Reimplementation of Self-Classifier on new datasets

Shaochen Bai
shaochen@kth.se

David Bardvall
bardvall@kth.se

Riccardo Periotto
periotto@kth.se

Abstract

In this work, we re-implement "Self-Classifier" [1], a novel self-supervised end-to-end classification learning approach presented in ECCV 2022. In addition to achieving better performance than its competitors over the self-supervised challenges, the approach guarantees important properties such as scalability, single-stage end-to-end training, and non-degenerate solution. We apply the proposed method to MNIST, CIFAR-10, CIFAR-20, and STL-10 in the evaluation of unsupervised clustering performance. Standard linear probing experiments and ablation studies are run to examine different aspects of the model. We also extend the visualization section to analyze the representation learning quality and prior enforcement situation. Although the mathematical formulations of the model as well as its implementation details are mostly covered in the original paper, additional code resources are referred to set the hyperparameters properly. Our code is available on KTH GitHub: [1].

Keywords: Self-Supervised Learning, Representation Learning, Clustering, Non-contrastive Learning

1 Introduction

Many of the most significant results achieved so far by deep learning algorithms come from supervised settings with access to large amounts of labeled data. Such large-scale labeled datasets are not always guaranteed, especially in the field where human labeling is expensive or restricted (medical field for example). On the other hand, with the development of big data, unlabeled data are easier to obtain through the Internet more than ever, and the idea of unsupervised learning is to train the model without supervised knowledge but achieve rather competitive performance nonetheless. The label-free technique greatly expands the possible application fields of deep learning and increases its influence in every modern industry.

There are many subcategories in the field of unsupervised learning, and the proposed paper [1] focuses on self-supervised learning aiming to train the model to obtain high-level natural features in the data by solving one or a set of pretext tasks. After learning to recognize these features, the model can either be used directly for representation retrieving or fine-tuned to solve task-specific downstream tasks [1][5] [8][13]. The main focus of the literature on self-supervised learning is to understand and develop the most suitable pretext tasks for some specific target tasks. At the moment, contrastive learning is the pretext task defining the state-of-the-art results [13][7], the idea of which is to train the model to predict the similarity (or the distance) between two augmented inputs. However, current contrastive learning methods are prone to degenerate solutions, where the models map all the inputs into a single representation. Targeting the problem, the paper proposed a loss function that enforces a uniform prior to all the classes during training which avoids the pitfall and learns to cluster data concurrently. From a general perspective, the prominent characteristics of Self-Classifier also include its strong performance, high scalability, and simple single-stage end-to-end training structure.

[1] https://github.com/riccardoperiotto/KTH-DD2412-DeepLearningAdvanced

The main contributions of our work are that:

• We implement and compare Self-Classifier with its competitors on smaller datasets CIFAR10, CIFAR20, MNIST, and STL10;

• We broadly study the related literature and conclude the main differences;

• Extensive visualization and ablation study are run to deepen our understanding of the model.

2 Related Work

2.1 Contrastive Learning

It is well worth noticing that contrastive learning is just one of the proposed pretext tasks for self-supervised learning. Outside of this category, many non-contrastive tasks have also demonstrated their validity, some of the most popular among which include colorization, jigsaw puzzle, image inpainting, relative patch prediction, context prediction, and rotation prediction[1][13]. The key difference between contrastive and non-contrastive learning lies in the self-generated label that the model aims to predict. In non-contrastive learning, labels correspond to the augmentation applied to the inputs by the model. While in contrastive learning, labels act as a distance metric between the model's predictions of two different input augmentations. The goal here is to train the model to learn a feature representation that maximizes the distances between augmentation of different inputs and minimizes those between augmentations of the same data point [13].

Within the specific category of contrastive learning, there is an additional distinction between models trained with negative and positive pairs and those trained using only the positive ones. While both methods can successfully be trained to obtain optimal results [7], the underlying pretext task can sometimes be non-aligned with the downstream one, as described in [1]. Also as negative samples act as a means to avoid degenerate solutions, models training only on positive pairs are more prone to the problem and additional care needs to be taken.

2.2 Downstream Tasks

Depending on the learning goals of subsequent downstream tasks, we can split the compared methods into two categories: those that only aim to solve representation learning and those that combine, in some way, representation learning and clustering. We highlight that both categories can solve the two tasks because: (i) It is always possible to apply additional clustering on the features obtained from a model of the first category; (ii) It is always possible to access the most informative features by retrieving representations from the backbone of a second category model.

