Small Batch SimCLR

Seongho Keum
School of Computing
KAIST
Daejeon, South Korea
matabear8@kaist.ac.kr

Taemin Kim
School of Computing
KAIST
Daejeon, South Korea
arminus0402@kaist.ac.kr

Seohyeon Jung
School of Computing
KAIST
Daejeon, South Korea
heon2203@kaist.ac.kr

***Abstract*—As machine learning rapidly emerges as a core technology in the 21st century, unsupervised learning algorithms that are easiest to apply to real life are becoming more important. Among them, the most representative algorithm is SimCLR(A simple framework for contrastive learning of visual representations), which has the disadvantage of being difficult to use at the personal level because it requires a large batch size. In order to compensate for this disadvantage, in this study, We use a small batch size model which uses a dataset to which various hyperparameters such as learning rate and projection head model size are applied. And we conducted research to make this model perform at a similar level to the original large batch size model. A total of 4 experiments were conducted, a method using a feature(z) that passed through the projection head for classification, a different activation function, a different optimization tool, and the projection head layer(g) to deeper stacking was used. As a result, the accuracy of the batch size 128 model using the lars optimizer instead of adam, change the projection head layer to 4 was 70.67%, and it showed a similar level of performance to the 71.59%, which is the accuracy of the existing batch size 256 model. Through this, the unsupervised learning algorithm can be used with sufficiently efficient performance even in a small batch size, so it is expected that it will be universalized so that it can be trained on the GPU of a personal computer.**

I. Introduction

Representative fields of machine learning include supervised learning, unsupervised learning, and reinforcement learning. Among them, unsupervised learning is learning a network model without data labels, and since most data on the Internet is in an unlabeled form, it is considered the future direction of machine learning. The most representative of these unsupervised learning algorithms is SimCLR(A simple framework for Contrastive Learning of visual Representations) [1], which is a self-unsupervised learning algorithm that applies contrastive learning to positive and negative samples obtained through data augmentation.

Nevertheless, SimCLR is inconvenient to apply directly in real life. That is, the default value of the batch size is set to 4096 because performance is poor in a small batch size. This requires a high-performance GPU, and is not feasible for training with our GPU.

This paper focuses on supplementing and improving the problems of the original version of SimCLR for the generalization of unsupervised learning algorithms. Furthermore, by studying the use of datasets to which various hyperparameters are applied, such as learning rate and the size of the projection head model, we want to present a method that shows good performance even with a small batch size or small negative pair.

II. Related Works

A. Baseline

Self-supervised learning is a learning technique without humans providing labeled data. Nowadays, self-supervised learning is getting attention because labeling and annotating data is very time-consuming process. Then, without supervision how can model learn visual representation? Contrastive learning is the answer of this question that our baseline paper suggests. Our baseline paper is a simple framework for contrastive learning of visual representation [1] called SimCLR.

Contrastive learning is a learning technique by contrasting positive pairs against negative pairs. These positive and negative pairs are made by various data augmentations. Data augmentation is that transform given input image to generate correlated views. In baseline paper, the authors used three simple augmentations, random cropping, random color distortions and random gaussian blur. According to these data augmentation, positive pairs are pair of images which from the same original input image. Different with that, negative pairs are pair of images which from the different original input image.

To learn visual representation, the authors suggested a simple framework for contrastive learning of visual representation in Fig 1. There is an original image and two separate data augmentation operators are applied to original image x. Then, it inputs augmented images to base encoder network and projection head. Base encoder network and projection head are trained to maximize agreement by minimizing the contrastive loss between vector zi and zj. Because they want different outlooks of the same image to have similar representations. Contrastive loss used in baseline paper is NT-Xent loss which is normalized temperature-scaled cross-entropy loss. After training process, projection head is removed. For downstream task, they use encoder and representation h.

However, there is a problem that original SimCLR need very large batch size like 4096 which is not feasible for typical GPU. In Fig 2, it shows that the accuracies are sharply decreases when batch size is decreased. The reason is that contrastive learning benefits more from larger batch sizes.

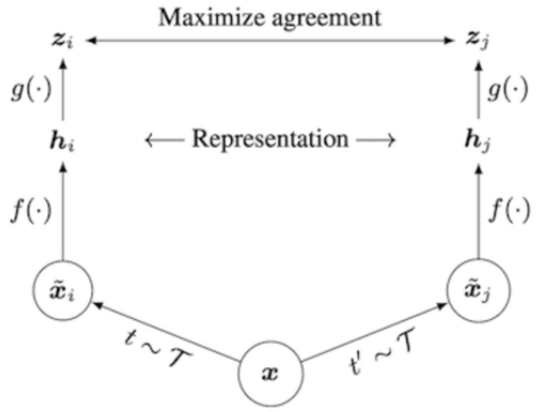


Fig. 1. A simple framework for contrastive learning of visual representation. Each notation in figure 1 means that original image x, augmentation operators t T and t' T, augmented images xi and xj, base encoder network f, projection head g, representation h and vector zi and zj.

Bigger batch size can provide more negative pairs and it can help for fast convergence.

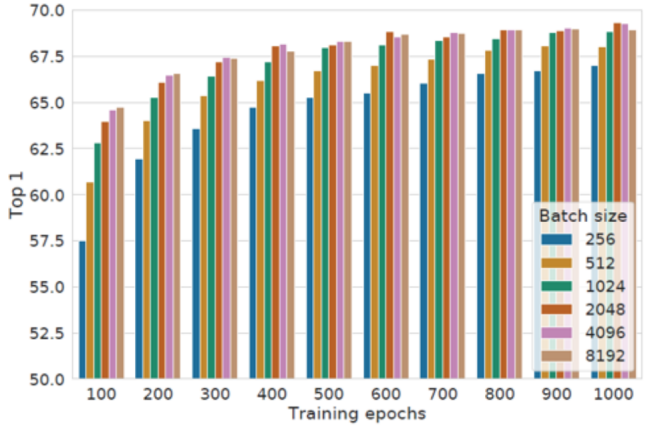


Fig. 2. Linear evaluation models trained with different batch size and epochs. [1]

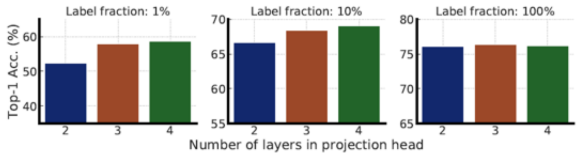
B. SimCLRv2

There is a modification of SimCLR which is SimCLRv2. SimCLRv2 is Big Self-Supervised Models are Strong Semi-Supervised Learners [4]. It is a semi-supervised learning method for learning from few labeled examples while making best use of a large amount of unlabeled data. The main differences between original SimCLR and SimCLRv2 are network and projection head.

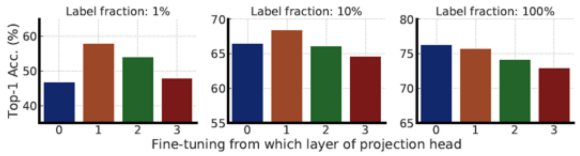
The key ingredient of this paper is the use of deep and wide networks during pretraining and fine-tuning. In case of base encoder network, they use larger ResNet models. In previous work like SimCLR, their largest model is ResNet-50. However, in SimCLRv2, the largest model trained is ResNet-152. By using deep and wide network, SimCLRv2 can get 29% relative improvement in top-1 accuracy when fine-tuned on 1

Depth	Width	Use SK [28]	Param (M)	Fine-tuned on 1%	Fine-tuned on 10%	Fine-tuned on 100%	Linear eval	Supervised
50	1×	False	**24**	**57.9**	**68.4**	**76.3**	**71.7**	**76.6**
		True	35	64.5	72.1	78.7	74.6	78.5
	2×	False	94	66.3	73.9	79.1	75.6	77.8
		True	140	70.6	77.0	81.3	77.7	79.3
101	1×	False	43	62.1	71.4	78.2	73.6	78.0
		True	65	68.3	75.1	80.6	76.3	79.6
	2×	False	170	69.1	75.8	80.7	77.0	78.9
		True	257	73.2	78.8	82.4	79.0	80.1
152	1×	False	58	64.0	73.0	79.3	74.5	78.3
		True	89	70.0	76.5	81.3	77.2	79.9
	2×	False	233	70.2	76.6	81.1	77.4	79.1
		True	354	74.2	79.4	82.9	79.4	80.4
152	3×	True	**795**	**74.9**	**80.1**	**83.1**	**79.8**	**80.5**

Fig. 3. In case of base encoder network, they use larger ResNet models



(a) Effect of projection head's depth when fine-tuning from optimal middle layer.



(b) Effect of fine-tuning from middle of a 3-layer projection head (0 is SimCLR).

Fig. 4. Also, the authors claim that increasing the capacity of the non-linear network, projection head, is important

Also, the authors claim that increasing the capacity of the non-linear network, projection head, is important. Furthermore, in case of SimCLR, they remove projection head after training process. Different with that, SimCLRv2 just fine-tuning from a middle layer. In result, compared to SimCLR with 2 layers projection head, by using 3 layers projection head and fine-tuning from the 1st layer of projection head, SimCLRv2 can get 14% relative improvement in top-1 accuracy when fine-tuned on 1

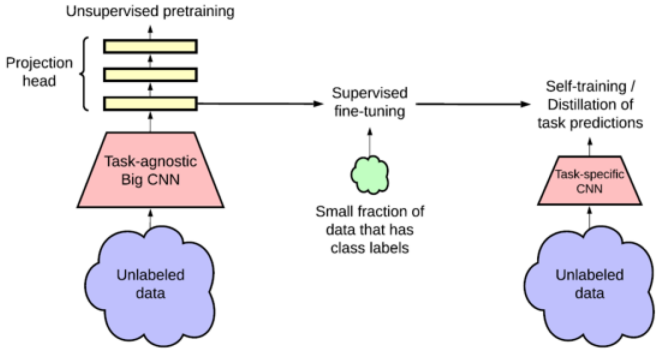


Fig. 5. The main differences between original SimCLR and SimCLRv2 are network and projection head

C. MoCo

MoCo(Momentum Contrast for unsupervised visual representation learning) [2], like SimCLR, is one of the unsupervised learning algorithms to which contrastive learning is applied. Here, contrastive learning is a type of unsupervised learning that learns a network model without data labels. It is a method of learning a model in a direction that increases the similarity between positive pairs and decreases the similarity between negative pairs. In order to perform contrastive learning, positive and negative pairs must be extracted. In SimCLR, positive and negative pairs were extracted in one batch. On the other hand, MoCo first defines a dictionary of a certain size and stores the key values of samples. Among them, values matching the query are extracted as positive keys, and the remaining values are extracted as negative keys.

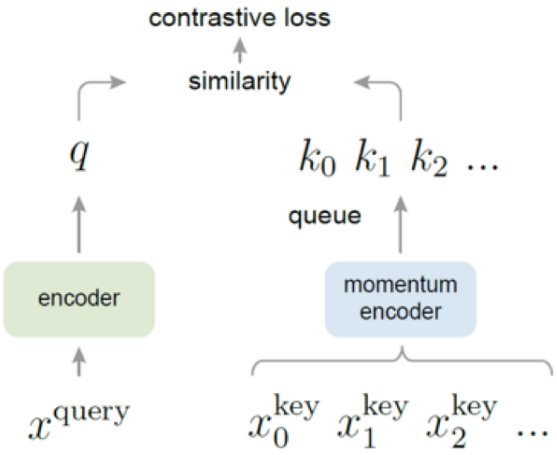


Fig. 6. Structure of MoCo [2]

The structure of MoCo is shown in Fig 6. Query and key for each image are generated by using query encoder and key encoder, respectively. Encoder uses CNN-based network, and MoCo uses ResNet. A query generated from one image consists of a key and a positive pair created from the same image, and a key and a negative pair created from another image.

For good contrastive learning, a sufficient number of negative pairs is required. MoCo puts a sufficiently large dictionary in the form of a queue, and extracts negative samples from the dictionary. The size of the dictionary is generally set larger than the batch size, and in this paper, 65536 is set as the default value. If we study in units of 256 batches, we create a query and a key for one input image to form a positive pair, and form a negative pair with 65536 keys already in the dictionary. Since the dictionary has a queue structure, after learning for 256 batches is completed, the 256 keys generated in the learning process will replace the oldest 256 keys.

Using a method of using a large dictionary of queue structures also has its drawbacks. In the process of learning the key encoder, it is difficult to learn because the gradient has to be propagated to many negative samples. In MoCo, to solve this problem, the learned query encoder was imported and used as it is without training the key encoder separately. Here, we used a method of slowly evolving the key encoder by applying momentum update to maintain the consistency of key values.

In the past, contrastive loss was created in two ways. The end-to-end method (a method that trains the key encoder in the same way as the query encoder) and the memory bank method (puts the representation of all samples in the data set in the memory bank, selects a few samples arbitrarily and constructs a dictionary, then updating the memory bank with samples corresponding to the query). Unlike this, MoCo can learn from a sufficient amount of negative samples because it uses a dictionary using a queue structure. It is also memory-efficient and has the advantage of being able to learn stably even when the size of the dataset is large.

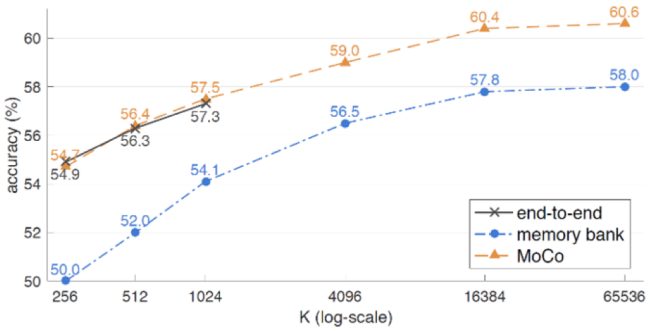


Fig. 7. K-Accuracy graph of three method(end-to-end, memory bank, MoCo) [2]

Fig 7 shows the results for the three methods. The end-to-end method shows similar performance to MoCo, but it was not possible to increase the batch size above 1024. Memory bank method shows low performance due to inconsistency issue. On the other hand, unlike the two methods, MoCo shows good performance because it can learn even in a large batch size and the keys of the data are consistent.

D. BYOL

The previously introduced contrastive-based unsupervised learning algorithms had some problems to train stably, such as having to select a negative pair well, learning in a large batch size, and having a large performance deviation even in the image augmentation option used for training. On the other hand, BYOL(Bootstrap Your Own Latent) [3] achieved better performance than conventional methods applying contrastive learning without using negative pairs. It is a method of using two networks instead of a method of using two images when learning a high-quality representation from images.

In this paper, a simple experiment was conducted first. In Step 1, the network A was randomly initialized and the weights were fixed. Here, the accuracy of the ImageNet dataset was measured through the linear evaluation protocol, which is the evaluation protocol of Self-Supervised Learning. That is, after random init, one linear layer is attached to the frozen feature

extractor, trained with the ImageNet dataset, and then the accuracy is measured. In this case, it is said that good performance cannot be obtained because the feature extractor only attaches a linear layer without learning any information, and it is said that the top-1 accuracy of 1.4% is actually achieved. In Step 2, the unlabeled dataset is fed forward to a random initialized A network + MLP to obtain predictions. Finally, in Step 3, one network called B is prepared. B is also subjected to random initialization in the same way, but this time, instead of directly performing linear evaluation, the images are fed forward to the A network, and the extracted prediction is used as a target to learn this target. Surprisingly, network B was trained to learn the inaccurate predictions from network A randomly initialized, and when linear evaluation was performed, top-1 accuracy of 18.8% was obtained. Of course, this number is a very low number, but the performance increased significantly compared to when random initialization was performed. The authors describe that this simple experiment became the core motivation for BYOL.

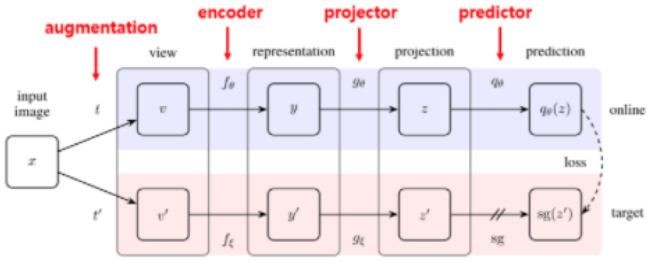


Fig. 8. Architecture of BYOL [3]

BYOL uses two networks, an online network and a target network. Each network consists of an encoder, a projector, and a predictor, and the target network creates a regression target for the online network to learn. And the weight of the target network is again determined by the exponential moving average of the weights of the online network. In this way, we apply different augmentation to the two networks to select a feature vector (prediction), then normalize l2 and train the online network in the direction of minimizing the mean squared error.

In BYOL, first, for the ImageNet dataset, pre-training with the Self Supervised Learning method with the ImageNet unlabeled dataset, then freezing the encoder, and a linear evaluation experiment to train the linear classifier, and a feature extractor with a part of the labeled training set A semi-supervised training experiment for fine-tuning was performed.

First, the results of the linear evaluation experiment are shown in Table 1 of the figure above, and it achieved better performance than previous studies such as SimCLR, MoCov2, and InfoMin Aug. Semi-supervised setup shows experimental results when fine-tuning is performed with 1% and 10% training samples, and shows superior performance compared to existing methods.

Through these experiments, the authors said that although BYOL showed performance improvement, it was still dependent on data augmentation, and the limitation was that data augmentation was also limited to vision applications. In order to apply BYOL to other applications(audio, video, text), the next step would be to obtain an appropriate augmentation technique for each, and this thesis presented the direction for the future.

Method	Top-1	Top-5
Local Agg.	60.2	-
PIRL [32]	63.6	-
CPC v2 [29]	63.8	85.3
CMC [11]	66.2	87.0
SimCLR [8]	69.3	89.0
MoCo v2 [34]	71.1	-
InfoMin Aug. [12]	73.0	91.1
BYOL (ours)	**74.3**	**91.6**

(a) ResNet-50 encoder.

Method	Architecture	Param.	Top-1	Top-5
SimCLR [8]	ResNet-50 (2×)	94M	74.2	92.0
CMC [11]	ResNet-50 (2×)	94M	70.6	89.7
BYOL (ours)	ResNet-50 (2×)	94M	77.4	93.6
CPC v2 [29]	ResNet-161	305M	71.5	90.1
MoCo [9]	ResNet-50 (4×)	375M	68.6	-
SimCLR [8]	ResNet-50 (4×)	375M	76.5	93.2
BYOL (ours)	ResNet-50 (4×)	375M	78.6	94.2
BYOL (ours)	ResNet-200 (2×)	250M	**79.6**	**94.8**

(b) Other ResNet encoder architectures.

Fig. 9. Top-1 and top-5 accuracies (in %) under linear evaluation on ImageNet [3]

Method	Top-1		Top-5	
	1%	10%	1%	10%
Supervised [64]	25.4	56.4	48.4	80.4
InstDisc	-	-	39.2	77.4
PIRL [32]	-	-	57.2	83.8
SimCLR [8]	48.3	65.6	75.5	87.8
BYOL (ours)	**53.2**	**68.8**	**78.4**	**89.0**

(a) ResNet-50 encoder.

Method	Architecture	Param.	Top-1		Top-5	
			1%	10%	1%	10%
CPC v2 [29]	ResNet-161	305M	-	-	77.9	91.2
SimCLR [8]	ResNet-50 (2×)	94M	58.5	71.7	83.0	91.2
BYOL (ours)	ResNet-50 (2×)	94M	62.2	73.5	84.1	91.7
SimCLR [8]	ResNet-50 (4×)	375M	63.0	74.4	85.8	92.6
BYOL (ours)	ResNet-50 (4×)	375M	69.1	75.7	87.9	92.5
BYOL (ours)	ResNet-200 (2×)	250M	**71.2**	**77.7**	**89.5**	**93.7**

(b) Other ResNet encoder architectures.

Fig. 10. Semi-supervised training with a fraction of ImageNet labels [3]

III. Methods

We now present our method for improving accuracy of simclr model for low batch size. First, we tried to set baseline model with high batch size such as 1024, 2048. But we failed due to 'out of memory' error. (We used extra server which includes 'Quadro RTX 6000'). So we set baseline with batch size 256, and tried to catch up this with lower batch size 128. Below are 4 methods that we have tried to improve out model.

- Use feature z that passed projection head for classification
- Use other activation functions
- Use other optimizer
- Stack the projection head layers deeper.

Some of the methods are succeeded but some don't. We have choose these 4 method based on following reasons.

- We want to clarify the effect of feature z which passed projection head. Although paper insists that feature h(feature before passing the projection head) has better expressiveness, but we want to check this is valid even in lower batch size.
- The original paper uses 'Relu' in the projection head. As 'Leaky Relu' get spotlight for its good performance, we also try it for batch size 128.
- The paper 'intriguing properties of contrastive losses' suggests lars optimizer may be good for lower batch size, and effective for stabilizing performance between different kinds of contrastive losses.
- There are many similar studies with deep projection head layers. From this, we tried to prove that deep layers are effective for accuracy.

Projection head	Batch size	Epoch			
		100	200	400	800
2 layers	512	65.4	67.3	68.7	69.3
	1024	65.6	67.6	68.8	69.8
	2048	65.3	67.6	69.0	70.1
3 layers	512	66.6	68.4	70.0	71.0
	1024	66.8	68.9	70.1	70.9
	2048	66.8	69.1	70.4	71.3
4 layers	512	66.8	68.8	70.0	70.7
	1024	67.0	69.0	70.4	70.9
	2048	67.0	69.3	70.4	71.3

Fig. 11. Effect of deep layers from "intriguing properites of contrastive loss" (tested with ImageNet) [5]

IV. Experiments and Results

Detail information of baseline-batchsize:128&256, layers of projection head:2, epoch for SimCLR model:40, epoch for classification linear layer:100, optimizer:Adam

We have set baseline using batch size 128, 256. The goal of our study is to make batch size 128 accuracy similar to batch size 256 accuracy by not increasing the batch size. Below is table for baseline model accuracy.

	Batch size	Epochs	Projection layers	Accuracy
Baseline 128	128	40	2	68.59
Baseline 256	256	40	2	71.59

TABLE I

- Use feature z that passed projection head for classification

	Batch size	Epochs	Projection layers	Accuracy
Baseline 128	128	40	2	68.59
SimCLR z	128	40	2	67.54
Baseline 256	256	40	2	71.59

TABLE II

As the 'simclr' paper suggested, feature z has bad representation. So, its accuracy has degraded compared with baseline model.

- Use other activation functions

	Batch size	Epochs	Projection layers	Accuracy
Baseline 128	128	40	2	68.59
Baseline 128 tanh	128	40	2	67.95
Basline 128 LeakyReLu	128	40	2	68.84
Baseline 256	256	40	2	71.59

TABLE III

As expected, activation function doesn't have big impact for accuracy. We have used 'tanh', 'leaky relu' for experiments and they got almost same accuracy with baseline within 1% of error range.

- Use other optimizer

	Batch size	Epochs	Projection layers	Accuracy
Baseline 128	128	40	2	68.59
Baseline 128 LARS	128	40	2	69.85
Baseline 256	256	40	2	71.59

TABLE IV

Following the paper 'intriguing properties of contrastive losses', we tried with 'lars' optimizer. Surprisingly, it gives us better result than baseline. However, lars optimizer won't be the 'universal' key for upgrading accuracy. It might be have good relation with 'nt xent' loss.

- Stack the projection head layers deeper. Stacking more

	Batch size	Epochs	Projection layers	Accuracy
Baseline 128	128	40	2	68.59
Baseline 128 layer3	128	40	3	69.53
Baseline 128 layer4	128	40	4	70.25
Baseline 256	256	40	2	71.59

TABLE V

layers for projection head have significant effect for accuracy. Due to resource time limit, we only experienced with 34 layers. We think deeper layer would be effective for not only STL but for other datasets.

- Final model

	Batch size	Epochs	Projection layers	Accuracy
Baseline 128	128	40	2	68.59
Baseline 128 LARS layer4	128	40	4	70.67
Baseline 256	256	40	2	71.59

TABLE VI

We have used two effective methods. First is using lars optimizer and second is deepening projection head layer. Mixing these two properties, we got 70.67 accuracy which is a lot higher than baseline with 128 batch size. Also, it is quite similar with batch size 256 accuracy. Following are detail configuration for our 'final model'. Detail information of final model-¿ batchsize:128, layers of projection head:4, epoch for simclr model:40, epoch for classification linear layer:100, optimizer:lars

V. Conclusion

In case of original SimCLR, there is a problem that the accuracy decreases sharply as the batch size decreases. However, the maximum batch size is that our GPU can handle is 256. Therefore, we set the result with batch size 256 as baseline and try to find out the environment where the accuracy becomes similar to 256 when the batch size is 128. In this work, we present advanced SimCLR version applied four solutions: use feature that passed projection head for classification instead of representation, use other activation functions instead of ReLu like tanh and LeakyReLu, use other optimizer LARS instead of Adam, stack the projection head layers deeper.

First of all, we observe that using feature that passed projection head for classification has lower accuracy than using representation. Then, using other activation functions instead of ReLu, we can get the highest accuracy with ReLu. Next, using LARS optimizer has higher accuracy than Adam optimizer. Lastly, stacking more projection head layers shows the deeper the depth, the higher the accuracy was. To summarize, we can get best performance when we use representation for classification, ReLu activation function, LARS optimizer and 4 layers of projection head.

Due to lack of our resource time, we can't train our model in more epochs. I think we can get more precise accuracy when we train our model with higher epochs. Also from batch size 512, we experienced 'out of memory error'. If we get better gpu resource later, we want to clarify that our results is valid even in higher batch size. Using mixed precision which uses both float 16, float 32 for gpu calculation might be a good solution for out of memory error. We will experiment it later.

References

[1] Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." International conference on machine learning. PMLR, 2020.

[2] He, Kaiming, et al. "Momentum contrast for unsupervised visual representation learning." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

[3] Grill, Jean-Bastien, et al. "Bootstrap your own latent: A new approach to self-supervised learning." arXiv preprint arXiv:2006.07733 (2020)

[4] Chen, Ting, et al. "Big self-supervised models are strong semi-supervised learners." arXiv preprint arXiv:2006.10029 (2020).

[5] Chen, Ting, and Lala Li. "Intriguing properties of contrastive losses." arXiv preprint arXiv:2011.02803 (2020)