

Application of a Self-supervised Learning Technique for Monitoring Industrial Spaces

V. Magalhães[1(✉)], M. Fernanda P. Costa[2], M. J. Oliveira Ferreira[1], T. Pinto[1], and V. Figueiredo[1]

[1] Neadvance - Machine Vision S.A., 4705-002 Sequeira, Braga, Portugal
{vmagalhaes,mferreira,tpinto,vfigueiredo}@neadvance.com
[2] Centre of Mathematics, University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
mfc@math.uminho.pt
http://www.neadvance.com/, http://www.cmat.uminho.pt/

Abstract. Supervised learning has reached a bottleneck as they require expensive and time-consuming annotations. In addition, in some problems, such as in industrial spaces, it is not always possible to acquire a large number of images. Self-supervised learning helps these issues by extracting information from the data itself, without requiring labels and has achieved good performance, closing the gap between supervised and self-supervised learning. This work presents the application of a self-supervised learning method - SwAV, that classifies anomalies in an industrial space, evaluates its performance and compares the results to the supervised paradigm.

Keywords: Computer Vision · Deep Learning · Self-Supervised Learning · SwAV · Industrial Spaces

1 Introduction

The rise of algorithms capable of detecting and recognizing objects in uncontrolled environments can bring significant benefits to logistics in industrial environments. These technologies provide a more efficient and intelligent management of factory space by automating surveillance and monitoring activities that are mostly performed manually or with CCTV systems that monitor critical areas. These systems are connected to monitors at a control post, where operators manually perceive the monitored areas and detect abnormal situations.

This work was funded by Project "SMARTICS - Platform for Smart Monitoring of Open Industrial Spaces using Computer Vision and Artificial Intelligence", Project n.º LISBOA-01-0247-FEDER-072553, financed by the European Regional Development Fund (ERDF), through the COMPETE 2020 - Competitiveness and Internationalization Operational Program (POCI) and PORTUGAL 2020.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14104, pp. 407–420, 2023.
https://doi.org/10.1007/978-3-031-37105-9_27

The use of intelligent logistics solutions reduces the need for human operators to manually monitor specific spaces, reducing the risk of human error and increasing the efficiency of the monitoring process. These algorithms can also process a larger amount of video images and cover a larger area than operators, allowing for more effective monitoring. They increase the efficiency of production by reducing operational and maintenance costs and decreasing waiting times for production and machine downtime.

Deep Learning has brought significant development in automated computer vision systems such as object detection [1], image classification [2], and image segmentation [3], which is useful for monitoring these industrial spaces. However, the success of these systems require a large amount of labeled data, which can be difficult to acquire in industrial settings. Therefore, researchers are working on systems that can adapt to changing conditions without extensive supervision, such as self-supervised learning [4,5,8]. Self-supervised learning methods construct feature representations without manual annotations using pretext tasks, which allows models to solve these tasks by extracting useful information that can later improve downstream tasks. Further self-supervised learning methods use contrastive learning to push positive instances closer together, and negative ones further apart, in the embedding space [9,12]. These methods have shown promising results in closing the performance gap with supervised learning, and some researchers believe that the next AI revolution will be in self-supervised learning rather than supervised learning [12].

In this work, we present the application of a self-supervised learning method - SwAV, for detecting anomalies in two different scenarios in an industrial space. One of the scenarios is to classify whether a shelf is empty or not and the other is to classify if a set of boxes is correctly positioned in a tilted conveyor belt.

The present paper is structured as follows. Section 2 provides some concepts of Self-Supervised Learning. Section 3 presents and explains the SwAV method. Section 4 presents the datasets used to evaluate the performance of the SwAV method. Section 5 presents the experiments made with each dataset and the results obtained. The paper ends with discussion and future work.

2 Self-supervised Learning

Deep Learning has achieved great success in the last decade, especially with supervised learning in computer vision tasks such as image classification [2,13], semantic segmentation [3,14,15], and object detection [1,16,17].