Regarding the first category, earlier models include Exemplar CNN [11] and Non-Parametric Instance Discrimination (NPID) [20], which suffer limitations such as poor scalability and lack of consistency across representations stored in the memory bank [1]. More recent methods belonging to the category are MoCo [14], BOYL [13], SimSiam [8] and DINO [6], which all adopt certain stop gradient operations to avoid the degenerative solution, regarding which constitutes the key difference between them and Self-Classifier. We will discuss this point thoroughly in the next section (section 3).

The second category methods mostly include the alternative clustering phase along with the representation learning, which results in the earlier models such as DEC [21], JULE [22] hard to be tested on a larger-scale dataset for their computation inefficiency. DeepCluster works better but still requires a costly clustering phase and specific precautions to avoid collapsing to trivial solutions [13]. More recent methods such as SwAV [5], SeLa [2], SCAN [19] and IIC [16] improves efficiency and stability at the same time, and are easier to compare with our model for their similar architecture.

What all the modern models from both categories have in common is that they are trained using only positive pairs (except MoCo), which as anticipated, makes them all prone to the degenerative solution. The main differences they have from one another consist of the various strategies implemented to avoid this problem. As in [4], we underline that the existence of the degenerative solution is not specific to the unsupervised training, but it is common to any method that jointly learns a discriminative classifier and the labels.

3 Methods

Although the former section covers the literature in general, we would discuss the key design aspects in more detail, and demonstrate the method of Self-classifier by comparing it with related models with a clear focus on the similarities and differences in how they avoid the degenerate solution.

3.1 Avoid the degenerate solution

For the self-supervised techniques that do not rely on negative samples, we split them into three categories, namely external clustering, momentum encoder, and non-collapsing function. Since the principle of avoiding the degenerate solution behind the first two categories is practically the same, we further identify them together as stop-gradient operations, whose distinguishment is where the stop operation is applied.

External clustering is employed in models that simultaneously learn how to represent and classify the data. We mention that the models solving these tasks together most similar to Self-Classifier are SwAV [5], SeLa [2] and SCAN [19]. The difference between these three concerning external clustering is that SwAV and SeLa implement it online through a Sinkhorn-Knopp distance algorithm, while SCAN implements it separately using K-means [1].

MoCo [14], BYOL [13], and DINO [6] do not have a clustering phase and the stop gradient operation is obtained through a second network called momentum encoder. This idea originates from MoCo [14], but despite acting as a precursor, this model is slightly different from the others as its training phase employs both positive and negative pairs. SimSiam [8] is a step in the direction of simpler models and a better understanding of the principles of Siamese networks. It prevents the degenerate problem by applying a stop-grad operation on one of the augmented views. Differently from BYOL [13] and other models for representation learning, SimSiam directly shares the weights between the branches elaborating the two augmentations. The SimSiam paper states that the method can be thought of as "SimCLR without negative pairs" and "SwAV without online clustering" [8].

The third category of non-collapsing function is the most related to the proposed work, which solves the degenerate problem by defining the loss function or the objective function that enforces the predicted class to be distributed more uniformly. The idea of Self-Classifier [1] is very similar to the one proposed in IIC [16] as their main point is to define a function that prioritizes solutions where the samples are spread among the different classes. IIC does that by defining an objective function that maximizes the mutual information between the predictions of two augmented views of the same sample. The loss function proposed by Self-Classifier instead is equivalent to the cross-entropy classification loss under a uniform label prior, which guarantees a non-degenerate, uniformly distributed optimal solution as explained in the original paper [1].

In Figure 1, we further illustrate the comparison in [8] with the models we described so far.

3.2 Loss function

The derivation, the mathematical properties, and the proofs behind the loss function implemented by Self-Classifier are clearly described in the original paper [1]. In short, the model adopts Bayes and total probability laws on each term of naive cross entropy and assumes a uniform prior for the class probability $p(y)$, providing a mathematical guarantee for the non-degenerate solution. The overall function can be represented as follow:

$$l(x_1, x_2) = \sum_{y \in [C]} \frac{p(x_2|y)}{\sum_{\tilde{y}} p(x_2|\tilde{y})} \log\left(\frac{N}{C} \frac{p(y|x_1)}{\sum_{\tilde{x}_1} p(y|\tilde{x}_1)}\right) \quad \mathcal{L} = \frac{1}{2}\Big(l(x_1, x_2) + l(x_2, x_1)\Big)$$

3.3 Implementation Details

Architecture Our architecture follows the same as described in the original paper [1], with the base architecture of Siamese network [3]. The model consists of a backbone, followed by a 2-layer MLP with batch normalization and LeakyReLU. A final layer of multiple classification heads is attached to the end. Differently from the original paper, we use Resnet-18 instead of Resnet-50 [15] to lower the computational cost and adapt to smaller datasets. We also scaled down the size of the final classification heads to be proportional to the number of classes in the datasets, keeping the ratio equal to the original paper's.

