Reproducing Self-Supervised Learning dynamics without contrastive pairs

Dennis Bogatov Wilkman
dwilkman@kth.se

Agnieszka Miszkurka
agnmis@kth.se

Tobias Höppe
thoppe@kth.se

Abstract

Self-Supervised learning without contrastive pairs has shown huge success in the recent year. However, understanding why these networks do not collapse despite not using contrastive pairs was not fully understood until very recently [1]. In this work we re-implemented the architectures and pre-training schemes of *SimSiam*, *BYOL*, *DirectPred* and *DirectCopy*. We investigated the eigenspace alignment hypothesis in *DirectPred*, by plotting the eigenvalues and eigenspace alignments for both *SimSiam* and *BYOL* with and without Symmetric regularization. We also combine the framework of *DirectPred* with SimCLRv2 in order to explore if any further improvements could be made. We managed to achieve comparable results to the paper of *DirectPred* in regards to accuracy and the behaviour of symmetry and eigenspace alignment. We release our code [1].

1 Introduction

Self-Supervised learning has become an important task in many domains, since labeled data is often rare and expensive to get. Many modern methods of Self-Supervised learning are based on Siamese-networks [2] which are weight sharing Neural networks for two or more inputs which representations then will be compared in latent space. The representation created by this approach can then be used for classification by fine-tuning on fewer labelled data-points. Traditionally, during pre-training positive pairs (same image, or two images from the same class) and negative pairs (different images or two images from a different class) are used. The distance of the representation of positive pairs is minimized while the distance of the representation of negative pairs is maximized, which prevents the networks from collapse (i.e mapping all inputs to the same representation). These methods have shown quite some success in the past [3], [4], [5], [6]. However, these methods rely on negative pairs, and large batch sizes which makes the training less feasible.

Recently, new methods have been proposed which rely only on positive pairs and yet don't collapse [7], [8]. In the paper "Understanding Self-Supervised Learning Dynamics without Contrastive Pairs" by Tian et.al. [1] the underlying dynamics are explored and based on the theoretical results, a new method, *DirectPred*, was proposed which does not need an update of the predictor via gradient descent but instead is set directly each iteration. These method has then been simplified with *DirectCopy* in [9] achieving similar performance.

The focus of this work is to test several assumptions made in [1] for the theoretical analysis and see if they hold. For this, we will concentrate especially on the eigenvalues of the predictor network and the eigenspace alignment with its input. Also, we will reproduce the results from [1], [9] [7] and [8] on CIFAR-10 to compare their learned representation via linear probes. In addition we will combine *DirectPred* with the method proposed in [8] and use a deeper projection head, as well as keeping some layers for fine tuning, in order to test if an increase in classification performance can be achieved, as reported by Chen et.al. [4].

[1] https://github.com/miszkur/SelfSupervisedLearning

Project report for DD2412 or FDD3412

2 Related work

A common approach to representation learning without Siamese networks is generative modelling. Typically these methods model a distribution over the data and a latent space, from which then embeddings can be drawn as data representations. Usually these approaches rely on Auto-encoding [10, 11] or Adversarial networks [12, 13]. However, generative models are often computatinaly heavy and hard to train.

Discriminative methods using Siamese networks like SimCLR [3, 4] and Moco [5] outperform generative models and have lower computational cost. However, these methods rely on very large batch sizes since they use contrastive pairs. Most recent methods, replicated in this work, like *BYOL* [7] and *SimSiam* [8], only rely on positive pairs and therefore can make use of smaller batch sizes. To understand why these methods do not collapse, the dynamics of these networks are analysed with linear models in [1, 9]. From this analysis, the authors could derive ablations of *BYOL* where part of the network is directly set to its optimal solution instead of being trained by gradient descent.