Supervised Learning methods heavily rely on having a large amount of labeled training data, and the quality of deep neural networks is directly impacted by the number of annotated images. However, creating annotated datasets is expensive and time-consuming, and even though there is a lot of data available, labeling it is not always feasible. While there are some datasets, like *Imagenet*, with over a million images [10], it is not always possible to gather labeled data on such a large scale in real-world scenarios. Furthermore, Supervised Learning can sometimes produce errors due to poor generalization and spurious correlations [8,12]. To

address these issues, researchers are exploring alternative methods that can make use of the vast amounts of available unlabeled data without requiring extensive labeling.

Self-Supervised Learning (SSL) is gaining attention as a promising alternative to Supervised Learning because it has shown remarkable performance in learning representations without requiring the annotation of large datasets [8,9]. In SSL, unlabelled data is provided to the model, which is then automatically labeled during the training phase by identifying and utilizing the connections between different input features. This compels the network to acquire semantic representations of the data, which can be applied as pre-trained models to downstream tasks to enhance their performance and prevent over-fitting [4,6,7,12].

The cognitive motivation behind SSL is the way infants learn, largely through observation. Within few months of birth, infants have meaningful expectations about the world around them, concepts such as gravity and object permanence. Infants' environment becomes a source of supervision that helps them develop a general understanding of how things work. SSL is an attempt to apply this concept to machines, where the data itself contains inherent features that provide supervision for training the model, rather than labels that tell the network what is right and what is not [12]. Some authors believe that SSL is a promising way to build background knowledge and approximate a kind of common sense in AI systems. Also, when humans do new tasks, information from life experience and prior knowledge is used. SSL mimics this when it transfers the learned features to other tasks [25,26].

During the initial phase of SSL, pretext tasks were introduced to train models to automatically generate labels based on data characteristics, which were used to learn representations. The model acquires features while solving the pretext task, and these features are subsequently transferred to other tasks by transfer learning. Examples of pretext tasks are coloring a grayscale image [27–30], predicting the degree of rotation of an image [31], filling in a missing part of an image [32,33], predicting the relative position of an image [34], solving a jigsaw puzzle [35–37], and more.

Recently, Contrastive Learning has emerged as a highly effective technique that bridges the gap between SSL and supervised learning [38]. It is a discriminative model that uses positive and negative pairs to learn representations by distinguishing between different views of the same image [6]. Essentially, it tries to bring similar views of the same instance closer and different instances farther apart in the representation space [5,9,11,12]. DeepCluster [18], ClusterFit [19], SimSiam [20], PIRL [21], MoCo [22], SimCLR [23], and SwAV [24] are examples of contrastive learning methods.

SwAV is the model used in this work and has achieved state-of-the-art performance on image classification. It is a unique method for learning visual representations by comparing different views of the same image and grouping similar images together. This allows the model to learn a more robust and generalized representation of the visual world. This capability allows companies to use their existing data to train models for a variety of tasks, such as quality control and

anomaly detection, without the need for expensive and time-consuming manual labeling. This method is described in more detail in the following section.

3 Swapping Assignments Between Multiple Views (SwAV)

Caron. M., et al. [24] purposed an online clustering-based self-supervised method for learning feature representations named **SwAV**.

The authors purposed a *multi-crop* augmentation strategy that generates multiple views of the same image instead of just one pair without quadratically increasing memory and computational requirements. They use two standard-resolution crops and take V additional low-resolution crops that cover only small portions of the image. Using low resolution images provides only a small increase in computational cost and the model becomes scale invariant. This strategy showed an improvement in performance not only for SwAV but also for other contrastive learning methods [24]. Next, random horizontal flips, color distortions, and Gaussian blur are applied to each resulting crop.

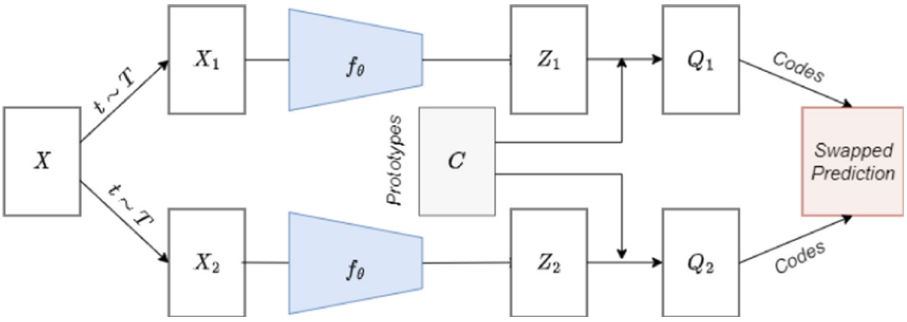


Fig. 1. SwAV Architecture

To generate augmented views of an image x, a random transformation t is selected from a set of image transformations T, and applied to create views x_1 and x_2. For simplicity, only 2 augmented views are listed, but there can be many more. The augmented views are then passed through a convolutional neural network (CNN) f_θ, such as ResNet50, which outputs feature vectors z_1 and z_2 after passing through a projection head (Fig. 1).