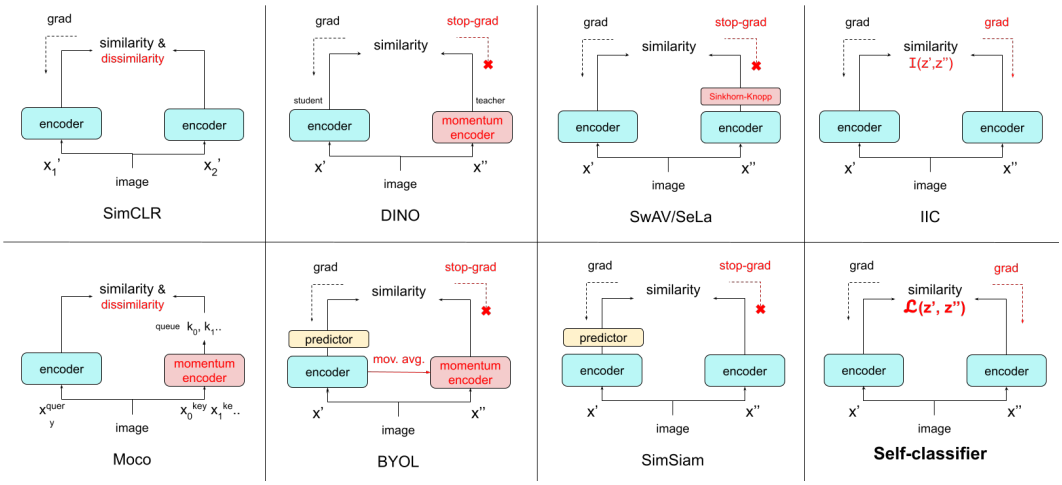


Figure 1: **Comparison of models on Siamese architectures.** The figure is inspired by the one in SimSiam [8]. We extend it with the main models described in this report. The difference between each model is highlighted in red and represents how the model solves the degenerate problem. The label "dissimilarity" indicates that the model employs negative pairs for training. The dash lines ending with an "X" indicate the presence of a stop gradient operation. The architecture for Self-Classifier is on the bottom right, which is similar to the one proposed in ICC [16] as they are both based on defining a non-degenerate function.

Image Augmentation We use the same augmentations presented in BYOL [13], which in turn took them from SimCLR [7][13]. These augmentations are almost standard in contrastive learning for visual representation and they are used by most of the models we see. The first augmentation consists in taking a random patch of the image and resizing it. In our implementation, the new dimensions depend on the dataset used. After this, we apply random horizontal flip, color distortion, Gaussian blur, and solarization [13]. The paper also lists two other augmentations: multi-crop from [5] and nearest neighbor from [12]. In our model, we skip the nearest neighbor augmentation for simplicity.

Optimization

Unsupervised training: We use a batch size of 512 with softmax temperatures of 0.1 row-wise (over all classes) and 0.05 column-wise (over all batches). We train the model with the SGD optimizer with a 0.9 momentum and 0.0001 weight decay. The learning rate starts at 0.01 and increased linearly to 0.4 within the first 10 epochs. After that, it decreases according to a cosine scheduler over the remaining 790 epochs, reaching a final value of 0.0004.

Linear training: We use a batch size of 512. We train the model with the AdamW optimizer with a momentum of 0.9 and no weight decay. The learning rate starts at 0.0004 and decreases toward zero following a cosine scheduler over the 100 epochs of training.

Note that we implemented LARS optimizer as the paper suggested, but it turned out to be worse than AdamW and SGD, which, according to [8], can be because that LARS is supposed to work better with large batch sizes, not applicable with our limited computation resources.

4 Data and benchmarks

We trained our model on common datasets and compared the obtained results with the standard unsupervised clustering benchmarks publicly available. These are: CIFAR-10 [18][2], CIFAR-20[17][3], STL-10 [9] [4] and MNIST [10][5].