3 Method

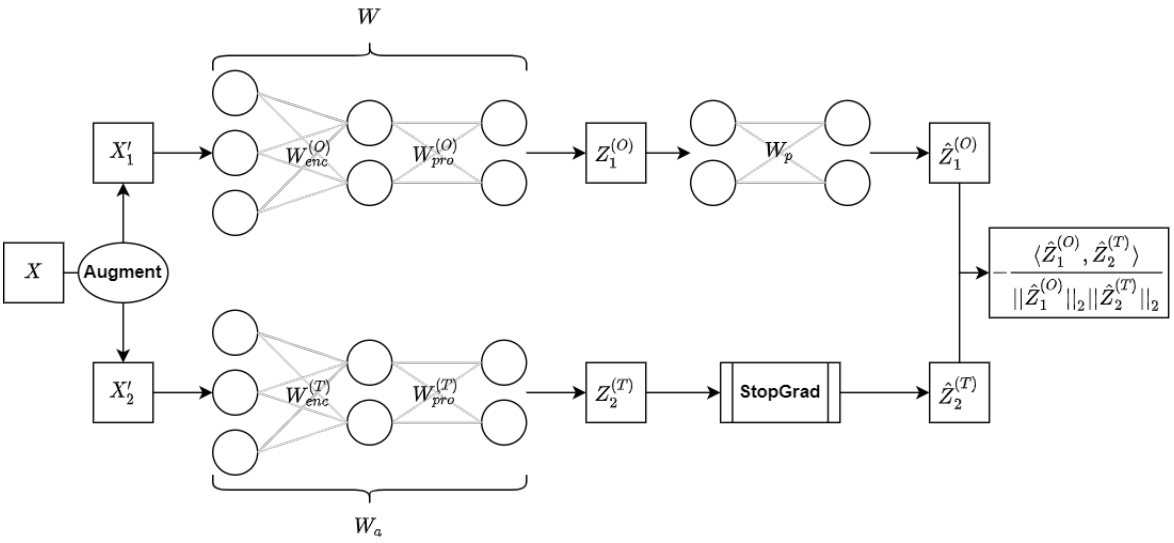

Figure 1: Network architecture for all presented methods

In this section we will describe the methods of *BYOL* and *SimSiam* as well as its successors *DirectPred* and *DirectCopy*.

3.1 BYOL & SimSiam

The network architecture of the models is shown in Figure 1. First, two augmented views X_1' and X_2' of an image X are created and fed into the online network W and target network W_a respectively. Both of these networks have the same architecture, a ResNet-18 (W_{enc}^x) as encoder [14], which is supposed to create hidden features and a projector head W_{pro}^x, which is a two layer MLP, with purpose to map the feature space into a lower dimensional hidden space. The online network also has an additional predictor head, again consisting of a two layer MLP. The target network has a *StopGrad* function instead of a predictor head. Therefore during back propagation, only the weights of the online network are updated. The loss between the output of the online and target network is equal to the cosine-similarity loss function.

$$\mathcal{L}(\hat{Z}_1^{(O)}, \hat{Z}_2^{(T)}) = -\frac{\langle Z_1', Z_2' \rangle}{||Z_1'||_2 ||Z_2'||_2} \tag{1}$$

Note, that the final loss of one image is the symmetric loss $\mathcal{L}(\hat{Z}_1^{(O)}, \hat{Z}_2^{(T)}) + \mathcal{L}(\hat{Z}_2^{(O)}, \hat{Z}_1^{(T)})$, since each augmentation is given to both networks. As mentioned, the target network is not updated with

gradient descent, but with an exponential moving average (EMA). After each batch the target network will be set to $W_a = W_a + (1 - \tau)(W - W_a)$. In *SimSiam* the target network is set directly to the online network after each update, i.e $\tau = 0$.