The feature vectors z_1 and z_2 are mapped to a set of K trainable prototype vectors $c_1, c_2, ..., c_K$, forming the columns of a matrix C. This mapping is represented by $Q = [q_1, \ldots, q_B]$ as shown in Fig. 2 (left). The matrix Q is optimized to maximize the similarity between the feature vectors and the prototypes, i.e.,

$$max_Q = Tr(Q^T C^T Z) + \epsilon H(Q), \text{ where } H(Q) = -\sum_{ij} Q_{ij} log Q_{ij} \qquad (1)$$

is the entropy function and ϵ is a parameter that controls the smoothness of the mapping. In practice, the value of ϵ is kept low to prevent a trivial solution, where all samples are mapped to a single representation and are uniformly assigned to all prototypes. To ensure an equal distribution of prototypes, the matrix Q is constrained such that each prototype is selected the same number of times. Once the optimal solution Q^* is found, it takes the form of a normalized exponential matrix:

$$Q^* = Diag(u)exp\left(\frac{C^T Z}{\epsilon}\right) Diag(v) \tag{2}$$

where $u \in R^K$ and $v \in R^B$ are renormalization vectors. These vectors are computed using the iterative *Sinkhorn-Knopp* algorithm [40].

A "swapped" prediction problem is formulated, consisting of predicting the code q_1 from the feature z_2, and the code q_2 from the feature z_1. The loss function used for this task is as follows:

$$L(z_1, z_2) = \ell(z_1, q_2) + \ell(z_2, q_1) \tag{3}$$

where $\ell(z_1, q_2)$ is the cross-entropy loss between the code and the probability obtained by taking a *softmax* of the dot products of z_i and all prototypes in C:

$$\ell(z_t, q_s) = -\sum_k q_s^{(k)} log p_t^{(k)}, \text{ where } p_t^{(k)} = \frac{exp((z_t^T c_k)/\tau)}{\sum_{k'} exp((z_t^T c_{k'})/\tau)} \tag{4}$$

where τ denotes a temperature parameter.

To learn the features, the loss function is minimized with respect to the trainable prototypes C and the parameters θ of the encoder f_θ. The SwAV method compares the features z_1 and z_2 using intermediate codes q_1 and q_2. If these features capture similar information, it should be possible to predict the code from the other feature. SwAV achieves this by swapping assignments between multiple views of the same image (Fig. 2 (right)). This allows the model to compare and learn from different views of the same image, and improve the quality of the learned representations.

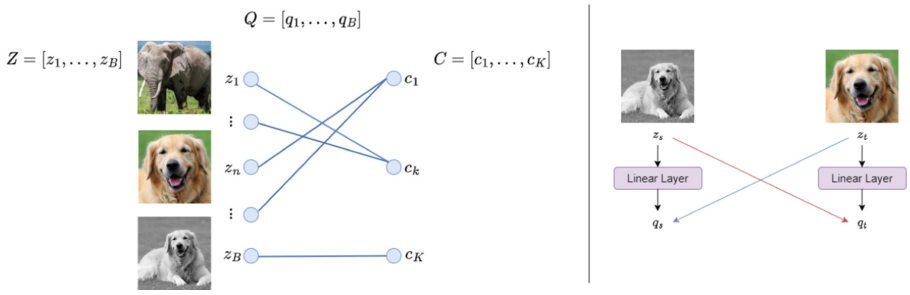


Fig. 2. Left: Assigning B samples to K trainable prototype vectors, **Right**: Swapped prediction problem between two views of the same image

4 Datasets

This work used three datasets for training and evaluating the model, two of which are laboratory scenarios that simulate real anomalies in an industrial setting.

