Project Report : CS 4803/7643 Spring 2020

Raymond Ortiz
rortiz9@gatech.edu

Karan Khosla
kkhosla7@gatech.edu

Tanvi Bhagwat
tbhagwat6@gatech.edu

Sahil Dhingra
sdhingra31@gatech.edu

Georgia Institute of Technology
North Ave NW, Atlanta

Abstract

This paper presents several new applications and modifications to Facebook's Momentum Contrast (MoCo) for unsupervised visual representation learning [6]. We explore how MoCo operates in the domain of small image datasets, how well it performs on retraining with different optimizers and performance improvements by adding visual attention on top of the existing architecture. Our notable results include the change of optimizers where we found that SGD is outperformed by RMSprop. We also found that as it currently is implemented, visual attention does not work as well with MoCo as we had expected.

1. Introduction/Background/Motivation

Image classification is a task where, given an arbitrary image, a model should be able to select the correct class or category from a set of options. There are two general machine learning approaches to this problem: supervised learning and unsupervised learning. The former requires a human annotator to hand-label each image in the training set, and the model uses these ground-truth labels to learn; the latter does not have this requirement and can operate on unstructured data. In the world of image classification, supervised learning is dominant. In particular, on small datasets such as CIFAR-10, unsupervised learning is significantly behind supervised learning performance-wise. Our goal is to take a recent success in unsupervised learning and apply it in a different environment, one to which it is not necessarily best suited.

1.1. Current Practice

Current state-of-the-art image recognition on small datasets spans a wide variety of methods. The best result on CIFAR-10 is 99% accuracy [12]. The current state-of-the-art in unsupervised learning for CIFAR-10 is only 86% [11]. Notably, there are dozens of supervised learning papers that achieve better results than this.

TResNet uses several customized layers to improve both accuracy and speed. It replaces all downscaling layers with an economical variant of Anti-Alias Downsampling [14], replaces all BatchNorm+ReLU layers with Inplace-ABN [1], and some Bottleneck Blocks [8]. The (Paine et al. (2015)) paper utilizes large ratios of 100:1 of unsupervised pre-training and supervised fine-tuning data, but not solely unsupervised samples. Notably, they also found that lower ratios of unsupervised data to supervised data actually hurt accuracy.

1.2. Importance Of Work

Labeling images is both costly and tedious, and it is generally avoided when possible. Unsupervised learning provides us with the means to bypass these issues, so if we can provide better results in this domain, it will save many researchers significant amounts of time and other resources.

1.3. Data

We used several well-known and widely used datasets. CIFAR-10, STL-10, and SVHN were our primary datasets. Every one of these was chosen because they are widely used as benchmarks for computer vision papers. There was no preprocessing of the images save for the sorting MoCo required. For each of these we had to take the images and sort them by their labels in different sub-directories so that MoCo would access them correctly. When the images were not stored in different folders inside the validation folder, the network learned only one label and would report 100% accuracy.

CIFAR-10 consists of 10 categories of images, each category having 6000 samples [10]. It is entirely self-contained and was created by the Canadian Institue For Advanced Re-

search for the exact purpose of being used in image recognition software.

STL-10 was inspired by CIFAR-10 but differs in that it has very little labeled data (500 training, 800 testing per class) and 100,000 unlabeled images [2]. There are 10 classes among the labeled images but many other classes among the unlabeled images. We chose this dataset to see the difference between how MoCo operated a labeled dataset (CIFAR) and a primarily unlabeled datset (STL) because of the successes seen in the (Paine et al. (2015)) paper.

1.4. Contrastive Learning

One of the core facets of MoCo is that it uses contrastive loss instead of the more typical L1 or L2 loss functions. Contrastive loss functions give lower values when the similarity between the given data point and its expected value is high *and* the similarity between the data point and all other values is low [5]. The MoCo paper utilizes the InfoNCE loss specifically [13].

2. Approach

2.1. MoCo Explained

Before understanding our approach, one should understand how MoCo works. Unsupervised learning is structured to "perform dictionary look-up: an encoded query should be similar to its matching key and dissimilar to others" [6]. MoCo takes mini-batch samples from the current set of keys, encodes them, and then enqueues them to form the dynamic dictionary. Every iteration takes in a new mini-batch of keys and dequeues the oldest mini-batch. This dictionary is then compared to the encoded query to generate the contrastive loss. The dynamic dictionary allows for comparison with a much larger set of data than the usual mini-batch comparisons, and the constant removal of the oldest keys allows for better updating [6].

The downside to using such a large dictionary is that it makes updates much more difficult. To counter this issue, MoCo uses the momentum update

$$\theta_k = m\theta_k + (1-m)\theta_q$$

where θ_k is the parameters for the key encoder, θ_q is the parameters for the query encoder, and m is the momentum hyperparameter (usually a high value close to 1 like .999). The large momentum allows for a very smooth change in the keys over time, likely a core requirement of using a queue as the dictionary [6].

2.2. Our Approach

Facebook AI Research (FAIR) published the code they used to run the experiments from the original paper [7]. We built our modifications on top of their repository, changing the experiment parameters and expanding the model architecture. As described below, we wrote custom code to load alternative datasets, since the Facebook repository is hard-coded to use ImageNet. We also swapped out the optimizer they used, as the only option they give is to use Stochastic Gradient Descent (SGD). Lastly, we provided our own code to provide the encoder models; the Facebook code was set up to only take in torchvision models, so we wrote our own extension of torchvision's ResNet50 that includes visual attention layers and injected it into the MoCo model.

Our goal with this paper is to benchmark the performance of the latest unsupervised learning method MoCo on a smaller and unseen dataset. We use pre-trained MoCo weights for the fully connected layer with a ResNet50 model and tune it to CIFAR-10 through transfer learning, as it has not been used for performance evaluation in the original paper. We also benchmark its performance against a similar ResNet50 model, which we tuned using supervised learning. Using the supervised learning procedure, we also test the effect of different image normalizations, to be utilized for appropriate tuning of pre-trained MoCo weights on CIFAR-10 data. Using the best hyperparameters, we also validate this model on STL-10 dataset to compare its performance with the latest public benchmarks.

Additionally, we also evaluate the effect of using a different optimizer for learning MoCo weights, with a pre-trained ResNet50 model through unsupervised training procedure on CIFAR-10 dataset. While the MoCo paper utilizes SGD for the unsupervised training procedure, we test the improvement in performance by using the optimizers Adam, Adagrad and RMSprop. Since the MoCo paper uses ImageNet dataset for training, we train a benchmark using SGD on CIFAR-10 as well.

The original paper utilized Stochastic Gradient Descent as its optimizer. SGD performs a parameter update once every iteration for each training sample as shown here.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

This paper compares the performance of three other optimizers against SGD. Adagrad is a gradient-based optimization that performs smaller updates to the parameters that are associated with frequently occurring features, and larger updates for the infrequent ones [3]. As Adagrad updates the parameters with different gradients, g_t is used to denote it:

$$g_{t,i} = \nabla_\theta J(\theta_{t,i})$$

Then each parameter is updated with:

$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}$$

For its update rule, using the past gradients for each parameter, along with the learning rate, the modified gradient is

calculated [3]:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

RMSprop is identical to Adadelta's update vector:

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

Adaptive Moment Estimation or Adam utilizes adaptive learning rates for its parameters [3]. In combination with momentum, these are used to update the parameters:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Finally, we were also inspired to try adding visual attention modules to improve the baseline MoCo performance. This concept uses intermediate representations within the model to boost the relevant spacial regions of the image and suppress the irrelevant ones [9]. Visual attention has seen success with supervised image classification, so we thought it might be worthwhile to try it with unsupervised learning. The technique may fit naturally within the overall framework, as attention could force the loose categories generated by MoCo to be more meaningful, focusing on distinct visual features of the image. Attention also provides the added benefit of extra interpretability, as they can easily be translated to heat maps that show which parts of the image are important to the model.

2.3. Trials And Tribulations

We expected to encounter a few challenges in carrying out the experiments, some due to lack of available solutions to problems while reproducing results since the paper was published quite recently, and others around getting the datasets in the right shape and format.

The main problems we encountered had to do with software challenges in implementation and as anticipated earlier, getting the data in the desired format. Since the procedure has been implemented using multiple GPUs and distributed systems, it does not run on a local system which lacks distributed computing capabilities. Therefore, we used Google Colab notebooks which solved this problem. However, another issue we encountered while using Colab was by using faster TPU functionality which did not work properly with MoCo, and we had to fall back to using GPUs for limited periods of time. In terms of the dataset, the default implementation expected images to be passed in .png format and not tensors or numpy arrays. To deal with this, we had to find and download a directory with CIFAR-10 data in .png format. Getting the STL-10 dataset in the .png format and the appropriate class labeling took additional time. We had to discard the SVHM dataset due to the time required to get the data in the required form.

After successfully downloading and getting the data in the right format, we encountered another challenge where all of our models returned 100% accuracy on both CIFAR-10 and STL-10 datastes. Only with a very small learning rate, it seemed to go lower than 100%. After hours of investigation, it turned out to be an issue with our data storage format which was not in the expected PyTorch's ImageFolder format.

Adding visual attention layers was also expected to be challenging from a software engineering perspective. We were torn between either rebuilding ResNet50 from scratch or attempting to inject the new layers into PyTorch's existing ResNet implementation. We ultimately chose the latter, and doing so indeed required pulling apart PyTorch's API and some complicated experimentation.

We landed using only the MoCo repository that FAIR published [7]. We briefly considered utilizing some popular ResNet50 implementations for comparison but ultimately discarded the idea due to our lack of sustained access to GPUs.

3. Experiments and Results

We measured success in two ways. Our first goal was to show the viability of our modifications to MoCo, which meant our first metric was to compare our results to those of the original paper. Additionally, in the spirit of the MoCo paper, we also wanted to achieve a favorable comparison with supervised learning techniques, thereby demonstrating the effectiveness of unsupervised learning in general.

With that in mind, our first step was to train an unmodified MoCo on our datasets, establishing the main baseline for our extensions.

In this section, we tuned a ResNet50 model with pretrained MoCo dictionary on the CIFAR-10 dataset. The image size for CIFAR-10 images was 32x32, compared to 224x224 for ImageNet. Using the original images at 32x32 for CIFAR-10 dataset resulted in a significantly lower accuracy, hence default cropping, resizing and normalization parameters were restored for the procedure using the pretrained modules. We use the same hyperparameters as the paper for validation on new datasets except for a few changes. Due to memory constraints, a batch size of 64 was used instead of 256. Training was done over 10 epochs due to time constraints and a single GPU was used due to Colab's functionality instead of the 8 that MoCo is designed to run on.

For baseline purposes, two experiments were run. The first experiment validated the effect of normalizing CIFAR-10 images with the mean and standard deviation from Imagenet data versus normalizing with the CIFAR-10 mean and

standard deviation by using transfer learning with a standard ResNet50 model. Normalizing with Imagenet parameters results in much better accuracy, as the model is trained on the significantly larger Imagenet dataset. The ImageNet parameters yield a top-1 accuracy of 78%, while the CIFAR-10 parameters yeild an accuracy of 70%. Therefore, the ImageNet parameters were used for the second experiment as well.