3.2 DirectPred & DirectCopy

[1] and [9] derive a one layer linear predictor analytically with the analysis of the underlying dynamics of these models presented in Section 3.1 with an approximation of the actual network as a purely linear model. In addition to the linearity of the encoder and predictor, three additional simplifying assumptions where made:

- The target network is always in a linear relationship with the online network (e.g. $W_a(t) = \tau(t)W(t)$
- The original data distribution $p(X)$ is Isotropic and its augmentation $\hat{p}(X'|X)$ has mean X and covariance $\sigma\mathbf{I}$
- The predictor W_p is symmetric

Based on these assumptions, one can show, that the eigenspaces of the output of the online network and the predictor W_p align. Let $F = WXW^\top$ (i.e output of the online network as linear model), then it follows with the three assumptions, that the eigenspaces of these two matrices allgin over time (e.g. for all non-zero eigenvalues λ_{W_p}, λ_F of W_p and F, the corresponding normalized eigenvectors v_{W_p}, v_F are parallel, $v_{W_p}^\top v_F = 1$). Therefore, we can derive an analytical expression for the predictor W_p. Let $F = U\Omega U^\top$ be the eigen-decomposition of F with $\Omega = \text{diag}(\lambda_F^{(1)}, ..., \lambda_F^{(d)})$ the diagonal matrix with the eigenvalues of F, then we can approximate the eigenvalues of W_p with

$$\lambda_{W_p}^{(j)} = \sqrt{\lambda_F^{(j)}} + \epsilon \max_j \lambda_F^{(j)} \tag{2}$$

and therefore set W_p to

$$W_p = U\text{diag}(\lambda_{W_p}^{(1)}, ..., \lambda_{W_p}^{(d)})U^\top \tag{3}$$

Note, that we cannot compute F directly, which is why we use a running average $\hat{F}$ as approximation in practice

$$\hat{F} = \rho\hat{F} + (1 - \rho)\hat{Z} \tag{4}$$

where $\hat{Z} = \hat{Z}_1^{(O)}\hat{Z}_2^{(O)\top}$.

In [9] a more general and simpler version of *DirectPred* is analyzed and *DirectCopy* is proposed. In *DirectCopy* the predictor is set directly to the normalized moving average of the projector's output, i.e:

$$W_p = \frac{F}{||F||} + \epsilon\mathbf{I} \tag{5}$$

This method omits the eigen-decomposition, which makes it cheaper in terms of computation while achieving similar results to *DirectPred*.

4 Data & Configurations

All experiments are conducted on CIFAR-10 [15], which contains 60 000 RGB images uniformly distributed over 10 classes. The pre-training and the finetuning are done on the entire training set, which consists of 50 000 images. For finetuning only a linear layer is used on top of the encoder, where the weights of the encoder are frozen (I.E. we test linear separability of the encoders output). The reported results are produced from a test set containing 10 000 images. Also, to account for the small dimension of the CIFAR-10 images ($32 \times 32 \times 3$) we use 3×3 convolutions and stride 1 without maximum pooling in the first block of the encoder.

To augment each image, we first do a random flip, take a random crop (up to 8% of the original size) of the image. Then we randomly adjust brightness, saturation, contrast and hue of the RGB image by a random factor [2]. Finally with a 20% chance we convert the image to grey scale.

[2] for brightness, saturation and contrast we chose a value uniformly at random between 0.6 and 1.4. For adjusting the hue, we set the maximal value to 0.1

Self-supervised pretraining In the basic setting, the online network use ResNet-18 as encoder, two layer projector MLP, two layer predictor MLP, where the first layer consists of 512 nodes, followed by BatchNorm and ReLU, and then a linear output layer with 128 nodes. For *BYOL* we use EMA to update target network and for *SimSiam* we directly set encoder and projector of target network to the weights of the online one ($\tau = 0$). We use SGD optimizer with learning rate 0.03, momentum 0.9 and weight decay (L2 penalty) of 0.0004. The predictor of *DirectPred* and *DirectCopy* are set directly and are not trained with gradient descent and consist of one linear layer with 128 nodes. By SGD baseline for those methods we mean a network pretrained with a one linear layer predictor with or without EMA. In all experiments, we use batch size of 128. For updating the target network we used the EMA parameter $\tau = 0.996$. For *DirectPred* we use $\epsilon = 0.1$ and $\rho = 0.3$ and for *DirectCopy* $\epsilon = 0.3$ and $\rho = 0.5$.

Linear evaluation In order to test the performance of the different models, we use linear evaluation, i.e we train a linear layer on top of the ResNet-18 with frozen weights for 100 epochs. This measures how linearly separable the learned representations are. We use Adam optimizer [16] with polynomial decay of learning rate from 5e-2 to 5e-4. Images are normalized but we do not use augmentation for this part of training just as in the original repository for *DirectPred*.

5 Experiments and findings

In this section, we will first show that the assumptions and theoretical findings from Section 3.2 hold in practice. Finally, we will pre-train and finetune the different models presented in Section 3 and test their performance.

5.1 Eigenspace alignment

First, we pre-train *BYOL* and *SimSiam* keep track of the predictor heads symmetry and eigenspace alignment. In Figure 2 we can see, that the assumption of an symmetric predictor W_p holds. Even without symmetry regularisation, W_p approaches symmetry during training. Also, we can see that for all non-zero eigenvalues of W_p the eigenspaces between F and W_p align as the training progresses.

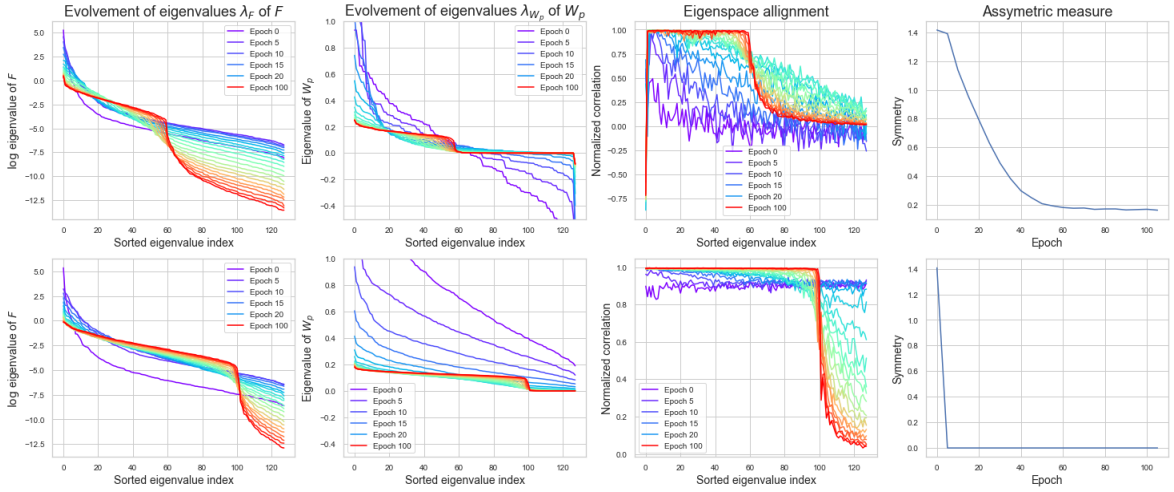

Figure 2: Pre-training *BYOL* for 100 epochs of CIFAR-10. **Top row:** *BYOL* without symmetry regularisation on W_p. **Bottom row:** *BYOL* with symmetry regularisation on W_p. The eigenvalues of F are plotted on the log scale, since the eigenvalues vary a lot.

We ran the same Experiment for *SimSiam*, and can also see the same effect on the predictor and the alignment (Figure 3). If we don't use a symmetric predictor, we also see that the eigenspaces for the non-zero eigenvalues align. However, once we use symmetry regularisation on W_p, all eigenvalues become zero, which shows that the network collapses.

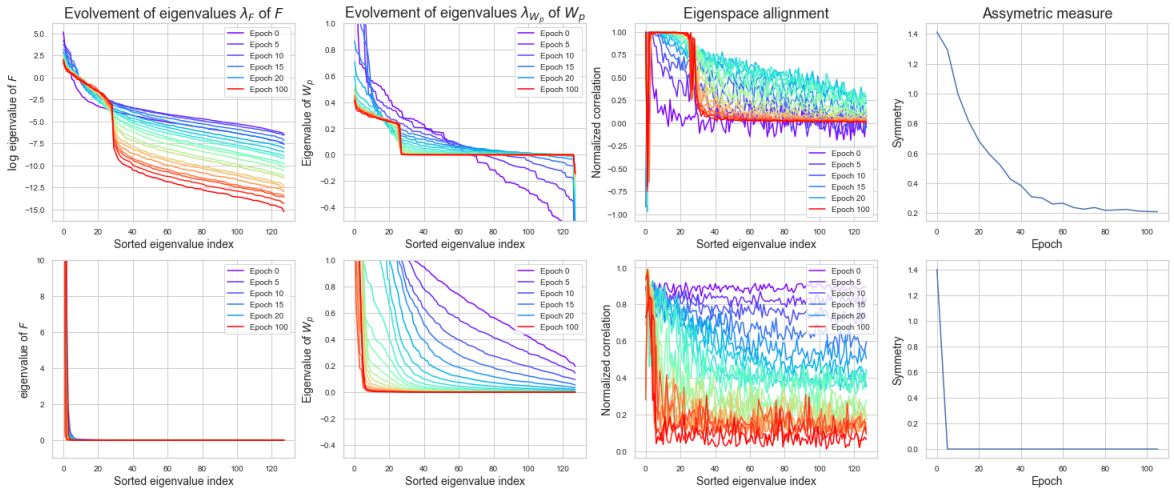

Figure 3: Pre-training *SimSiam* for 100 epochs of CIFAR-10. **Top row:** *SimSiam* without symmetry regularisation on W_p. **Bottom row:** *SimSiam* with symmetry regularisation on W_p. Note that the eigenvalues of F are not plotted on the log scale here, since we get 0 values.

We can prevent the collapse of *SimSiam* with symmetric predictor by choosing very large and different learning rates α, α_p for W and W_p, as well as using different weight decay η, η_p for W and W_p Figure 4. The predictor has to have a higher learning rate in order to successfully remove the target network. This suggests that EMA brings some stability to the learning dynamics of the network [1], [7].

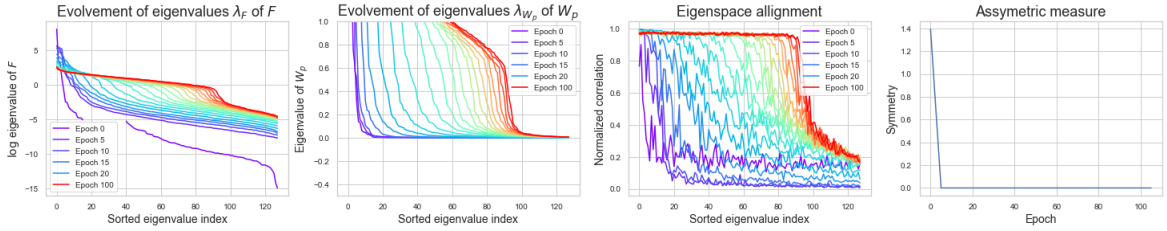

Figure 4: *SimSiam* with symmetric predictor but learning rates $\alpha = 0.2$, $\alpha_p = 2$ and weight decay $\eta = 0$, $\eta_p = 4e - 4$

Eigenvalues From these plots we can already approximate the relative performance of the networks. If the predictor has less 0 eigenvalues, the matrix W_p has less singularities i.e less eigenvectors which correspond to the nullspace of the matrix. Therefore, its output W_pF has higher rank, which is desired in representation learning, as it enables a more separable representation.

5.2 Performance

Byol & SimSiam In table 1 we can see that the performance of *BYOL* increases slightly when using symmetry regularisation on the predictor. However, as already seen in Figure 3, when using no EMA, we observe that the network collapses. As shown in Figure 4, we can prevent this by adjusting the learning rates α and α_p as well as the weight decay for the different parts of the model η, η_p. This leads to a performance of 79.2 %. Also, we observe in general better performance for models trained with EMA, given the same hyperarameters. However, we did not use extensive hyperparameter tuning, as performance is not the focus of our work.

DirectPred & DirectCopy As we can see in Figure 2 & 3, the eigenspaces for both models align and therefore the theoretical assumptions of [1, 9] hold. As we can see in Figure 5, all models perform reasonably well, and can achieve almost the same performance as *BYOL* or *SimSiam*. However, as

	symmetric W_p	non symmetric W_p
EMA	**85.7**	84.2
No EMA	20.3	**79.4**

Table 1: Comparision of a two layer predictor with and without symmetry regularisation as well as with and without EMA (i.e first row is *BYOL* and second row is *SimSiam*).

already mentioned earlier, we can see that models with EMA outperform models without EMA in every setting. Also *DirectPred* and *DirectCopy* achieve almost the same accuracy, in [9] it is stated that *DirectCopy* can outperform *DirectPred* when pre-trained for more epochs. We did not replicate this experiment, due to computational constraints.

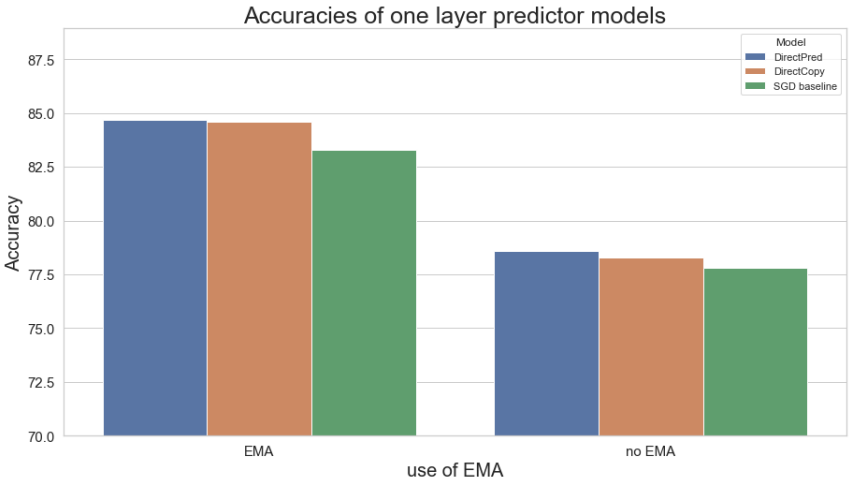

Figure 5: The accuracy of *DirectPred* and *DirectCopy* compared to their SGD baselines with and without EMA

Deeper Projection Head In [13] a deeper projection head was proposed to increase accuracy. In addition a reuse of the first projector layer for finetuning was also suggested. We attached a deeper projection head to our *DirectPred* model trained with EMA. However, we could not increase performance. Actually we could observe a clear decrease and also an even strongrer decrease if the projector was reused.

re-use projector	only encoder
80.7	82.3

Table 2: Performance of two *DirectPred* models which where pre-trained with a deeper projection head. Re-using part of the projector for finetuning did decrease performance most.

6 Challenges

Thanks to the detailed description of *BYOL* by Grill et. al. [7] we were able to reproduce the paper achieving similar results as the authors. Due to time constraints we decided to use CIFAR-10 instead of STL-10 which was used in most of the experiments in the reproduced paper.