[2]https://paperswithcode.com/sota/image-clustering-on-cifar-10
[3]https://paperswithcode.com/sota/unsupervised-image-classification-on-cifar-20
[4]https://paperswithcode.com/sota/image-clustering-on-stl-10
[5]https://paperswithcode.com/sota/image-clustering-on-mnist-full

5 Experiments and findings

5.1 Clustering results

We report the results based on standard clustering evaluation metrics illustrated in Table 1. The numbers are sub-optimal compared to the state-of-the-art methods in the unsupervised clustering benchmarks but achieve substantial improvement over the earlier methods. Although part of the reason for the unsatisfactory result may be that the hyperparameters are not tuned to the best, we also inferred that it could be because the method does not fit well in smaller datasets with low-resolution image inputs, i.e. the prior enforcement of the loss function requires abundant unseen data while none of the datasets we adopted can provide. The difference in the number of classes could also be a contributing factor. Further analysis on middle and large datasets needs to be run in order to explain the performance drop here.

Table 1: **Unsupervised image clustering using ResNet-18.** NMI: Normalized Mutual Information, AMI: Adjusted Normalized Mutual Information, ARI: Adjusted Rand-Index, ACC: Clustering accuracy. The performance falls behind SOTA models but is competitive compared to earlier models. The results are aligned with the difficulty of each benchmark, somewhat validating our implantation.

Dataset	NMI	AMI	ARI	ACC
MNIST[10]	83.9427	83.9140	76.8142	81.9000
CIFAR-10 [18]	50.9317	50.8450	41.5156	60.7300
CIFAR-20 [17]	30.9210	30.4974	17.8501	32.2400
STL-10 [9]	42.5199	42.3921	31.6741	51.8125

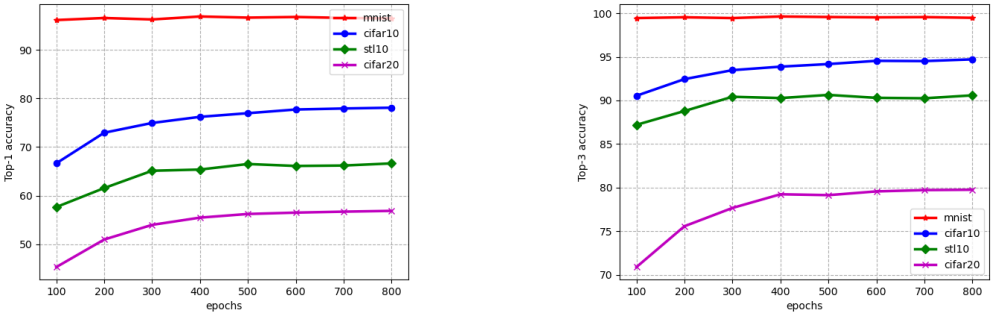
5.2 Linear probing

Following the linear classification protocol introduced in the original article, we froze the backbone parameters of the model after the unsupervised training and replaced the MLP and head layers with a single linear layer for a 100-epoch supervised training using standard cross-entropy loss. Figure 2 reports Top-1 and Top-3 accuracy for the classification performance. It is clearly shown in the figure that the quality of representations learned by the model improves with the number of epochs and the linear model converges to a satisfactory state at approximately epoch 500. The results of this experiment are compatible with those of unsupervised clustering, which validates the fact that the model simultaneously improves representation learning and clustering performance.