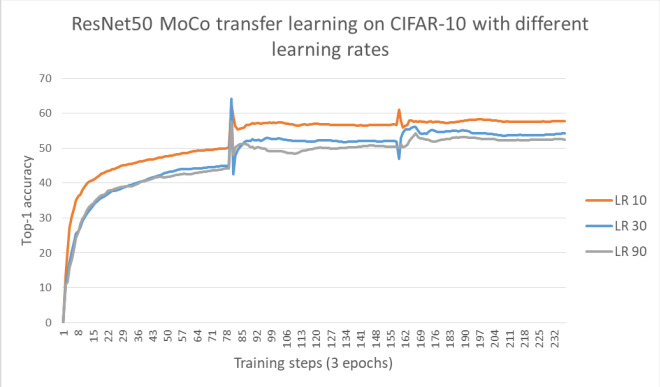

Figure 1. Top-1 Accuracy of ResNet50 MoCo (CIFAR-10)

The second experiment compares the performance of models tuned using transfer learning - ResNet50 using MoCo pre-trained module versus a ResNet50 model with a fully connected layer and a softmax layer. The ResNet50 MoCo model achieves a top-1 accuracy of 69.3% compared to 78% for the standard ResNet50 model. Through hyperparameter testing, as shown in 1, a learning rate of 10 shows better accuracy compared to 30 over 10 epochs. Thus, the difference between the standard model compared to model with MoCo can be expected to decrease with further training and hyperparameter tuning. Even though MoCo doesn't outperform the standard architecture, the difference is not huge and demonstrates that pre-trained MoCo is powerful enough for an unsupervised method on the CIFAR-10 dataset.

Next, we tried pre-training the model with each of our proposed changes.

Since the original paper only utilized Stochastic Gradient Descent as an optimizer, Adam, Adagrad and RMSprop were used in separately to compare the results against the default Stochastic Gradient Descent to measure the average loss per epoch. Using the built-in PyTorch optimizers, we pre-trained with CIFAR-10. With the same hyperparameters as used originally, we trained for 20 epochs, with a learning rate of 0.03, momentum of 0.9 (where applicable) and a weight decay of 0.0001. For further decrease in the loss, the model could be trained for a longer epoch range. Since, the optimal loss is unknown, it is hard to say which model converged to the optimal solution, and would have been the optimal optimizer for this dataset.

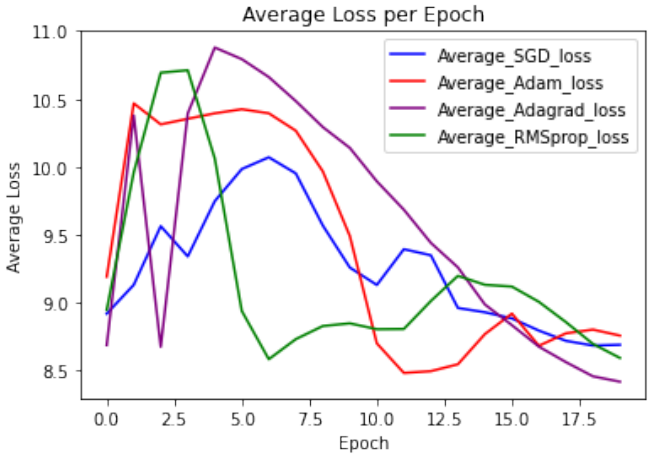

Figure 2. Average Loss per Epoch

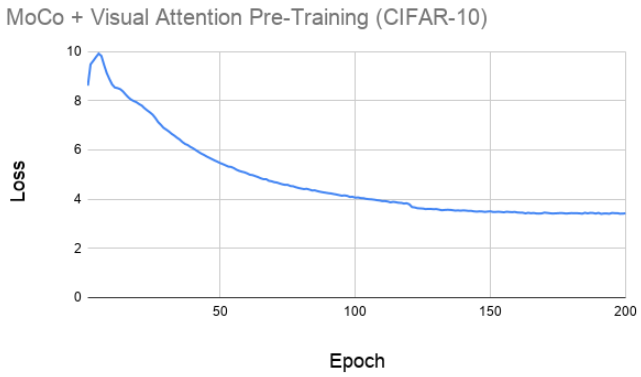

Figure 3. MoCo-VA pre-training loss curve on CIFAR-10

Compared to the original MoCo model which was pre-trained with ImageNet, adding visual attention (VA)and pre-training on CIFAR-10 showed a slight drop in performance; the MoCo-VA model pre-trained and fine-tuned on CIFAR-10 ultimately only reached 66.9% top-1 accuracy, compared to the 69.3% we found without visual attention. While the accuracy is lower, comparing ImageNet and CIFAR-10 pre-training is not a perfect comparison. Optimally, we would have pre-trained using ImageNet, but the size of the dataset was not feasible for the scope of this project.

Another source of error may be implementation mistakes. For one, we had to implement visual attention from scratch, and while the software runs without issue there is always the possibility that we overlooked an implementation detail that is throwing off the results. Additionally, the repository we were working with is designed specifically for training on multi-GPU distributed systems with ImageNet data, neither of which were feasible for our project. We tried to the best of our ability to adapt the script for our environment, but the complexity of the system that FAIR

built could be concealing some hard-coded component that we overlooked.

Lastly, it may just be that visual attention is not applicable to this domain. While we had good reason to choose visual attention as a possible improvement, as it has seen success in adjacent domains, our hypothesis may simply be the victim of Occam's Razor and over-complicating the MoCo model.

4. Future Work

MoCo has significant potential to become a viable model for mastering unsupervised learning. There are many hyperparameters available, far more than we actually used in our project. It has significant customizability, and as explained in the MoCo paper it transfers well to different recognition and/or segmentation tasks [6]. Given significant time and computing resources, hyperparameter tuning could result in improved results on multiple detection and segmentation tasks. A particular aspect of our project that could show improvement to the current industry practice for unsupervised learning is the application of attention. As was shown with BERT in the domain of natural language processing, attention is a powerful tool that, when applied perfectly, dramatically improves the accuracy of a network [4]. Of course, NLP and unsupervised image recognition are vastly different fields, but if the application of attention can be further honed, we may see a new rise in unsupervised image recognition models.

5. Work Division

This is a section on the delegation of work among team members at the end of the report, in the form of a table. Refer to Table 1.

Student Name	Contributed Aspects	Details
Sahil Dhingra	Implementation and Analysis	Trained/tested the pre-trained models on new datasets.
Raymond Ortiz	Implementation and Analysis	Implemented visual attention and analyzed the results.
Tanvi Bhagwat	Implementation and Analysis	Trained the model with different optimizers and analyzed the results.
Karan Khosla	Implementation and Data cleaning	Created scripts for sorting data and re-arranging filesystem.
		Implemented base MoCo notebook for running untrained model on CIFAR-10.
The Whole Team	Writing the Report	Every team member made significant contributions to the report

Table 1: Contributions of team members.

References

[1] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. *arXiv:1712.02616*, 2018. 1

[2] Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single layer networks in unsupervised feature learning. 2011. 2

[3] Alexandre Defossez, Leon Bottou, Francis Bach, and Nicolas Usunier. On the convergence of adam and adagrad. *arXiv:2003.02395*, 2020. 2, 3

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2019. 5

[5] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. 2006. 2

[6] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv: 1911.05722*, 2020. 1, 2, 5

[7] Kaiming He and Yuxin Wu. Moco: Momentum contrast for unsupervised visual representation learning, Mar. 2020. 2, 3

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, page 770. 1

[9] Saumya Jetley, Nicholas A. Lord, Namhoon Lee, and Philip H. S. Torr. Learn to pay attention. *CoRR*, abs/1804.02391, 2018. 3

[10] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 1

[11] Tom Le Paine, Pooya Khorrami, Wei Han, and Thomas S. Huang. An analysis of unsupervised pre-training in light of recent advances. *arXiv: 1412.6597v4*, 2015. 1

[12] Tal Ridnik, Hussam Lawen, Asaf Noy, and Itamar Friedman. Tresnet: High performance gpu-dedicated architecture. *arXiv: 2003.13630*, 2020. 1

[13] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. 2018. 2

[14] Richard Zhang. Making convolutional networks shift-invariant again. *arXiv: 1904.11486*, 2019. 1