4.1 Dataset 1: Flowers Dataset

One of the datasets used was *Tensorflow's flowers dataset* [39], which comprises of 3670 images of flowers and their respective labels for classification. The dataset consists of five distinct types of flowers, namely Dandelion (represented by 0), Daisy (represented by 1), Tulip (represented by 2), Sunflower (represented by 3), and Rose (represented by 4) (Fig. 3).



Fig. 3. Examples of images from Tensorflow's flowers dataset with respective labels

4.2 Dataset 2: BookCase Dataset

The *bookcase dataset* created was designed to resemble a real problem in an industrial setting, where the goal is to identify if a shelf is empty or not, so that an alert can be triggered if it is empty, to be consequently refilled by an automated guided vehicle. This is done in real-time where the SwAV algorithm is applied to classify the shelf. In real industrial settings the ranging background, the lighting conditions, obstructions, the different shapes, colors and sizes of the objects all contribute to make the task of teaching the model what to expect difficult.

To prepare the dataset, photographs of a bookcase were taken with different proximity, and the images were cropped and sliced to create individual images of each shelf. The resulting dataset contains 4597 images, out of which 1094 were empty shelves (represented as 0) and 3505 were not empty shelves (represented as 1). Some examples of images from the bookcase dataset are shown in Fig. 4.

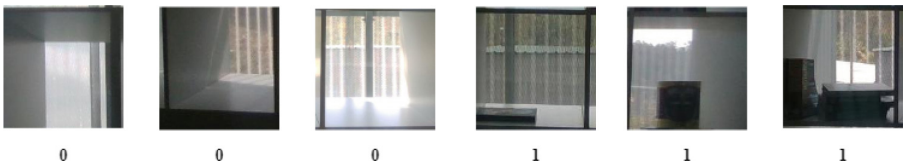


Fig. 4. Examples of images from the created bookcase dataset with respective labels

4.3 Dataset 2: Boxes Dataset

Another industrial problem presented was a tilted conveyor belt where boxes slide. Sometimes the boxes get stuck or mispositioned, ruining the automated process at hand. Again, in real-time SwAV is used to classify this anomaly, so that an alert can be sent and the situation corrected by an operator. The boxes dataset was created to simulate this scenario. Several images were taken with boxes in different positions or even with new, different, and incorrectly present boxes. The goal is to classify if the correct boxes are positioned correctly or if there is an anomaly in the positioning or the presence of other types of boxes.

This dataset contains a total of 116 images, 37 correct images (labelled 1) and 79 incorrect images (labelled 0). Figure 5 shows some image examples from the dataset.



Fig. 5. Examples of images from the created boxes dataset with respective labels

5 Experiments and Results

In this section, we present the experiments made with each dataset and analyze their performance.

5.1 Training

In all the experiments *multi-crop* is applied to every batch of images fed to the network. From each image, two 224×224 high-resolution views and three 96×96 low-resolution views are generated. Random horizontal flips, color distortions, and Gaussian blur are applied to each resulting crop. The images are then fed to the backbone ResNet50 model and through 2 Dense layers with ReLU (projection head). SwAV is trained with mini-batch SGD using batches of 32 different instances. Using the original SwAV article [24] as a reference the temperature parameter τ was set to 0.1 and the Sinkhorn regularization parameter ϵ was set

to 0.05 for all runs, as was the number of prototypes that was set to 15. Early stopping was defined to stop training if the loss did not improve in 15 epochs.

We evaluate the features of the ResNet-50 trained with SwAV on these datasets by training a linear classifier on top of the frozen representations. This linear layer is trained during 100 epochs, with categorical cross-entropy loss, Adam optimizer [41], and a learning rate of 0.001. Early stopping was defined to restore the best weights and stop training if the validation loss did not improve in 5 epochs.

5.2 Results Obtained with Dataset 1: Flowers Dataset

SwAV was initially trained on the flowers dataset to see its performance, to then train it on the custom dataset.

For the SwAV training 5 experiments were made. In all these experiments SwAV was trained with 85% of the data, corresponding to 3120 images, and the evaluation of the model was with the remaining 15%, corresponding to 550 images. To evaluate the feature representations a linear classifier is trained on the frozen features with a single dense layer with 5 neurons, with softmax activation function and a l^2 regularizer. This linear layer is trained with the same 85% of data that SwAV is trained with and then evaluated on the 550 unseen images.

The results obtained from all experiments are displayed in Table 1.

Table 1. Results of a linear classifier on SwAVs frozen features with flowers dataset

Experiment	Loss	Accuracy	Precision	Recall	F1-Score
Flowers A	1.298	0.447	0.48	0.46	0.44
Flowers B	0.412	0.856	0.86	0.86	0.86
Flowers C	**0.327**	**0.882**	**0.88**	**0.88**	**0.88**
Flowers D	0.397	0.860	0.86	0.86	0.86
Flowers E	0.348	0.871	0.87	0.87	0.87

Given the experiments done, experiment **Flowers C** was the one that achieved better performance. In more detail, the ResNet50 was initialized with *imagenet* weights. The mini-batch SGD decayed the learning rate to 0.0001, and the number of epochs was set to 300 but stopped at epoch 225 due to early stopping.

The linear classifier achieved 88,2% of accuracy. Table 2 shows the confusion matrix obtained of the best model. The model predicted incorrectly 65 images out of a total of 550 images. Dandelion is the class with the highest recall as it is the class that fewer images were misclassified, only 8.9%. Rose has the highest percentage of misclassified images - 15.9%. Daisy is the class with fewer false positives, having the highest precision.

These experiments could have been optimized to achieve better results but because this dataset was not the focus of the work no more experiments were made.

Table 2. Confusion matrix from experiment Flowers C

True Labels	**Dandelion**	102	3	2	5	0
	Daisy	8	83	1	2	1
	Tulip	0	3	119	3	9
	Sunflower	4	1	5	91	1
	Rose	4	1	10	2	90
		Dandelion	**Daisy**	**Tulip**	**Sunflower**	**Rose**
	Predicted Labels					

5.3 Results Obtained with Dataset 2: BookCase Dataset

As described earlier, the bookcase dataset was created to address a real-world problem in an industrial space context. Since the Flowers C experiment performed best on the flower dataset, SwAV training and the linear classifier were performed on the bookcase dataset with exactly the same settings. The only difference was that the early stopping in the SwAV training was set to 30. The dataset was also split into 85% training and 15% for evaluation, corresponding to 3907 and 690 images, respectively. SwAV was trained during 308 epochs. This experiment was defined as **BookCase F** and the results of the linear classifier on top of the frozen features are shown in Table 3.

Table 3. Results of a linear classifier on SwAVs frozen features with bookcase dataset

Experiment	Loss	Accuracy	Precision	Recall	F1-Score	MCC
BookCase F	0.032	0.988	0.99	0.99	0.99	0.98

The linear classifier achieved 98.8% accuracy, 98% MCC, and 99% on the remaining metrics. The score obtained shows that the binary classifier was able to predict the majority of positive and negative instances.

Table 4 displays the confusion matrix from this experiment. The model only misclassified 6 images from a total of 690: 4 images were misclassified as not empty and 2 as empty.

The bookcase dataset was also trained in a fully supervised manner with hyperparameter optimization to determine the best possible model. The model with the best results achieved 99.6% accuracy. The model was a DenseNet121 with max pooling, 4 dense layers, Adam optimizer, and a learning rate of 0.0001. Table 5 displays the results of a comparison between the self-supervised learning method SwAV and an optimized supervised learning setting. SwAV achieves only –0.8% compared to the supervised learning approach.

Table 4. Confusion matrix from experiment BookCase F

True Lables	**Empty (0)**	152	4
	Not Empty (1)	2	532
		Empty (0)	**Not Empty (1)**
		Predicted Labels	

Table 5. Results comparing SwAV and supervised learning with the bookcase dataset

Method	Accuracy
Supervised	0.996
SwAV	0.988

5.4 Results Obtained with Transfer Learning

The other experiment made was with transfer learning. The learned representations from SwAV training on the bookcase dataset were used to solve the classification problem with the boxes dataset. A linear classifier was trained during 500 epochs on the frozen features (learned with the bookcase dataset) with a single dense layer of 2 neurons, with softmax function and a l^2 regularizer. The dataset was split into 70% for training and 30% for evaluation. The dataset was split this way due to the small amount of data. Dividing it as previously 85–15 would only leave 17 images for evaluation, which would not be enough to evaluate the model. Table 6 shows the results achieved. The model correctly predicted all of the 35 images, achieving 100% in all metrics.

Table 6. Confusion matrix of the linear classifier on the boxes dataset with frozen features from the bookcase dataset

True Lables	**Incorrect (0)**	27	0
	Correct (1)	0	8
		Incorrect (0)	**Correct (1)**
		Predicted Labels	

This dataset was also trained in a fully supervised manner, with 10 models trained with different hyperparameters to obtain the best possible model. To allow a fair comparison, the dataset was also split into 70–30%. Figure 7 shows the results obtained. The model achieved an accuracy of 81.2% and an MCC of 56.6%, with 8 images being misclassified. These results are due to the dataset containing very few images that the model can learn from.

In this particular case and with these settings, the information from the bookcase dataset was useful to improve the performance compared to the supervised model, outperforming it by 18.8%. Another advantage is reduced training time.

As a result of using information already trained with the bookcase dataset, it is only necessary to train the linear classifier, which is much quicker. If appropriate, the features trained with SwAV can be transferred to various other problems.

Table 7. Confusion matrix of the boxes dataset in a supervised setting

True Lables	**Incorrect (0)**	17	6
	Correct (1)	2	10
		Incorrect (0)	**Correct (1)**
		Predicted Labels	

However, for new images, the model did not classify all of them correctly. Figure 6 was one of these images that the model incorrectly classified as incorrect. As shown the background in these images was quite different from that of the original dataset and contributed to the incorrect classification. This shows that the model did not generalize well. This is also because the dataset is small and very specific.



Fig. 6. New image with different angle and background noise. The model predicted it incorrectly as 0

During the SwAV training on the bookcase dataset, certain features were learned. The linear classifier picked up these features and tried to fit them into the two classes of the boxes dataset. However, the meaningful features of the bookcase problem may cause the model to assign the new images to the wrong classes when looking at the new problem. This happens because the learned features are not appropriate for the new problem. One way to achieve better results would be to acquire more images with different noise and train SwAV with this dataset as well. SwAV would extract specific features from this dataset which would improve performance.

6 Discussion and Future Work

Self-Supervised Learning is progressing and achieving results close to supervised learning. In this work, SwAV was trained with Tensorflow's flowers dataset

using different parameters. To evaluate the method a linear classifier, with the frozen representations of the ResNet-50 trained with SwAV, was trained. The SSL method achieved a 88.2% accuracy. SwAV was then trained with the same settings on a custom bookcase dataset that simulated a real-world industrial scenario. It achieved an accuracy of 98.8% and an MCC of 98%, approaching the 99,6% accuracy accomplished by a supervised learning setting.

Regarding transfer learning to downstream tasks, the self-supervised method improved performance compared to the supervised model. However, the model did not generalize well as when given new images the model failed to predict them correctly. The features learned from the bookcase model were not adequate to solve the new problem with the boxes dataset. This is not surprising because SwAV was trained on a very specific dataset with only two classes. It does not extract the same features as if it was trained on, for example, *ImageNet* that has over a million images and 1000 classes.

In general, SwAV extracts useful features that can add information to solve other tasks. However, this depends on the datasets, the task, and the features learned during SwAV training. Also, SSL methods require a lot of computational power which can be a limitation. Depending on the problem and resources, SSL can be a good option.

In terms of future work, the aim is to train SwAV on a big industrial dataset with various images from different projects. There will be a variety of classes and the model will learn better features as it separates the classes in the embedding space and consequently generalize better. We intend to then apply transfer learning of the learned features to other problems and analyze if and how much improvement is made.

References

1. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
3. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille. A.L.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 834–848 (2018). https://doi.org/10.1109/tpami.2017.269918
4. Kolesnikov, A., Zhai, X., Beyer, L.: Revisiting self-supervised visual representation learning. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1920–1929 (2019). https://doi.org/10.1109/CVPR.2019.00202
5. Goyal, P., Mahajan, D., Gupta. A., Misra, I.: Scaling and benchmarking self-supervised visual representation learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6391–6400 (2019)
6. Albelwi, S.: Survey on self-supervised learning: auxiliary pretext tasks and contrastive learning methods in imaging. Entropy **24**(4), 551. MDPI (2022)

7. Longlong, J., Yingli, T.: Self-supervised visual feature learning with deep neural networks: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **43**(11), 4037–4058 (2020)
8. Xiao, L., et al.: Self-supervised learning: generative or contrastive. IEEE Trans. Knowl. Data Eng. **35**(1), 857–876 (2021)
9. Jaiswal, A., Babu, A.R., Zadeh, M.A., Banerjee, D., Makedon, F.: A survey on contrastive self-supervised learning. Technologies **9**(1), 2. MDPI (2020)
10. Schmarje, L., Santarossa. M., Schröder, S.-M., Koch, R.: A survey on semi-, self- and unsupervised learning for image classification. IEEE Access **9**, 82146–82168. IEEE (2021)
11. Yeh, C.-H., Hong, C.-Y., Hsu, Y.-C., Liy, T.-L., Chen, Y., LeCun, Y.: Decoupled contrastive learning. In: Avidan, S., Brostow, G., Cisse, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision – ECCV 2022. ECCV 2022. LNCS, vol. 13686, pp. 668–684. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19809-0_38
12. Ohri, K., Kumar, M.: Review on self-supervised image recognition using deep neural networks. Knowl.-Based Syst. **224**, 107090. Elsevier (2021)
13. Simonyan, K., Zisserman, A: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations (ICLR), pp. 1–14 (2014)
14. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
15. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969 (2017)
16. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)
17. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, vol. 28 (2015)
18. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 132–149 (2018)
19. Yan, X., Misra, I., Gupta, A., Ghadiyaram, D., Mahajan, D.: ClusterFit: improving generalization of visual representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6509–6518 (2020)
20. Chen, X., He, K.: Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15750–15758 (2021)
21. Misra, I., Maaten, L.V.D.: Self-supervised learning of pretext-invariant representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6707–6717 (2020)
22. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9729–9738 (2020)
23. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International Conference on Machine Learning, pp. 1597–1607. PMLR (2020)
24. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. Adv. Neural Inf. Process. Syst. **33**, 9912–9924 (2020)

25. Dickson, B.: Meta's Yann LeCun is betting on self-supervised learning to unlock human-compatible AI, 2020. https://thenextweb.com/news/metas-yann-lecun-is-betting-on-self-supervised-learning-to-unlock-human-compatible-ai. Accessed 16 Jan 2023
26. LeCun, Y., Misra, I.: Self-supervised learning: the dark matter of intelligence (2021). https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/. Accessed 16 Jan 2023
27. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 649–666. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_40
28. Zhang, R., Isola, P., Efros, A.A.: Split-brain autoencoders: unsupervised learning by cross-channel prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1058–1067 (2017)
29. Larsson, G., Maire, M., Shakhnarovich, G.: Learning representations for automatic colorization. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 577–593. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_35
30. Larsson, G., Maire, M., Shakhnarovich, G.: Colorization as a proxy task for visual understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6874–6883 (2017)
31. Komodakis, N., Gidaris, S.: Unsupervised representation learning by predicting image rotations. In: International Conference on Learning Representations (ICLR) (2018)
32. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: feature learning by inpainting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2536–2544 (2016)
33. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM Trans. Graph. (ToG) **36**(4), 1–14 (2017)
34. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1422–1430 (2015)
35. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9910, pp. 69–84. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46466-4_5
36. Kim, D., Cho, D., Yoo, D., Kweon, I.S.: Learning image representations by completing damaged jigsaw puzzles. In: IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 793–802. IEEE (2018)
37. Wei, C., et al.: Iterative reorganization with weak spatial constraints: solving arbitrary jigsaw puzzles for unsupervised representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1910–1919 (2019)
38. Ericsson, L., Gouk, H., Hospedales, T.M.: How well do self-supervised models transfer? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5414–5423 (2021)
39. The TensorFlow Team: Flowers (2019). http://download.tensorflow.org/example_images/flower_photos.tgz
40. Cuturi, M.: Sinkhorn distances: lightspeed computation of optimal transport. In: Advances in Neural Information Processing Systems, vol. 26, pp. 2292–2300 (2013)
41. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)