Project of Self-Supervised Learning for ELEC 825

Xiaodong Wu*, Anusha Katha†, Rain Wei‡
*20382304, †20359557, ‡20161993
Department of Electrical and Computer Engineering
Date of Submission: 12 December, 2023

Abstract

This paper explores various methodologies—Momentum Contrast (MoCo), Bootstrap Your Own Latent (BYOL), and SimCLR—aimed at crafting unsupervised and effective visual representations without human supervision. The investigation focuses on training and validating these methodologies across small and medium-sized datasets, subsequently analyzing and comparing their performance. The paper begins with a summary of related work and their contributions to the development of the studied methodologies, followed by an in-depth examination of each model's implementation details and an overview of the experimental environment's setup. The experimental results are analyzed to ensure that overfitting and underfitting problems do not exist in the model.

I. Introduction

Computer vision has long grappled with the challenge of deriving effective visual representations without direct human supervision. Historically dominated by generative models stressing pixel-level generation, recent developments have seen a boom in discriminative techniques based on unlabeled datasets. These tasks have evolved from heuristic-based designs to complex methods that use contrastive learning in the latent space, setting new standards in the process.

Momentum Contrast (MoCo) is a significant advancement in this arena. MoCo creates dynamic dictionaries for contrastive learning utilizing a queue-based technique, allowing it to accommodate enormous sets of negative pairs without being constrained by mini-batch sizes. This method, supported by a slow-moving average key encoder, guarantees that these dictionaries remain consistent throughout training. MoCo outperforms its supervised counterparts in specific cases when tested across a variety of downstream tasks [1][2][3]. Similarly, Bootstrap Your Own Latent (BYOL) is a paradigm shift. Unlike classic contrastive approaches, which rely on negative pairs[4] BYOL does not. It works on a novel principle in which two networks, termed 'online' and 'target,' learn from each other in a bootstrapping manner. Intriguingly, this design decision results in robust representations that are less vulnerable to image augmentation selection, defying previously accepted assumptions about the importance of negative pairs in contrastive learning [5][6][7].

SimCLR considerably simplifies the unsupervised learning landscape. It eliminates the requirement for the specialized architectures[8][9] and memory banks[1][10][11] that were common in previous models. SimCLR highlights the relevance of data augmentation compositions at its core and includes a learnable nonlinear transformation into the representation-learning process. Diverging from typical supervised learning paradigms, the framework benefits greatly from bigger batch sizes and longer training durations. This method has broken records in self-supervised and semi-supervised learning tasks, especially on the ImageNet benchmark [12][13][14].

Together, MoCo, BYOL, and SimCLR represent a seismic shift in unsupervised visual representation learning. Each, with its distinct methodology, has contributed to the broader understanding and capabilities in the field. They collectively challenge the existing norms of supervised learning, paving the way for more efficient, scalable, and adaptable methods in extracting and learning visual features from unlabelled data[1][11][12][15][16]. These developments also highlight the changing nature of unsupervised learning, from one that relied primarily on heuristics and limited pretext tasks to one that welcomes complexity and depth in learning paradigms. As these methods are polished and expanded, they have the potential to unleash new potentials in computer vision, spurring innovation and applications across a wide range of areas [17][18].

II. Related Work

Starting from BERT [16], self-supervised learning was proposed to learn from massive data without any annotation, which makes cheap but efficient training possible. In BERT, they propose to mask part of the sentence to make model predict the masked words so that the relevance among words in a sentence can be learned. The big success in the text domain make the self-supervised learning be quickly applied to the image domain. Apart from the BYOL [19], MOCO [11], and SimCLR [20] which apply contrastive learning as introduced in this report, some other methods are transfered from the language models to solve this problem. For example, in Bao et al. [21], they propose a new model called BEIT, developed from BERT to deal with image data. Specifically, BEIT first split the images into separate patches which preserve raw pixels. Then, a discrete variational autoencoder (dVAE) is trained to tokenizer these image patches into image tokens and a followed decoder to use to re-construct the original image from these tokens. The training process is operated by masking some of the image patches and learning to predict them. Based on this work, IBOT was proposed in [22] by Kaiming He, the author of MoCo. It is highlighted in this work that unlike text domain, where the tokenization can transfer the raw text to tokens with rich semantics, in image

domain, the previous patching based methods have difficulties in extract semantic of images because the image patches are continuous distributed with considerable abundant information. Based on this, they propose to use Self-distillation to remove unnecessary abundant information to extract significant semantics. Specifically, they utilize an online tokenizer as the teacher model to distill the masked tokens that the student model, i.e., the vision models to be learned, need to re-construct. With this design, IBOT can achieve 83.8% accuracy, which makes it the state of the art method in self-supervised learning at that time. Besides, there are some works designing CNN-based models to do learning without labeled data. For instance, Gao et al. [23] designed a ConvMAE structure, combining the advantages from multi-scale hierarchical representation and local inductive bias. In detail, ConvMAE also has a encoder-decoder structure. In encoder, there are two masked convolution blocks followed by a transformer block which embedding the images from multi-scale. Then, these embeddings are entered into a transformer block based decoder, providing rich sensor fields from the original image for the re-construction.

III. Method

A. MoCo

This algorithm was proposed by Kaimining et al. [11]. The primary goal of it is to learn a visual representation from unlabeled images by contrasting positive pairs and negative pairs in a momentum space. Specifically, MoCo is built on the principles of contrastive learning. The model is trained to maximize the similarity between positive pairs (augmented views of the same image) and minimize the similarity between negative pairs (augmented views of different images). To achieve this, similar to other representation learning methods, MoCo also maintain a dynamic dictionary and try to meet two key requirements for this dictionary: **large** and **consistent**.

For the first one, in stead of directly using the current mini-batch as the dictionary, a queue of data samples is maintained as the dictionary in MoCo. As a result, the size of the dictionary can be larger than the batch size, so that a rich set of negative samples can be covered, which would greatly improve the performance of the representation learning. Regarding **consistency**, it is about how soon can the update of the encoder in the current step has influence on the next few steps of the training. It would be the best if the influence is generated immediately when the next step starts. MoCo ensure this by proposing a momentum update. Specifically, MoCo introduces the concept of a momentum encoder, which uses a moving-average (momentum) update strategy. Denoting the parameters of the query encoder f_q as θ_q and those of the momentum encoder f_k as θ_k, the update of θ_k is formally defined as:

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q, \tag{1}$$

where $m \in [0, 1)$ is a momentum coefficient controlling the exact contribution from θ_k and θ_q. By applying this momentum update, θ_k evolves more smoothly, resulting in better learning performance.

The details of MoCo's training procedure is shown in Alg.1.

Algorithm 1 MoCo Training Algorithm

f_k.params $\leftarrow$ f_q.params ▷ Initialize key network
for each minibatch x in loader **do**
  $x\_q \leftarrow$ aug(x) ▷ Query sample
  $x\_k \leftarrow$ aug(x) ▷ Key sample
  $q \leftarrow f\_q$.forward(x_q) ▷ Queries: $N \times C$
  $k \leftarrow f\_k$.forward(x_k) ▷ Keys: $N \times C$
  $k \leftarrow k$.detach() ▷ No gradient to keys
  l_pos $\leftarrow$ bmm$(q$.view$(N, 1, C), k$.view$(N, C, 1))$ ▷ Positive logits: $N \times 1$
  l_neg $\leftarrow$ mm$(q$.view(N, C), queue.view$(C, K))$ ▷ Negative logits: $N \times K$
  $logits \leftarrow$ cat$([l_\text{pos}, l_\text{neg}], \text{dim} = 1)$ ▷ Logits: $N \times (1 + K)$
  $labels \leftarrow$ zeros(N) ▷ Positives are the 0-th
  $loss \leftarrow$ CrossEntropyLoss$(\frac{logits}{t}, labels)$ ▷ Contrastive loss, Eqn.(1)
  Backward and update: f_q.params
  f_k.params $\leftarrow m \times f\_k$.params $+ (1-m) \times f\_q$.params ▷ Momentum update: key network
  Enqueue the current minibatch: enqueue(queue, k)
end for

Regarding pretraining, samples in the training dataset should be augmented with different ways to provide distinct keys: x_k and x_q. We implement this by using an existed package especially designed for MoCo, which is:

$$from \quad lightly.transforms \quad import \quad MoCoV2Transform. \tag{2}$$

This package was created by a computer vision framework for self-supervised learning called: Lightly SSL. It can be used by install lightly with:

$$pip \quad install \quad lightly \tag{3}$$

Specifically, this function $MoCoV2Transform$ would apply different augmentation methods like random cropping, random flipping, color jittering, and etc, to each image to create encoding vector x_k and x_q.

Then, regarding the finetune step,We use a preprocessing technique called **Random Vertical Flip** by the command:

$$torchvision.transforms.RandomVerticalFlip(), \tag{4}$$

for all training samples in the finetune step.

B. simCLR

The SimCLR model proposed by Chen et al. [20] is a self-supervised learning framework designed to learn visual representations without labelled data. The model is based on the conventional ResNet-18 architecture, ending in a global average pooling layer that generates feature representations. We add a projection head to these representations, which consists of a multi-layer perceptron with one hidden layer and produces a 2048-dimensional embedding. The projection head enables the contrastive learning goal, allowing the model to train to optimize the agreement between several augmented perspectives of the same image. A contrastive loss, especially the normalized temperature-scaled cross-entropy loss (NT-Xent Loss), is used to train the model. This loss promotes the model to bring positive pair representations (two augmented versions of the same image) closer together in the embedding space while pushing negative pair representations (two augmented versions of different images) apart, with the augmentations represented by :

$$t \sim \mathcal{T}, t' \sim \mathcal{T} \tag{5}$$

The approach includes a stochastic data augmentation module. The module generates two correlated views, making a positive pair, for any given input image. Random cropping and resizing, colour distortions, and Gaussian blurring are all part of the augmentation process. These actions bring unpredictability into the visual data, imitating the variation seen in real-world events and enabling the model to learn invariant and robust characteristics. Stochastic Gradient Descent (SGD) was used to optimize the model, using an initial learning rate of 0.06. To speed up training, the batch size was set to 256 and data parallelism was used across available GPU resources. Each minibatch was made up of N randomly selected samples, with positive couples forming within the same minibatch. We did not use a memory bank for negative samples to avoid simple solutions and to enhance meaningful representation learning. The model was trained for 100 epochs due to computational efficiency and requirements. To ensure the experiment's practicality under resource-constrained settings, the training was performed on a single GPU. The PyTorch Lightning framework was used to build the implementation, which streamlines the training loop and enables for easy scaling across multiple compute setups. We were able to focus on the model's logic rather than boilerplate training code because to the framework's flexibility. We accomplish this by utilizing an existing package specifically intended for SimCLR:

$$\text{from} \quad \texttt{lightly.transforms.simclr\_transform} \quad \text{import} \quad \text{SimCLRTransform} \tag{6}$$

Preprocessing Techniques The model uses a series of preprocessing procedures before training to condition the input data for best model performance. These preparation methods were deliberately selected to increase data diversity and boost the learning of robust features. Among the techniques are:

Random Resized Crop: This adjustment crops and resizes the image at random, ensuring that the model experiences a diversity of spatial views throughout training. We used this technique to randomly crop the image to 64x64 pixels.

Color Jitter: We applied random variations in brightness, contrast, saturation, and hue to simulate different lighting conditions and color profiles.

Random Grayscale: Images were most likely turned to grayscale to push the model to focus on texture and shape rather than colour. With a chance of 0.2, we inserted a transformation that turns the photos to grayscale.

Random Horizontal Flip: Images are flipped horizontally with a default 50% chance unless specified otherwise.

Normalization: Finally, we normalized the photos using a mean and standard deviation that corresponded to the conventional procedure for pre-trained models, which aids in the stabilization of the training process. We normalized the photos using the mean and standard deviation values provided for each RGB channel.

Using **Compose** function, these preprocessing procedures were integrated into a composite transformation pipeline, which was then applied to each image as it was loaded into the batch. This aggressive augmentation strategy was critical to the success of our self-supervised learning approach since it generated the difficult job of detecting positive couples among diverse views of the same image.

Algorithm 1: BYOL: **B**ootstrap **Y**our **O**wn **L**atent

Inputs :

$\mathcal{D}$, $\mathcal{T}$, and $\mathcal{T}'$	set of images and distributions of transformations
θ, f_θ, g_θ, and q_θ	initial online parameters, encoder, projector, and predictor
ξ, f_ξ, g_ξ	initial target parameters, target encoder, and target projector
optimizer	optimizer, updates online parameters using the loss gradient
K and N	total number of optimization steps and batch size
$\{\tau_k\}_{k=1}^{K}$ and $\{\eta_k\}_{k=1}^{K}$	target network update schedule and learning rate schedule

for $k = 1$ **to** K **do**

  $\mathcal{B} \leftarrow \{x_i \sim \mathcal{D}\}_{i=1}^{N}$ // sample a batch of N images

  for $x_i \in \mathcal{B}$ **do**

    $t \sim \mathcal{T}$ and $t' \sim \mathcal{T}'$ // sample image transformations

    $z_1 \leftarrow g_\theta(f_\theta(t(x_i)))$ and $z_2 \leftarrow g_\theta(f_\theta(t'(x_i)))$ // compute projections

    $z'_1 \leftarrow g_\xi(f_\xi(t'(x_i)))$ and $z'_2 \leftarrow g_\xi(f_\xi(t(x_i)))$ // compute target projections

    $l_i \leftarrow -2 \cdot \left(\frac{\langle q_\theta(z_1), z'_1 \rangle}{\|q_\theta(z_1)\|_2 \cdot \|z'_1\|_2} + \frac{\langle q_\theta(z_2), z'_2 \rangle}{\|q_\theta(z_2)\|_2 \cdot \|z'_2\|_2} \right)$ // compute the loss for x_i

  end

  $\delta\theta \leftarrow \frac{1}{N} \sum_{i=1}^{N} \partial_\theta l_i$ // compute the total loss gradient w.r.t. θ

  $\theta \leftarrow \text{optimizer}(\theta, \delta\theta, \eta_k)$ // update online parameters

  $\xi \leftarrow \tau_k \xi + (1 - \tau_k)\theta$ // update target parameters

end

Output : encoder f_θ

Fig. 1. BYOL algorithm

C. BYOL

The BYOL method is proposed by Jean-Bastien et al. [19]. This method addresses the weakness of contrastive learning by eliminating negative pairs entirely. This is achieved through a dual network architecture comprising the online and target networks. The online network undergoes stages of representation, projection, and prediction, aiming to predict the target network and train the representation encoder. On the other hand, the target network incorporates representation and projection stages, with its weights serving as an exponential moving average of the online network. This creates a more stable and delayed version of the online network.

The loss calculation focuses on minimizing the difference between predicted and actual target network values. Since the online network includes an additional prediction stage compared to the target network, the loss calculation is asymmetric. A secondary loss is computed by exchanging the views between the networks. The final loss function results from the sum of these two losses, optimizing the encoder weights within the online network. An overview of the algorithm is summarized in Fig 1.

The model is implemented using the library from [24]. This implementation uses Resnet-50 as the encoder and uses Adam optimizer.

Augmentation is applied to input images before feeding into the online and target networks. This implementation uses same augmentation techniques as the BYOL paper[19], including colour jitter, random grey scale, random horizontal flip, random crop, random gaussian blur, random crop, and normalization. The implementation detail of augmentation techniques is shown in Fig 2.

Additionally, Random vertical flip is used to further transform the images using the command:

$$torchvision.transforms.RandomVerticalFlip() \tag{7}$$

IV. Experiment Setup

The experimental environment is set up using PyTorch. Specifically, PyTorch and and TorchVisison libraries are used for model training. CUDA is used for members of the team who have access to GPU resources to enhance the speed of processing data and model training using parallel processing.

To prepare the datasets for efficient data handling and batching, subsections of the database are divided into training and validation sets using PyTorch's Dataset and DataLoader classes. For certain databases, PyTorch also supports downloading and importing the datasets directly to DataLoader using built-in libraries. One or more data processing technique is included in

```
DEFAULT_AUG = torch.nn.Sequential(
    RandomApply(
        T.ColorJitter(0.8, 0.8, 0.8, 0.2),
        p = 0.3
    ),
    T.RandomGrayscale(p=0.2),
    T.RandomHorizontalFlip(),
    RandomApply(
        T.GaussianBlur((3, 3), (1.0, 2.0)),
        p = 0.2
    ),
    T.RandomResizedCrop((image_size, image_size)),
    T.Normalize(
        mean=torch.tensor([0.485, 0.456, 0.406]),
        std=torch.tensor([0.229, 0.224, 0.225])),
)
```

Fig. 2. Augmentation technique implemetation

the model. The data is processed and augmented using selected methods such as vertical flip or random crop using Pytorch's Transform function.

Additionally, the investigated model architecture is built using PyTorch's neural network module, specific layers, and activation functions. Pytorch's built-in loss functions and optimizers are also used in model training and evaluation. In the process of training, PyTorch enables the network to move both forward and backward. This involves sending input data through the model, calculating loss based on specific criteria, and then using backpropagation to adjust weights and biases to optimize the model and minimize the losses.

The evaluation criteria involve the comparison of the final accuracy of each model trained with different datasets. The training and validation losses from each model are compared to ensure that the model is not overfitting or underfitting. Additionally, understanding the impact of implementation techniques, such as augmentation and the selection of hyperparameters, on the final accuracy of the model is crucial.

A. Datasets

Tiny ImageNet: The TinyImageNet dataset is a modified subset of the original ImageNet dataset with 800 fewer classes than the ImageNet dataset. The TinyImageNet contains 100 000 images divided into 200 classes. For every class, there are 500 training images, 50 validating images, and 50 test images. Every image in the dataset is downsized to a 64x64 colored image.
EMNIST: The EMINST dataset contains a set of handwritten character digits in 28x28 pixel image format. EMINST digits is a split in this dataset and it contains 280000 images and 10 classes.
Street View House Numbers (SVHN): The SVHN dataset is a real-world image dataset for developing and evaluating machine learning models on the task of recognizing and classifying digits in natural scene images. It is obtained from house numbers in Google Street View images. The dataset includes over 600,000 MNIST-like 32-by-32 digit images for training, validation, and testing. Images in the SVHN dataset are captured in diverse real-world settings, representing various lighting conditions, orientations, and backgrounds.

B. Evaluation Metrics

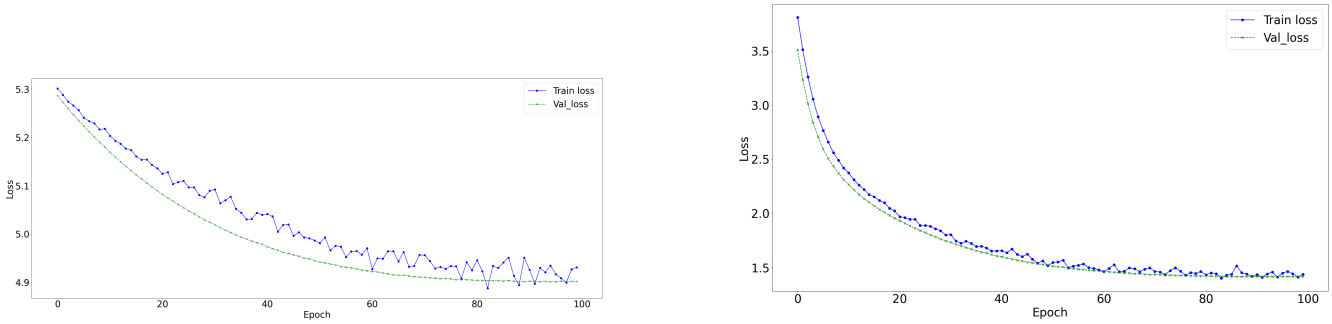
For evaluating models, we use the averaged **accuracy** to measure. Because we are dealing with classification tasks, the definition of accuracy is the number of correctly classified samples over the total classified samples.

V. Results and Analysis

A. Moco

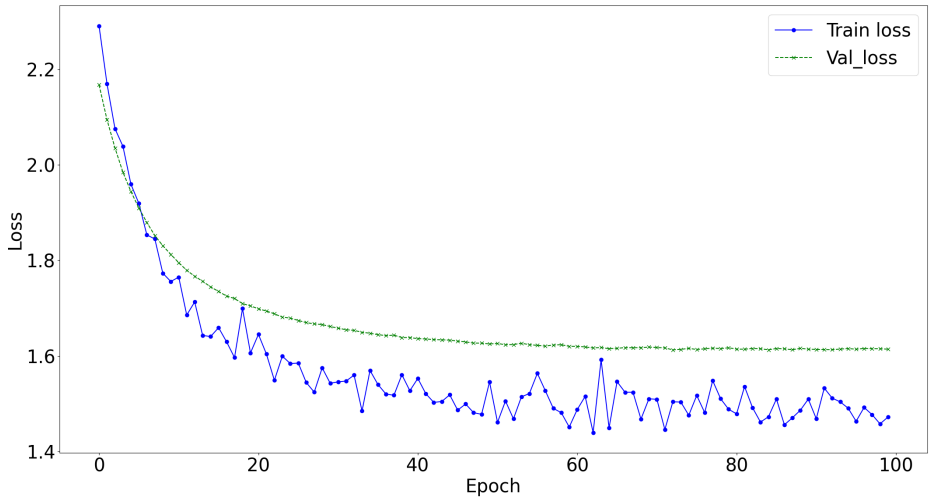
Firstly, We evaluate MoCo on a small dataset, i.e., TinyImageNet, and a medium dataset, i.e., EMNIST. Specifically, as mentioned in the subsection III-A, in the pretraining procedure, we apply two augmentation methods on the images and apply Resnet-18 as the encoder. Then, in the finetune process, we add an additional linear layer to the end of our Resnet model

Fig. 3. Training evaluation of MoCo on two datasets from the scope of small and medium.



(a) TinyImageNet (b) EMNIST

Fig. 4. Training evaluation of MoCo on a large dataset: SVHN



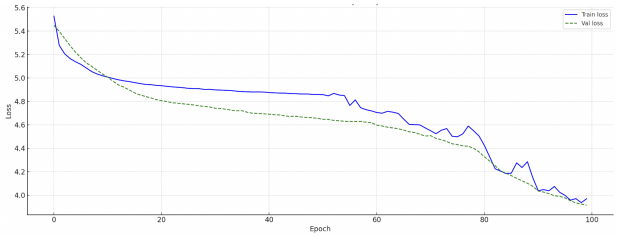
and train it using different image augmentation, i.e., $RandomVerticalFlip()$, with the one in the pretraining process. The losses of the training are shown in Fig. 3. Since both figures show that the validation losses decrease with the reduction of the training losses, there is no overfitting or underfitting problem in the training of MoCo on TinyImageNet and EMNIST. Secondly, we evaluate MoCo on a large dataset, i.e., SVHN. As shown in Fig. 4, similar to other datasets, our training of MoCo on SVHN also did not meet overfitting or underfitting problem. Its final accuracy is 46.2, which is good considering this is a huge dataset.

Besides, regarding the hyperparameters, we use the default setting according to the original paper. However, we found that the adjustment of the learning rate for different datasets is necessary. The default learning rate is 0.06, which turns out to be large for our datasets, because the reduction of the training losses under this setting is not satisfying. Therefore, we try to improve this by decreasing the learning rate gradually to find a balance between the convergence speed and optimization performance. In the end, we apply the learning rate as 0.006 to have a good training within considerable convergence time. Then, we try different batch size for the training. When increasing it, the model can converge faster but has a heavier burden on the memory of the GPU. Thus, to balance these issues, we use 128 for all experiments with MoCo.

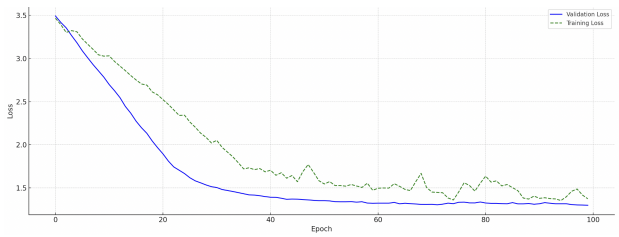
B. SimCLR

The SimCLR model was used to learn representations on two different datasets: Tiny ImageNet, a smaller subset of the larger ImageNet dataset, and EMNIST-digits, a dataset containing handwritten digit characters. On the Tiny ImageNet dataset, the model achieved an accuracy of 6.34%. Given the ImageNet dataset's complexity and breadth, even its smaller subset provides a significant problem due to the high intra-class variation and low inter-class variation. Tiny ImageNet's loss graph shows a consistent decline in both training and validation loss, with the validation loss exhibiting minor oscillations but maintaining a general decreasing trend. This suggests a decent fit, with no obvious evidence of over or underfitting. The model attained a substantially higher accuracy of 58.6% on the EMNIST-digits dataset. This is a significant improvement that displays the model's ability to capture features in a more restricted domain, such as handwritten numbers. The matching loss graph showed

Fig. 5. Training evaluation of SimCLR on two datasets from the scope of small and medium.

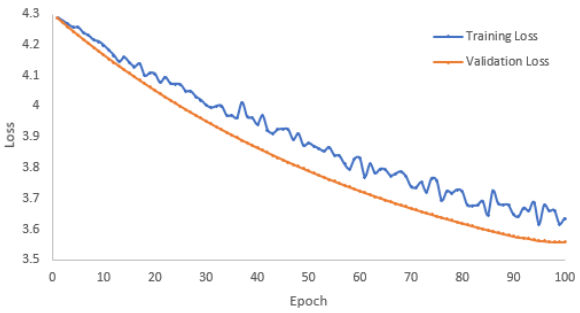


(a) TinyImageNet

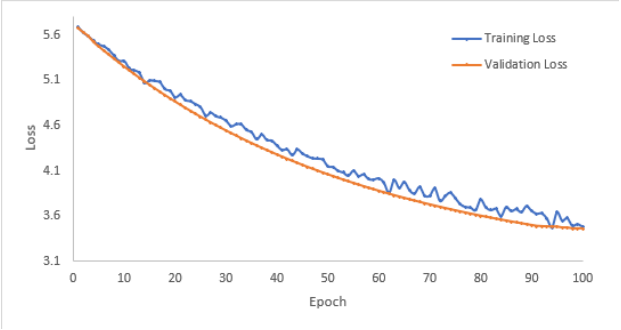


(b) EMNIST

Fig. 6. Training evaluation of BYOL on two datasets from the scope of small and medium.



(a) TinyImageNet



(b) EMNIST

a more prominent and persistent convergence of the training and validation losses, with both measures rapidly dropping. The close relationship between validation and training losses shows that the model generalizes successfully without overfitting to the training data. The substantial discrepancy in accuracy between the two datasets can be traced to the datasets' complexity and variety. Despite being a smaller version of the ImageNet dataset, Tiny ImageNet contains a broad variety of classes with subtle distinctions, making it a more difficult challenge than EMNIST-digits, which contains a single data modality (handwritten digits).

C. BYOL

TinyImageNet and EMINST-digits are selected as small and medium dataset to train and validate the model. Hyperparameters such as projection size, hidden dimension of MLP for projection and prediction stage, and the moving average decay factor are selected by recommendation of the paper[19]. Therefore, batch size and learning rate are tuned to achieve higher performance in training the model. Through trial and error, it has been found that the model yields the lowest losses with a learning rate set to 5e-5 and 8e-5 for TinyImageNet and EMINST-digits respectively.

It has been tested that a larger batch size will significantly increase the convergence time and computational time of each epoch. Due to limited computational resources, a batch size of 2 has been selected.

The model is trained and validated using small and medium datasets over 100 epochs to verify the overfitting and underfitting problem. Cross Entropy loss function is used in evaluating the validation loss. From Figure 6, it can be seen that the training loss and validation loss both decrease exponentially and converge to a similar value at the end. This shows that the overfitting and underfitting problem does not exist in TinyImageNet and EMINST-digits.

D. Results comparison

The accuracy of them is shown in Tab. I. It is shown that on small dataset TinyImageNet, all methods achieve pretty low accuracy. This is mainly because TinyImageNet has 200 classes to classify but with only 100 thousand samples to learn, which is obviously not enough for our self-supervised learning algorithms. Regarding the medium dataset Emnist, MoCo and SimCLR achieve considerable accuracy, while the performance of BYOL is still not satisfying. The reason MoCo here is better is that it applies a memory bank for negative samples, which is excluded in SimCLR for semantic learning. The performance of the SimCLR model differed dramatically across the two datasets, indicating that, while contrastive learning is powerful, its usefulness is greatly dependent on the character and diversity of the dataset. The final accuracy of BYOL was noticeably lower than the accuracy of the other two models. This may be caused by the selected batch size of 2. Small batch sizes can introduce noises in gradient estimation and may converge to a local maxima, which decreases the model performance. It is

TABLE I
ACCURACY OF OUR VISUAL REPRESENTATION LEARNING METHODS ON TINYIMAGENET AND EMNIST

Acc	TinyImageNet	Emnist
MoCo	7.9	67.9
BYOL	9.6	10.9
SimCLR	6.34	58.6

expected that with higher batch size, such as 128, this model could achieve a similar final accuracy compared to the other two models.

VI. CONCLUSION

In this study, an extensive exploration of Momentum Contrast (MoCo), Bootstrap Your Own Latent (BYOL), and SimCLR methodologies across small and medium datasets was conducted. We started by exploring the historical context of self-supervised learning methods. Then we thoroughly examined how each method was implemented and set up the experiment environment. The final accuracy of each model was analyzed and compared, showing each model's performance for each dataset. The analysis of the results confirmed that the trained model does not contain overfitting or underfitting issues. This reinforces their reliability for practical use.

VII. CONTRIBUTION STATEMENT

All members have spent equal amount of time on the project.

REFERENCES

[1] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3733–3742.

[2] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[3] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," *arXiv preprint arXiv:1808.06670*, 2018.

[4] N. Saunshi, O. Plevrakis, S. Arora, M. Khodak, and H. Khandeparkar, "A theoretical analysis of contrastive unsupervised representation learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5628–5637.

[5] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, no. 2. Atlanta, 2013, p. 896.

[6] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 132–149.

[7] P. Bachman, O. Alsharif, and D. Precup, "Learning with pseudo-ensembles," *Advances in neural information processing systems*, vol. 27, 2014.

[8] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," *Advances in neural information processing systems*, vol. 32, 2019.

[9] O. Henaff, "Data-efficient image recognition with contrastive predictive coding," in *International conference on machine learning*. PMLR, 2020, pp. 4182–4192.

[10] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer, 2020, pp. 776–794.

[11] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[12] I. Misra and L. v. d. Maaten, "Self-supervised learning of pretext-invariant representations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 6707–6717.

[13] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[14] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[17] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[18] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[19] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

[20] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[21] H. Bao, L. Dong, S. Piao, and F. Wei, "Beit: Bert pre-training of image transformers," *arXiv preprint arXiv:2106.08254*, 2021.

[22] J. Zhou, C. Wei, H. Wang, W. Shen, C. Xie, A. Yuille, and T. Kong, "ibot: Image bert pre-training with online tokenizer," *arXiv preprint arXiv:2111.07832*, 2021.

[23] P. Gao, T. Ma, H. Li, Z. Lin, J. Dai, and Y. Qiao, "Convmae: Masked convolution meets masked autoencoders," *arXiv preprint arXiv:2205.03892*, 2022.

[24] lucidrains, "Bootstrap your own latent (byol), in pytorch," 2023. [Online]. Available: https://github.com/lucidrains/byol-pytorch/tree/master/byol$_pytorch$