5.3 Qualitative results

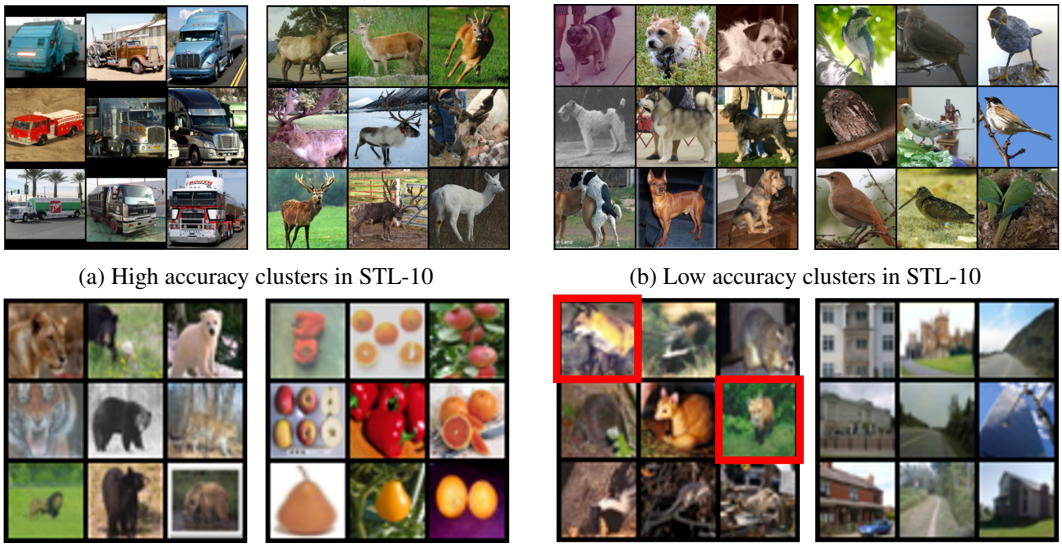
Cluster images visualization Following the paper, we report image samples of clusters with high and low accuracy from different datasets. As observed in Figure 3, the displayed images for each assigned cluster show strong consistency in the semantic meaning which validates the success of our implementation. While it is harder to identify miss-classified samples from STL-10 [9], due to its competitive clustering performance the representations in CIFAR-20 contain more errors. Specifically, Figure 3d shows images of fox and $squirrel$ clustered together, while they belong to separate classes, $medium-sized\ mammals$ and $small\ mammals$ respectively.

Additional visualization with t-SNE and predicted class number We also ran additional feature visualization over the model trained on STL-10 using t-SNE shown in Figure 4a. Interesting insights can be concluded from the figure including 1) There is a noticeable separation between natural classes ($cat, dog...$) displayed on the right side of the figure and artifact classes ($car, airplane$) on the left side. 2) Aside from inner-class consistency, the figure also demonstrates inter-class connection regarding their similarity $e.g.\ bird$ and $airplane$ class stay rather close at the bottom. Overall the result confirms how well the model learns to distinguish the different classes by extrapolating their main features. We would like to ensure the uniform prior distribution is well enforced through the model training, so we visualized the sample number over each predicted class inFigure 4b. While the numbers of each class are not exactly the same, the degenerate solution is clearly avoided with the proposed loss function.



(a) Top-1 Accuracy (b) Top-3 Accuracy

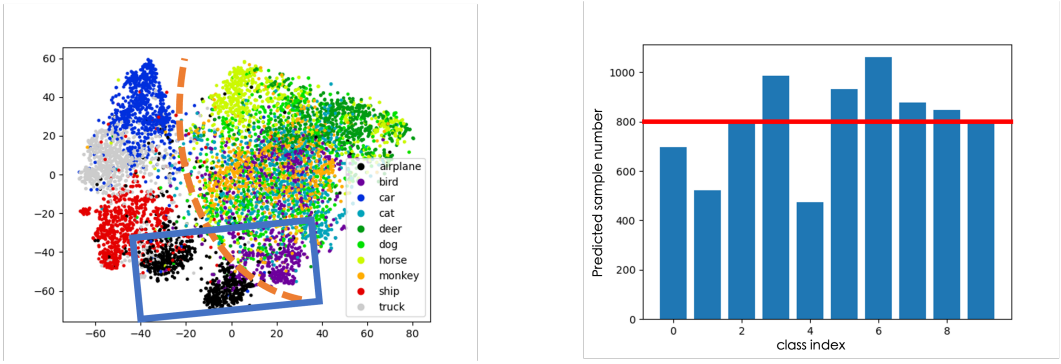
Figure 2: **Linear probing.** The x-axis indicates the pre-trained epochs of unsupervised learning before linear probing while y-axis represents the supervised performance.



(a) High accuracy clusters in STL-10 (b) Low accuracy clusters in STL-10

(c) High accuracy clusters in CIFAR-20 (d) Low accuracy clusters in CIFAR-20

Figure 3: **Image visualization for high and low accuracy classes in different datasets.**



(a) T-SNE feature visualization. (b) Predicted class number

Figure 4: **Additional visualization on dataset STL-10.** In subfigure (a), the orange dotted line separates human artifacts from natural animal features, and the blue square indicates the similarity between birds and airplanes; In subfigure (b), the red line is the actual class number (800 samples)

