Human artist* (c) *Network prediction*

Figure 1: *Example of image inpainting task. Given an image with a missing region (a), a human artist has no trouble inpainting it (b). Automatic inpainting predicted by the network in [Pat+16] (c). This figure is reproduced based on [Pat+16].*

One generation-based method that does not use adversarial training is image colourization by Larsson et al. [LMS17]. The image colourization method gives the network a gray-scale image as input and it has to generate the colour. Here, the intuition is again that the network has to understand the objects and context of the image in order to make a good prediction of the colours. Zhang et al. also use image colourization as a pretext task for self-supervised learning [ZIE16]. Although image colourization can work without adversarial training, the two methods can also be combined, as proposed by Treneska et al. [Tre+22].

2.2 CONTEXT-BASED METHODS

Context-based methods have a pretext-task based on the context of the image. Specifically, we look at the spatial context. Most of the methods we have seen so far make use of the context features in the image, but we call a model context-based if the context features are the main focus. Although generation-based methods have proven to be useful in self-supervised learning, they are often computationally expensive.

One context-based method is solving jigsaw puzzles. Doersch et al. take patches from the image, give the location of one patch and let the model predict the relative position of another patch [DGE15]. An example is shown in Figure 2, the position of the face is given and the model has to predict the location of the ear. This model is designed in such a way that it learns to recognize the parts of the object in the image and cannot solve the task trivially. There are for example gaps between the patches and the position of the patches have some randomization, such that the model cannot connect the lines or patters too easily. Moreover, normalization is done for each patch separately, such that the statistics are not too similar along the borders. A variant of this method is done by Chen et al., in their work small patches are swapped and the network has to learn to restore

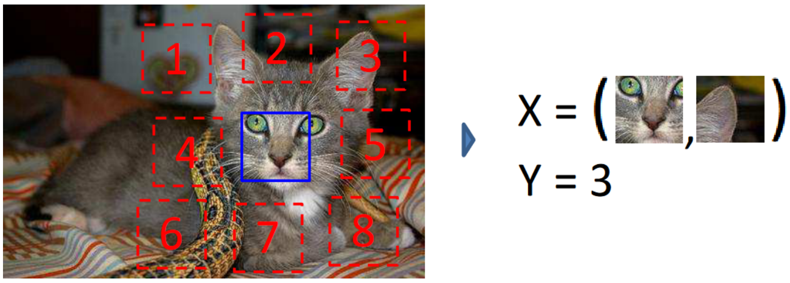


Figure 2: *Example of a jigsaw puzzle pretext task. In this example the position of the face is given and the model has to predict the location of the ear. This figure is reproduced based on [DGE15].*

the image [Che+19]. Similarly, Noroozi et al. take 9 patches from an image and give them shuffled to the network, the model then has to predict the order of all the patches [NF16]. They later combine jigsaw puzzles with clustering to obtain even better results [Nor+18].

Other context-based method often use geometric transformations. For example Dosovitskiy et al. train their model to recognize for any transformed image from which original image it was constructed [Dos+14]. In order to do this they take a subset of the dataset, and make a class for each of those images, consisting of the original image and a lot of transformed images. Some of the applied transformations are translation, scaling, rotation, change to the colour, or a combination. Chen et al. use a combination of random crop and colour distortion, but instead of making classes they use contrastive learning [Che+20]. A paper by Purushwalkam et al. explains why these methods with aggressive augmentation are very successful, they conclude it is partially due to learning a lot of invariances [PG20]. There are also methods based on only one transformation. For example, Gidaris et al. train a model to predict image rotations [GSK18]. One of the advantages of this model is that it does not need special rules to make sure the model does not solve the task in a trivial manner.

3 METHODOLOGY

In this chapter we explain the structure of the model used in this project. Furthermore, we discuss the options and challenges in choosing a transformation for the pretext task. Lastly, we will introduce the datasets that we have used in the experiments.

3.1 MODEL

Our model is based on the paper and implementation by Gidaris et al. [GSK18]. We decided to re-implement their method ourselves in order to fully understand how the algorithm works and how we can adapt it later on. One major difference is that our model is implemented with TensorFlow, where their model is implemented with PyTorch. This results in different values for some of the hyperparameters, due to differences of implementation between PyTorch and TensorFlow. The structure of our implementation is modular, such that we can easily choose a different transformation, dataset or network. We will give a short overview of some of the choices we made. For a more in-depth explanation we refer to the code and the corresponding readme file. Our implementation can be found at https://github.com/TanjaDV/MasterThesis.

Our model is trained to perform image classification in two parts.The pretext task learns the geometric transformations, and the downstream task learns the final classification. The model learns by minimizing a loss function. We use for both tasks the categorical cross-entropy loss. This needs a probability distribution as input, so we will first transform the output of the model using a softmax function. Categorical cross-entropy loss measures how different two probability distributions are. The loss value depends, among other things, on the used weight decay and cannot easily be compared between experiments. Therefore, we report here the more intuitive accuracy score.

Gidaris et al. conclude that keeping all transformations of an image in the same batch improves results and we also follow this guideline [GSK18]. We shuffle the batch such that the network does not always process images and their transformations together and in the same order. The model is then trained to recognize the applied rotation. After learning the pretext-task we also want to train on the original image classification. For this we use transfer learning. We take part of the pretext model and lock the weights which means we can no longer train these layers, then we add a few dense layers which are trainable. The downstream task is then trained on these layers using the human-annotated labels.

3.1.1 *Rotation*

The paper of Gidaris mentions that their algorithm could be used with any geometric transformation, but they think rotation works best. They give several reasons for this, the first being that rotations are usually well-defined. That is, most natural images have a clear definition of being 'upright'. Secondly, by using flip and transpose operations, rotations of multiples of 90 degrees can be applied without visual artefacts on square images. Lastly, they expect learning to recognize rotation is a good pretext-task, because the model has to identify the objects in the image in order to determine the rotation. In that way it learns useful information for the downstream task.

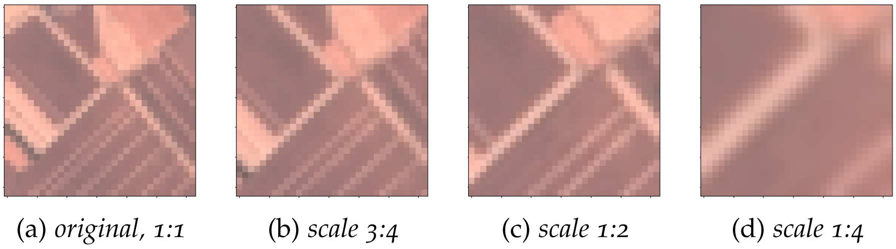
Although rotation can be very useful, there are datasets that contain images where rotation is not well-defined. These include for example top-down images and images with a lot of round objects. Moreover, there can be datasets where it can learn the rotation in a trivial manner, for example if there is always a visible horizon line in the top half of the picture. So our rotation model is limited in the datasets it can be used for. Therefore we want to explore the possibilities with other transformations.

3.1.2 *Other transformations*

We can modify the network to train on other transformations, however that does not mean it will prove to be an useful pretext task. The examples Gidaris et al. give are rotation and scale, and image transformations that change the aspect ratio, i.e. non-uniform scale. Other examples of transformations include translation and shear. However, there are also more complicated transformations, i.e. projection to a different shape.

So, what would make a transformation suitable for our network? We want a transformation that can be learned by a neural network, which is the case for most if not all transformations. It has to be a transformation that is easy to compute and that does not give a lot of visual artefacts, so we want to keep the transformation simple. Last but not least, it has to learn something useful for the downstream task. It is not easy to know what transformations will work as a pretext task.

Combining this information, we decided to look into scale. This is an easy operation, and together with rotation a basis for a lot of other transformations. So knowing more about scale will also give useful information for future research on other transformations.



(a) *original, 1:1* (b) *scale 3:4* (c) *scale 1:2* (d) *scale 1:4*

Figure 3: *Example of applied scales on an EuroSAT image. Scale 1:4 is also scaled up.*

3.1.3 *Scale*

Scale is not as well-defined as rotation. As humans we can easily determine whether an image is rotated. For scale however, it is usually impossible to see if it is zoomed from the original image, because an image could also be taken closer to the object. So, we do need some additional information to make scale well-defined. If we would look at images of people and give the additional information that the original scale always consists of an image with the whole body depicted, then we could easily tell the photo was zoomed a lot if we see a picture which shows only a head. For humans it will be difficult to determine how much it was zoomed in, but we know that is easier for a computer. Moreover, we do not ask for a continuous scale, we will have discrete options. Thus, in order to use scale as pretext task we need a dataset which consists of pictures that have the same original scale in terms of pixel size. That is, all pictures are taken from the same distance. These include for example datasets with satellite images or microscopic images.

There are multiple ways to scale an image. We use uniform scale, so the image is scaled with the same percentage horizontally and vertically. To scale an image we crop the image down to the middle part. Then the different scaled images will all be resized to the same size. To reduce visual artefacts we try to only scale the images down. We have experimented a little with different options for scales and found the combination $\{1:1, 3:4, 1:2, 1:4\}$ to give the best result. An example of the four scales is shown in Figure 3.

3.2 DATASET DESCRIPTIONS

In this section we describe the datasets that are used in the experiments. CIFAR-10 is used for the rotation experiments and EuroSAT for the scale experiments.

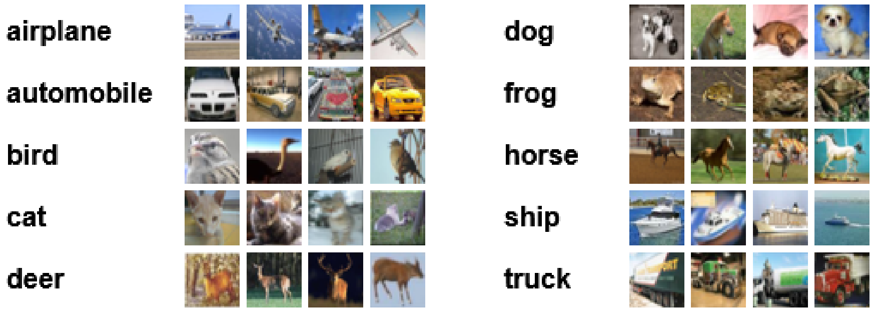


Figure 4: *CIFAR-10 classes as shown in CIFAR-10 documentation.*

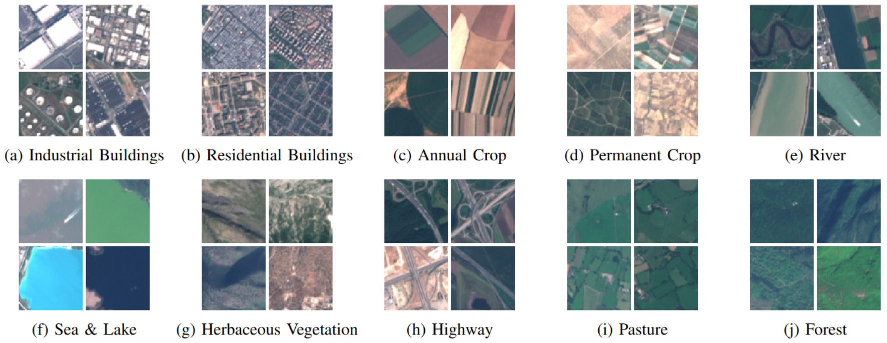


Figure 5: *EuroSAT classes as depicted in [Hel+19].*

3.2.1 *CIFAR-10*

CIFAR-10 is a labelled dataset that consists of 60.000 coloured images of animals and vehicles [KH+09]. It consists of the 10 following classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. See Figure 4 for example images of the classes. The images are 32x32 and evenly split among the classes. So each class has 6.000 images. There are 50.000 training images and 10.000 test images.

3.2.2 *EuroSAT*

EuroSAT consists of 27.000 satellite images divided over 10 classes [Hel+19]. See Figure 5 for the classes and some example images. Each class contains 2.000 to 3.000 images. So, they are not evenly split among the classes. The images are 64x64 pixels. We use the RGB images, which have a spatial resolution of 10 meters per pixel. This dataset does not have a test set. We have chosen this dataset because satellite pictures are made from a set distance. Moreover, the dataset is similar to CIFAR-10 in number of classes and size of images, so it is expected to work with the same network architecture.

4 EXPERIMENTS AND RESULTS

In this chapter we will go over the experiments that we performed and show the results. In the experiment setup we explain the settings that are the same for all experiments. To test whether our self-supervised network learns useful features, we compare it with a supervised network and a randomly initialized network, this is also explained in the experiment setup. The next section shows the results of the hyperparameter search for these three settings, and gives the first conclusions about the effectiveness of our network. Lastly, we show the results of two experiments that explore the effect of only using a subset of the labelled data for the downstream task and the effect of the performance of the pretext task. Because we know the rotation experiments should be reproducible we do every experiment first for rotation to make sure our implementation was working as expected.

4.1 EXPERIMENT SETUP

We strive to keep the settings similar throughout the experiments. We use the same architecture for all experiments and use the same values for the hyperparameter searches. Furthermore, all experiments run for 100 epochs and the learning rate is scheduled to be divided by five on the epochs 30, 60, and 80. They can be stopped before reaching epoch 100 if the validation loss does not improve for 35 epochs.

All rotation experiments are run on the CIFAR-10 dataset and use the four rotations that according to Gidaris et al. yield the best result. These rotations are

$$\{0^\circ, 90^\circ, 180^\circ, 270^\circ\}.$$

For the scale experiments we use the EuroSAT dataset. These experiments use the four scales

$$\{1:1, 3:4, 1:2, 1:4\},$$

as explained in the methodology.

4.1.1 *Architecture*

In the paper by Gidaris et al. the architecture is changed between experiments, which makes it harder to compare and understand the results. Here, we use the same architecture for all experiments and downstream models.

We use a network called network-in-network (NIN) that consists of blocks of layers [LCY13]. Based on an experiment from Gidaris et

	Pretext-task		Downstream-task
	Weights	Trainable	Weights
Supervised	random	yes	trained
Self-supervised	trained	no	trained
Random initialization	random	no	trained

Table 1: *An overview of how the weights are trained for the three training settings; supervised, self-supervised, and random initialization.*

al. we use 4 blocks for the pretext task, and the feature vectors are extracted from the end of the second block. For the downstream task we train 2 dense layers with size 200 on the feature vectors. To do so, we lock the weights of the two blocks after training the pretext task and train only the two dense layers for the downstream task.

In order to learn more about the performance of our network, we compare it with a supervised network and a randomly initialized network. The supervised model takes the same architecture of the two blocks and the two dense layers, but it trains all of the layers at once. The random initialization is similar to the self-supervised setting in that the weights of the first two blocks are locked and only the two dense layers are trained during the downstream task. However, the two blocks are not trained during the pretext task, we use the random initialization weights of these layers. This way, we can clearly see if the pretext task feature vectors improved upon the random feature vectors. See Table 1 for an overview of the three training settings.

4.1.2 *Hyperparameter search*

For each experiment we do a hyperparameter search to find the best model. The hyperparameter search is performed on all combinations of the parameters given in Table 2. So, the model is trained 27 times. As mentioned before, the training is only stopped when the validation loss does not improve for 35 epochs.

The search for optimal parameters for the self-supervised network is a bit more challenging. The network consists of 2 networks, the pretext task and downstream task, and ideally we would optimize

Weight decay (WD)	0.1	0.01	0.001
Momentum (M)	0.9	0.09	0.009
Learning rate (LR)	0.1	0.01	0.001

Table 2: *The hyperparameter values used for the hyperparameter search.*

both networks. However it is not trivial to optimise the pretext task, because the best network in terms of accuracy might not produce the best feature vector for the downstream task. So, the ideal parameter search would have to try every combination of parameters for pretext and downstream task. This means the experiment should be run $27 \times 27 = 729$ times, which would take too long to train. Therefore we decided to always use the same parameters for pretext task and downstream task.

4.2 PERFORMANCE OF THE NETWORK

In this section we look at the results of the network and the hyperparameter search. As mentioned in the experiment setup, we perform a hyperparameter search with the parameters in Table 2 for the three training settings in Table 1. For the hyperparameter searches, we only show the combination of parameters that yields the best result. However, we discuss some of the patterns we find when looking at all results. The full overview of these results can be found in the Appendix.

4.2.1 *Rotation*

CIFAR-10 has a defined test set, and from the remaining data we select a random, stratified subset of 20% as validation set. We make a validation set once and use it for all CIFAR-10 experiments. During the hyperparameter search we choose the validation accuracy of the epoch with the lowest validation loss.

In Table 3 the results and chosen hyperparameters are shown. We see that the self-supervised network performs significantly better than the random initialization, so the model learns useful features. Moreover, the validation accuracy of the self-supervised network is close to the supervised network.

The full hyperparameter search overview is shown in Figure 8 in the Appendix. We see that the hyperparameters that give good results for self-supervised learning also give good results for supervised learning.

	WD	M	LR	Validation accuracy
Supervised	0.01	0.9	0.01	87.79
Self-supervised	0.01	0.09	0.1	85.48
Random initialization	0.01	0.9	0.001	70.58

Table 3: *The chosen parameters during the hyperparameter search for rotation, with corresponding validation accuracy. The validation set is 20% from the training set.*

Method	Test Accuracy
Supervised	87.51
Self-supervised	84.97
Random Initialization	69.17
Supervised NIN [GSK18]	92.80
RotNet + conv [GSK18]	91.16
Random Init. + conv [GSK18]	72.50
RotNet + non-linear [GSK18]	89.06

Table 4: *A comparison of the test set accuracy of our network and RotNet on CIFAR-10. Our results are close to the results of Gidaris.*

We use the network weights of the validation experiments to evaluate the test set. Specifically, we use the weights of the epoch with the lowest validation loss. Those results are shown together with the the results from Gidaris et al. in Table 4. We see that all our results are slightly lower than those of the validation set, so our validation set could be easier than the test set. The comparison of Gidaris et al. uses a convolutional neural network for the downstream task, the bottom result has similar architecture to our model. We managed to get close to the results of Gidaris et al., and the difference in performance between self-supervised, supervised, and random is similar. We conclude that our rotation pretext-task performs well and learns useful features for the downstream task.

4.2.2 *Scale*

The EuroSAT dataset does not have a test set, therefore we use 5-fold cross-validation. Here the validation accuracy is taken from the last epoch, and we show the mean and standard deviation. We stop training for a setting if one of the folds has a validation accuracy lower than a set limit. We scale as explained in Section 3.1.3.

The results of the three training settings are given in Table 5. We see that the self-supervised setting performs slightly better than the random initialization setting, however the gap with supervised is a lot bigger. The EuroSAT paper has a classification accuracy of 98.57 [Hel+19]. Note that we cannot compare it directly due to the lack of a test set, but our supervised network gets close to this result.

Another conclusion we take from the result is that all training settings have completely different hyperparameters that give the best result. Moreover, they are also different compared to the best parameters for rotation. The full overview of the hyperparameter search is shown in the Appendix, Figure 9. We see that the pattern of what

				5-fold accuracy	
	WD	M	LR	Mean	STD
Supervised	0.1	0.9	0.001	97.13	0.13
Self-supervised	0.01	0.09	0.01	86.37	0.41
Random initialization	0.001	0.9	0.01	83.71	0.71

Table 5: *Results of the hyperparameter search for EuroSAT.*

parameters work well is a lot more varying than for rotation. For example, when we take the best hyperparameters from self-supervised and look at the supervised result for those, we have an accuracy of only 92 percent. So, the model seems quite sensitive to these hyperparameters.

Moreover, looking at the results, we see that random already performs very well at almost 84 percent, so this dataset might be too easy for self-supervised learning to have impact.

We conclude that scale does not work as well as rotation. However it is still better than random initialization, so it learns some useful features. And it might be partially due to a dataset that is too easy.

4.3 LESS LABELLED DATA FOR THE DOWNSTREAM TASK

One of the main advantages of self-supervised learning is the ability to train the pretext task on a large unlabelled dataset, while using a small labelled dataset to train the downstream task. In this experiment we simulate this situation by training the downstream task on a stratified subset of the dataset. The downstream task and the supervised and random network are trained on several subsets of the dataset, and the validation set stays the same. We train the pretext task on all training data, and it does not see the validation set.

4.3.1 *Rotation*

In this experiment we use the test set as given by the CIFAR-10 dataset to determine the accuracy. Furthermore, for each setting we use the hyperparameters that gave the best result for the full dataset, as shown in Table 3. The subsets we used are

$$\{1, 0.6, 0.2, 0.1, 0.02, 0.004\},$$

where the smallest subset results in 200 training images, so 20 images per class. The results are shown in Figure 6. We see that self-supervised does perform even better than supervised for small datasets used for the downstream task. This demonstrates that the pretext task provides a good feature vector that does not need a lot of human-annotated labels to learn the downstream task.

Annotated label

691	18	54	20	22	4	7	13	124	47
25	770	4	7	5	1	14	5	56	113
102	5	529	51	124	62	68	40	12	7
36	6	76	494	62	184	73	40	16	13
32	4	75	73	605	36	53	96	15	11
14	3	59	168	57	603	26	59	7	4
13	8	44	47	39	24	804	9	9	3
16	3	38	57	84	70	15	680	10	27
104	47	10	10	5	6	5	0	765	48
35	119	6	17	6	5	5	6	42	759

Predicted label

(a) *self-supervised*

Annotated label

575	35	51	35	23	20	14	22	164	61
59	549	8	40	15	11	24	23	63	208
93	28	332	93	149	83	115	48	41	18
62	35	83	262	101	227	124	59	19	28
46	15	144	94	353	62	136	111	25	14
37	14	89	173	92	401	72	84	23	15
26	28	87	57	116	49	574	35	8	20
44	20	43	70	103	98	49	487	16	70
155	75	13	20	11	37	15	14	591	69
66	200	14	36	16	17	25	24	53	549

Predicted label

(b) *supervised*

Table 6: *Confusion matrix for rotation, subset 0.02. The class labels are in alphabetical order.*

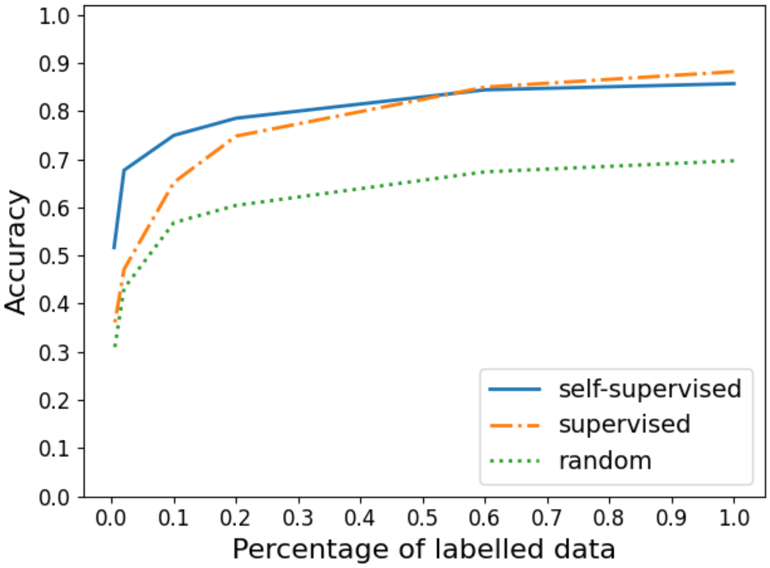


Figure 6: *Less labeled data used for downstream task on rotation.*

Looking at the confusion matrices for subset 0.02 in Table 6, we see the pattern is similar for supervised and self-supervised. For both, images from all classes are most often correctly predicted to be from that class. However, there are some small difference in the performance between classes. For example, the cats prove to be difficult, they are often predicted to be a dog, and vice versa. However, the airplane class, of which the rotation is not always well-defined, has a nice result. So, we can conclude the model has a normal challenge in distinguishing classes that are similar when we do not give it many labels, but it does not seem to have additional issues surrounding rotation.

4.3.2 *Scale*

In this experiment we use a validation set, with 20 percent of the EuroSAT dataset. Furthermore, for each setting we use the hyperparameters that gave the best result for the full dataset, as shown in Table 5. The subsets used are

$$\{1, 0.6, 0.2, 0.1, 0.05, 0.01\},$$

so the smallest subset results in 216 training images with 16-24 images per class. The results are shown in Figure 7. We see the same pattern as with rotation, when subsets get smaller, the accuracy goes down faster for the supervised network. The difference in accuracy is a lot more when using the whole dataset, so it is promising that even here we see a similar result for small datasets. Moreover, we again see that

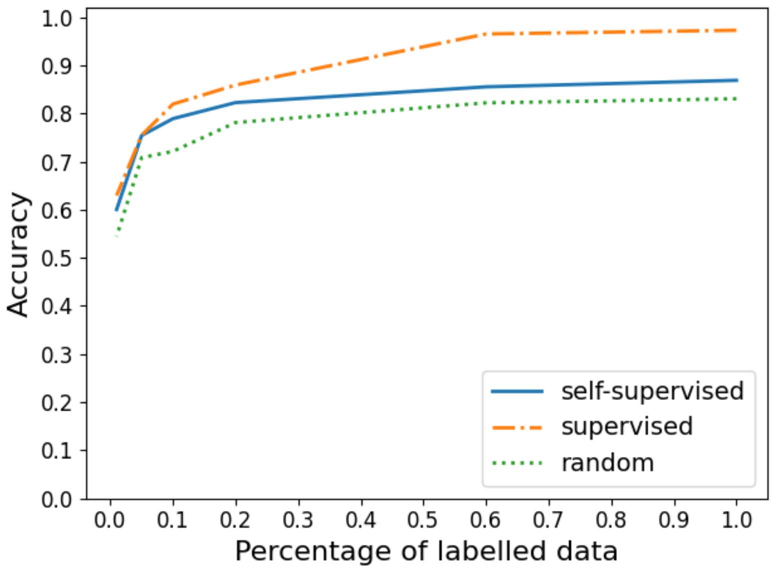


Figure 7: *Less labeled data used for downstream task on scale.*

the result for random stays a lot higher than we saw for rotation, which might confirm the EuroSAT dataset being too easy.

Looking at the confusion matrices in Table 7, we see the supervised and self-supervised model again agree on what classes are more difficult. Although some classes barely suffer from having less labels, for example the *Sea & Lake* class, images from all classes are classified most often as the correct class.

4.4 EFFECT OF PERFORMANCE OF THE PRETEXT TASK

In this experiment we look at the effect of the performance of the pretext task. Does a better result for the pretext task result in better feature vectors, and thus a better result for the downstream task?

We test this by training the downstream task on the feature vector produced by the pretext task during multiple epochs. Gidaris et al. perform this experiment by taking fixed epochs, however we noticed that the validation accuracy of our pretext task model fluctuates too much for this. We take the epochs that give increased performance for the pretext task, and we always include the last epoch. The following pseudocode shows how the epochs are chosen.

```
SET value = 0
FOR each epoch
    IF validation accuracy of this epoch > value
        SET value = validation accuracy of this epoch + 0.02
        ADD epoch to list
    END IF
END FOR
```

Annotated label

461	2	29	23	1	18	37	0	14	15
0	559	6	2	0	29	0	0	0	4
23	6	369	30	15	16	89	39	10	3
38	0	27	227	35	18	56	53	46	0
1	0	1	3	463	0	5	26	1	0
5	15	22	8	0	311	13	6	17	3
34	0	89	24	19	13	288	25	8	0
0	0	3	1	17	0	7	570	2	0
44	18	26	77	10	30	9	17	261	8
6	10	5	0	0	8	0	0	4	567

Predicted label

(a) *self-supervised*

Annotated label

436	3	18	24	2	19	55	1	27	15
0	538	5	1	0	32	0	0	2	22
14	3	418	19	19	20	81	13	8	5
36	1	33	194	38	22	67	46	63	0
0	0	3	7	469	0	4	16	1	0
11	21	24	6	0	274	14	2	45	3
34	0	64	19	9	15	337	12	10	0
0	0	2	8	7	0	5	576	2	0
34	29	28	54	11	65	15	9	250	5
8	7	2	0	0	0	0	0	2	581

Predicted label

(b) *supervised*

Table 7: *Confusion matrix for scale, subset 0.05. The class labels are in alphabetical order.*

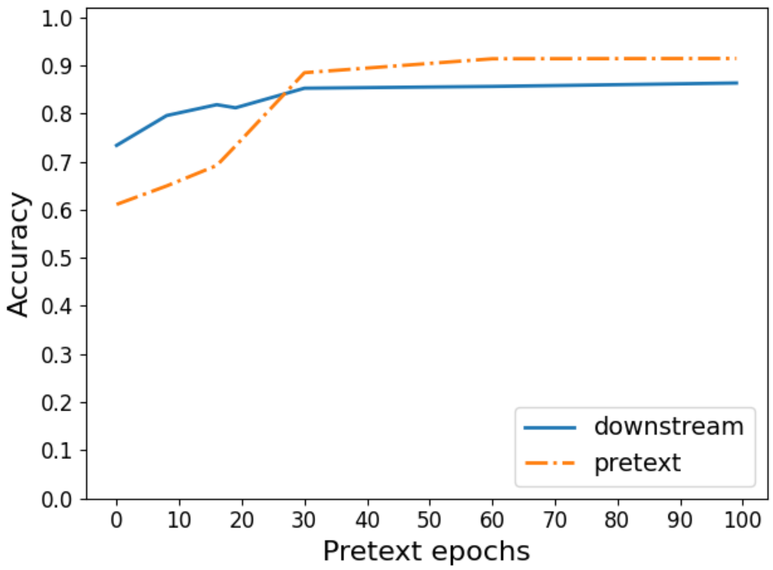


Figure 8: *Effect of the performance of the pretext task on rotation.*

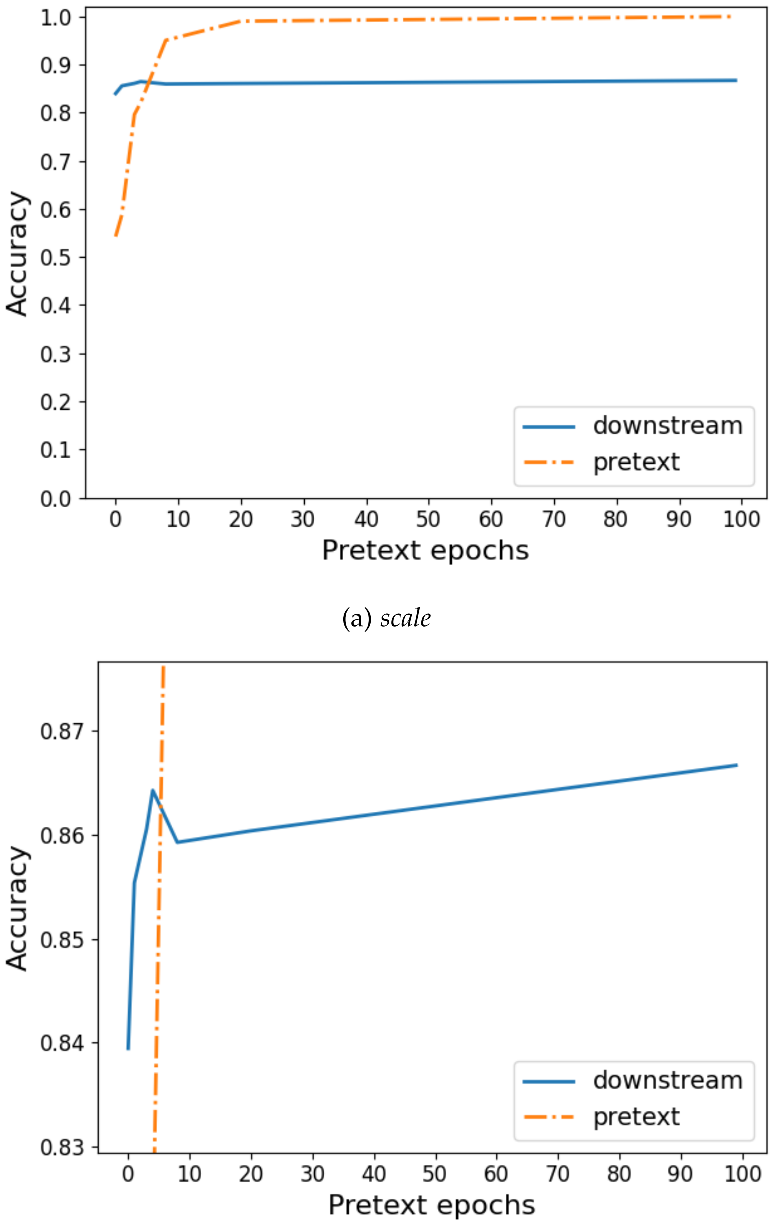
In the results we then plot for each of these epochs the pretext task accuracy, and the corresponding downstream task accuracy. For this experiment we use the hyperparameter values that gave the best result during our hyperparameter search.

4.4.1 *Rotation*

The results for our rotation experiment are shown in Figure 8. For this experiment we use the whole dataset, so the accuracy displayed is on the test set. Moreover, the evaluated pretext task epochs are $\{0, 8, 16, 19, 30, 60, 99\}$. We see the downstream model improves a little bit for improved pretext accuracy. However, random initialization has an accuracy of 69.17 on the test set, so even training the pretext task for a few epochs gives already a better result than random.

4.4.2 *Scale*

For this experiment we use the same validation set as before, using 20 percent of EuroSAT. The evaluated epochs are $\{0, 1, 3, 4, 8, 20, 99\}$. The results are shown in Figure 9a. The downstream task accuracy seems very stable, however we have to keep in mind that the random initialization is already at 83.71 percent. Hence, we cannot make a lot of conclusions from this graph. Therefore, we added a zoomed in graph in Figure 9b. Here, we again see that the first epochs of training give the fastest improvement, and training the pretext task better only slightly improves the downstream results.



(a) *scale*

(b) *zoomed in scale*

Figure 9: *Effect of the performance of the pretext task on scale.*

5 CONCLUSION AND FUTURE WORK

In this chapter we draw conclusions, look at the limitations of our work, and give some ideas for future work.

5.1 CONCLUSION

We looked into self-supervised learning and rebuilt a network based on a rotation pretext task by Gidaris et al. [GSK18]. We were able to reproduce their results and added a scale pretext task to our network. We can see potential for self-supervised learning with scale, but the results are not as good as for rotation. Moreover, we found that scale can only be useful for very specific datasets where all pictures are taken from the same distance. Since the rotation network cannot be used for top-down images, we believe our scale network could still have added value.

5.2 LIMITATIONS

In our implementation we made some choices that could have resulted in a lower result. For example, we limit our hyperparameter search to always use the same value for the pretext task and downstream task, it is possible we could find a better setting without this restriction. Another example of a limiting choice are the parameters for the less labelled data experiments. We use the setting that worked best for the whole dataset. However, a new hyper parameter search might result in other settings that work better for a smaller dataset.

A limitation that we found for self-supervised learning with one geometric transformation in general, is that they only work for specific datatest where the chosen transformation is well-defined.

5.3 FUTURE WORK

In the discussion of our scale experiments we mention the random initialization already gives good results. This could indicate that the EuroSAT dataset is not challenging enough for our self-supervised network to be useful. Therefore, it would be interesting to try our scale network on a different dataset. In addition to datasets with satellite images, we could try a dataset with microscopic images, which are also taken from a set distance. Another thing that we could explore is the effect of a dataset with bigger images. This could also give us the opportunity to experiment with other combinations of scales.

Furthermore, for both rotation and scale we could combine our datasets with a bigger unlabelled dataset. This way we can see if we get the same results as in our simulation with only using a subset of the dataset for the downstream task.

We would also be interested to use the geometric transformation shear, which is a combination of rotation and scale. Since scale does learn some useful features, the combination might improve the results on CIFAR-10.

Lastly, another idea is to combine our work with a paper that builds upon RotNet by Feng et al. [FXT19]. They split the feature vector in two, a rotation related part (RotNet), and a rotation unrelated part. They show improved results, especially for the rotation agnostic images, for example round objects or top-down images. So, maybe their idea could make our model less sensitive to specific datasets.

Part I

APPENDIX

weight	momentum	learning	Validation accuracy		
decay		rate	self-supervised	supervised	random
0.1	0.9	0.1	0.1000	0.1000	0.1000
0.1	0.9	0.01	0.3679	0.7353	0.4087
0.1	0.9	0.001	0.5033	0.8571	0.5854
0.1	0.09	0.1	0.2447	0.1525	0.2165
0.1	0.09	0.01	0.6722	0.8560	0.5766
0.1	0.09	0.001	0.6625	0.7410	0.6439
0.1	0.009	0.1	0.2320	0.1885	0.2393
0.1	0.009	0.01	0.6745	0.8260	0.5627
0.1	0.009	0.001	0.6499	0.7349	0.6422
0.01	0.9	0.1	0.7159	0.7710	0.5562
0.01	0.9	0.01	0.8490	0.8779	0.6840
0.01	0.9	0.001	0.6851	0.7315	0.7058
0.01	0.09	0.1	0.8548	0.8691	0.3821
0.01	0.09	0.01	0.7018	0.7433	0.6937
0.01	0.09	0.001	0.6342	0.6792	0.6487
0.01	0.009	0.1	0.8522	0.8671	0.6591
0.01	0.009	0.01	0.6982	0.7394	0.6876
0.01	0.009	0.001	0.6342	0.6721	0.6383
0.001	0.9	0.1	0.8436	0.8570	0.6376
0.001	0.9	0.01	0.7648	0.8060	0.6569
0.001	0.9	0.001	0.6748	0.7161	0.6784
0.001	0.09	0.1	0.7747	0.8191	0.6622
0.001	0.09	0.01	0.6826	0.7239	0.6746
0.001	0.09	0.001	0.6277	0.6774	0.6417
0.001	0.009	0.1	0.7684	0.8188	0.6266
0.001	0.009	0.01	0.6745	0.7148	0.6780
0.001	0.009	0.001	0.6232	0.6693	0.6309

Table 8: *Hyperparameter search results for rotation on CIFAR-10. The chosen settings are highlighted.*

weight	momentum	learning	Validation accuracy		
decay		rate	self-supervised	supervised	random
0.1	0.9	0.1	0.1111	0.1111	0.1111
0.1	0.9	0.01	0.1111	0.1111	0.3763
0.1	0.9	0.001	0.3789	0.9713	0.6137
0.1	0.09	0.1	0.1111	0.1111	0.287
0.1	0.09	0.01	0.4393	0.97	0.6387
0.1	0.09	0.001	0.8451	0.9217	0.8005
0.1	0.009	0.1	0.1111	0.1111	0.2613
0.1	0.009	0.01	0.4761	0.9711	0.6419
0.1	0.009	0.001	0.8441	0.9152	0.7874
0.01	0.9	0.1	0.1111	0.1111	0.3333
0.01	0.9	0.01	0.6722	0.9652	0.7963
0.01	0.9	0.001	0.8594	0.9246	0.8325
0.01	0.09	0.1	0.7130	0.961	0.7015
0.01	0.09	0.01	0.8637	0.9243	0.8268
0.01	0.09	0.001	0.8369	0.8817	0.8124
0.01	0.009	0.1	0.6950	0.9614	0.6494
0.01	0.009	0.01	0.8608	0.9226	0.8269
0.01	0.009	0.001	0.8429	0.8728	0.8114
0.001	0.9	0.1	0.6967	0.9543	0.5189
0.001	0.9	0.01	0.8484	0.953	0.8371
0.001	0.9	0.001	0.8510	0.9063	0.836
0.001	0.09	0.1	0.8352	0.9539	0.8203
0.001	0.09	0.01	0.8585	0.9046	0.8321
0.001	0.09	0.001	0.8416	0.8767	0.8159
0.001	0.009	0.1	0.8462	0.9511	0.8275
0.001	0.009	0.01	0.8578	0.9015	0.8322
0.001	0.009	0.001	0.8385	0.8691	0.8151

Table 9: *Hyperparameter search results for scale on EuroSAT. The chosen settings are highlighted. The validation accuracy displayed is the mean of 1-5 folds. When the accuracy is lower than certain value, we do not train more folds. This value is 0.95 for supervised, 0.84 for self-supervised, and 0.8 for random.*

BIBLIOGRAPHY

[Ahm+08] Amr Ahmed, Kai Yu, Wei Xu, Yihong Gong, and Eric Xing. "Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks." In: *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III 10*. Springer. 2008, pp. 69–82.

[Che+19] Liang Chen, Paul Bentley, Kensaku Mori, Kazunari Misawa, Michitaka Fujiwara, and Daniel Rueckert. "Self-supervised learning for medical image analysis using image context restoration." In: *Medical image analysis* 58 (2019), p. 101539.

[Che+20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations." In: *arXiv preprint arXiv:2002.05709* (2020).

[Den+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

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

[Dos+14] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. "Discriminative unsupervised feature learning with convolutional neural networks." In: *Advances in neural information processing systems*. 2014, pp. 766–774.

[FXT19] Zeyu Feng, Chang Xu, and Dacheng Tao. "Self-supervised representation learning by rotation feature decoupling." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10364–10374.

[GSK18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised representation learning by predicting image rotations." In: *arXiv preprint arXiv:1803.07728* (2018).

[Goo+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[Gui+23] Jie Gui, Tuo Chen, Qiong Cao, Zhenan Sun, Hao Luo, and Dacheng Tao. "A Survey of Self-Supervised Learning from Multiple Perspectives: Algorithms, Theory, Applications and Future Trends." In: *arXiv preprint arXiv:2301.05712* (2023).

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[Hel+19] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification." In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.7 (2019), pp. 2217–2226.

[JF18] Simon Jenni and Paolo Favaro. "Self-supervised feature learning by learning to spot artifacts." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2733–2742.

[JT20] Longlong Jing and Yingli Tian. "Self-supervised visual feature learning with deep neural networks: A survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[Kar+20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. "Analyzing and improving the image quality of stylegan." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8110–8119.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images." In: (2009).

[LMS17] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. "Colorization as a proxy task for visual understanding." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6874–6883.

[LeC+98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[LCY13] Min Lin, Qiang Chen, and Shuicheng Yan. "Network in network." In: *arXiv preprint arXiv:1312.4400* (2013).

[NF16] Mehdi Noroozi and Paolo Favaro. "Unsupervised learning of visual representations by solving jigsaw puzzles." In: *European Conference on Computer Vision*. Springer. 2016, pp. 69–84.

[Nor+18] Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, and Hamed Pirsiavash. "Boosting self-supervised learning via knowledge transfer." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9359–9367.

[PY10] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning." In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[Pat+16] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. "Context encoders: Feature learning by inpainting." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544.

[PG20] Senthil Purushwalkam and Abhinav Gupta. "Demystifying Contrastive Self-Supervised Learning: Invariances, Augmentations and Dataset Biases." In: *arXiv preprint arXiv:2007.13916* (2020).

[QS22] Guo-Jun Qi and Mubarak Shah. "Adversarial Pretraining of Self-Supervised Deep Networks: Past, Present and Future." In: *arXiv preprint arXiv:2210.13463* (2022).

[Ran+07] Marc'Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. "Unsupervised learning of invariant feature hierarchies with applications to object recognition." In: *2007 IEEE conference on computer vision and pattern recognition*. IEEE. 2007, pp. 1–8.

[Tre+22] Sandra Treneska, Eftim Zdravevski, Ivan Miguel Pires, Petre Lameski, and Sonja Gievska. "GAN-Based image colorization for self-supervised visual feature learning." In: *Sensors* 22.4 (2022), p. 1599.

[ZZC20] Kaige Zhang, Yingtao Zhang, and HD Cheng. "Self-supervised structure learning for crack detection based on cycle-consistent generative adversarial networks." In: *Journal of Computing in Civil Engineering* 34.3 (2020), p. 04020004.

[ZIE16] Richard Zhang, Phillip Isola, and Alexei A Efros. "Colorful image colorization." In: *European conference on computer vision*. Springer. 2016, pp. 649–666.