Overall the main challenge was the large amount of computations required for all the experiments, it took around 4 hours and 30 minutes to pre-train and fine tune a single model, and in total we trained for around 100+ hours due to bug fixing and testing various schemes and hyper parameters.

Due to some missing details in [1] we had to check the original repository, which was written in PyTorch. Which brought another challenge as there are differences in TensorFlow and PyTorch libraries. Example being, in PyTorch one of the parameters of the SGD optimizer is weight decay (L2 penalty), in TensorFlow we had to implement it by hand as TensorTlow's SGDW implements

Decoupled Weight Decay Regularization [17]. Image augmentation methods such as ColorJitter from PyTorch do not have exact corresponding method. For example, adjusting brightness works differently so we defined our custom way to do it so that augmentations are as close as possible to the original version.

7 Conclusion

We could successfully re-implement several methods for unsupervised representation learning without contrastive pairs, namely *BYOL*, *SimSiam*, *DirectPred* and *DirectCopy*. Our experimental results aligned well with both the theoretical analysis about the eigenspaces made in [1] and the symmetric assumptions. We achieved comparable behaviour with regards to eigenspace alignment, symmetry. But we cannot report that *DirectPred* or *DirectCopy* could outperform their two layer opponent with or without EMA. However, we compared these models on CIFAR-10, whereas in [1] the experiments where run on STL-10.

This leaves us with the conclusion, that *DirectPred* and *DirectCopy* give valuable insights into the dynamics of unsupervised representation learning without contrastive pairs, but do not necessarily build new state of the art models themselves.

8 Ethical consideration, societal impact, alignment with UN SDG targets

Self-supervised learning circumvents label scarcity which is one of the most common problems when applying ML to new scenarios. This can have both positive and negative consequences. On one hand, it can accelerate important developments for example in medical diagnosis. However, it can also be used in unethical ways such as in surveillance or military equipment. Furthermore, there will be less need for people labelling datasets which will result in reduction of job positions in this area.

9 Self Assessment

We think that our project should be graded with an A for four main reasons:

- We have **successfully implemented** *BYOL* [7] from scratch and its ablations:
  - SimSiam [8]
  - DirectPred [1]
  - DirectCopy [9]

  BYOL requires implementing Exponential Moving Average and *DirectPred* requires more in depth theory compared to *SimSiam* which was an alternative paper for the project. Moreover, we achieved **results matching the ones in the reproduced paper** and observed the same behaviour of the networks when investigating eigenspaces and symmetry.
- **Reimplementation in a deep learning framework for which an online public repository is not available**: the original repository [18], is written in PyTorch. As we used TensorFlow, this added an extra layer of complexity.
- **New combination with other papers**, i.e. combining *DirectPred* with SimCLRv2 framework [4]. We implemented deeper projection head which is saved and used in fine-tuning the model and successfully investigated the performance impact of this ablation.
- **Comparison with other relevant methods**: we compared *DirectPred* to recently proposed *DirectCopy*.

We achieved all the success measures from our project proposal and in addition experimented with influence of the learning rate and weight decay on stability of the training for *SimSiam* (4).

References

[1] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs, 2021.

[2] Jane Bromley, James Bentz, Leon Bottou, Isabelle Guyon, Yann Lecun, Cliff Moore, Eduard Sackinger, and Rookpak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:25, 08 1993.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[4] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners, 2020.

[5] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020.

[6] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

[7] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020.

[8] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning, 2020.

[9] Xiang Wang, Xinlei Chen, Simon S. Du, and Yuandong Tian. Towards demystifying representation learning with non-contrastive self-supervision, 2021.

[10] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[11] Pascal Vincent, Hugo Larochelle, Y. Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. pages 1096–1103, 01 2008.

[12] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning, 2017.

[13] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning, 2019.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[15] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10.

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

[17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[18] Papers with code. `https://paperswithcode.com/paper/understanding-self-supervised-learning`. Accessed: 2020-10-22.