5.4 Ablation Study

We focus the ablation studies on the STL-10 [9] dataset. Table 2 shows comparisons between the base model and its variants, which are different in batch sizes, classification head sizes, temperature settings, or multi-crop augmentation usage. As in [4], we trained the model for 300 epochs for each configuration, as fewer iterations may not guarantee stable behavior. We can state that Self-Classifier is somewhat robust to different hyperparameter settings, with performance fluctuations under 5% for NMI and 8% for ACC. We assume ACC is more subject to oscillations because of the second step (linear sum assignment) adopted for unsupervised accuracy calculation. Interestingly, larger batches and more heads do not necessarily lead to better results. The numbers show a $1-4\%$ decrease in the accuracy when using the largest batch size compared to the smallest, which is quite counterintuitive. We assume that, while the loss function expects a larger batch size to enforce the uniform priors, a smaller batch size brings more randomness in the optimization resulting in a trade-off between the two above properties. Also, the performance drop brought by the additional multi-crop augmentation can be explained by the small resolution of images causing the local patches (even smaller) with limited knowledge to aid the entire training process.

Table 2: **Ablation study on unsupervised clustering metrics.** The base model has 512 batch size, $[10, 20, 40, 80]$ classification head size, and (0.10, 0.05) temperature setting.

Model Variants		NMI	AMI	ARI	ACC
Batch Size	128	39.0715	38.9363	27.6416	48.8500
	256	41.0984	40.9675	27.3900	46.4750
	1024	39.6265	39.4921	26.2468	45.2375
Head Size	$1 * 10$	39.8843	39.7506	27.4368	48.7750
	$5 * 10$	38.2264	38.0888	26.1061	47.3250
	$10 * 10$	38.8565	38.7202	25.2499	44.5375
	$15 * 10$	39.9663	39.8326	25.9415	44.0750
Temperature (τ_{row}, τ_{col})	$(0.07, 0.03)$	40.8495	40.7174	28.4016	48.4375
	$(0.07, 0.05)$	**42.5199**	**42.3921**	**31.6741**	**51.8125**
	$(0.10, 0.03)$	40.6805	40.5485	27.7525	45.7875
Base with Multi-Crop		40.5333	40.4014	25.5025	45.0875
Base		40.3517	40.2191	29.2059	50.3375

6 Challenges

The main challenge that occurred in the implementation is to check for the specific values of some of the hyperparameters in the official repository (e.g. optimizer and augmentation specifics like kernel size for the Gaussian blur) Even with reference to the official code, we did not implement the nearest neighbor augmentation mentioned in the paper for its significant increase in computational cost, and the improvement is rather not obvious. Also, we spent a lot of time tuning the hyperparameters like which optimizer to choose, the batch size, learning rate, etc. with some of the results included in the ablation study. Also, we noticed an interesting thing that sometimes a minor modification in learning rate (0.04 to 0.05 for example) could cause a significant increase or drop in performance.

7 Conclusion

In this work, we successfully re-implement Self-Classifier [1] and explained in detail the peculiarities it has compared to other models designed to solve the degenerate solution problem and why it is revolutionary. We then demonstrate its capabilities to achieve comparable results on multiple smaller datasets and how it simultaneously solves the feature representation and clustering tasks through both quantitative experiments and extensive visualizations, which suggest that the model may be more suitable in larger datasets. Overall, most of the obtained results are in alignment with the paper suggested, but we do fall behind the intended SOTA performance. If our work is correct then future work could be done to improve the model's performance on smaller datasets and try to reduce its reliance on a known class number, for that can sometimes be hard to obtain.

8 Self Assessment

Overall, we did comprehensive and concrete experiments over our implemented model, with some evaluation metrics going beyond the original paper's. We have put a lot of effort into report writing, clearly addressed interesting findings during the implementation as well as summarized the previous literature with detail and our own opinions. We believe this to be in line with the assessment criteria for the grade "very good" or "excellent".

References

[1] Elad Amrani, Leonid Karlinsky, and Alex Bronstein. Self-supervised classification network.

[2] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning.

[3] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pages 737–744. Morgan Kaufmann Publishers Inc.

[4] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features.

[5] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments.

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers.

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

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

[9] Adam Coates, A. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning.

[10] Li Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. 29(6):141–142.

[11] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks.

[12] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations.

[13] 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.

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

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

[16] Xu Ji, João F. Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation.

[17] A. Krizhevsky. Learning multiple layers of features from tiny images.

[18] A. Krizhevsky. Learning multiple layers of features from tiny images, technical report.

[19] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. SCAN: Learning to classify images without labels.

[20] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination.

[21] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis.

[22] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters.