Data Efficiency of Self-Supervised Learning with Momentum Contrast

Author: Makar Kuleshov Supervisors: Alex Manolache, Petter Reijalt Responsible professor: Jan van Gemert

Delft University of Technology

Introduction

Currently, developing vision foundation models is mostly available to a few, often commercial, organizations that have resources to work with massive datasets. This creates a fragile dependency, and the lack of transparency about the training data raises concerns about bias, copyright, and privacy.

Momentum Contrast (MoCo) is one of the Self-Supervised Learning methods that can be used to train foundation models. Its latest version, MoCo v3, achieves some of the best results among SSL methods on ImageNet, but its performance has not been studied when only a small dataset is available. We close this gap and study the data-efficiency of MoCo: we train it on subsets of Tiny-ImageNet of varying sizes and evaluate the resulting models on the Visual Task Adaptation Benchmark (VTAB).

Research Questions

1. How should MoCo's training parameters be chosen to get the best accuracy for different sizes of pre-training data?
2. How does the downstream accuracy of MoCo scale with the amount of pre-training data?
3. How does this scaling vary across different types of downstream tasks?

Methodology

We use an adapted implementation of the Momentum Contrast from the MoCo v3 paper, the architecture is illustrated in Figure 1. MoCo learns from two augmented views of an image: their embeddings are pulled together, while the embeddings of other images (negative keys) are pushed away.

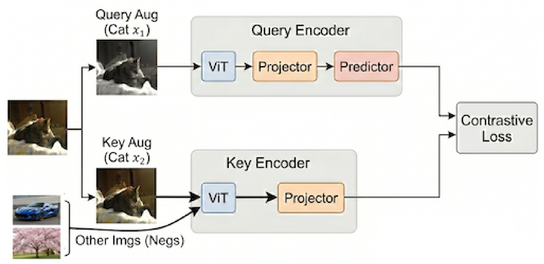

Figure 1. MoCo v3 architecture.

In our work we analyze two parameters of MoCo that most strongly affect its downstream accuracy:

- **Number of negative keys:** determines how many other images each image is contrasted against in the loss. We use in-batch negatives, so it is set by the batch size.
- **Momentum coefficient:** the key encoder is an exponential moving average (EMA) of the query encoder, and this coefficient defines the update function, influencing how fast the learned representations are allowed to change during training.

After finding the best values for these parameters we continue to the downstream accuracy evaluation using them.

Experimental setup

- **Pre-training data:** nested, class-balanced splits of **Tiny-ImageNet** (200 classes, 64×64), with sizes following powers of two: $1\text{k}, 2\text{k}, 4\text{k}, 8\text{k}, \ldots, 64\text{k}$, plus the full 100k images.
- **Downstream data: VTAB**, a benchmark of diverse visual tasks covering three categories: 7 natural, 4 specialized, 8 structured.
- **Backbone:** ViT-Tiny/8 (patch size 8 suits the small images)
- **Training duration:** dynamic with early stopping. kNN accuracy is monitored on the Tiny-ImageNet validation set every 5 epochs, and training stops once the best value has not improved for the last 15% of epochs. The best checkpoint is kept (illustrated in Figure 2).
- **Evaluation:** linear probing on the frozen backbone: on the Tiny-ImageNet validation set to select MoCo's parameters, and on VTAB for downstream accuracy.

Results

Optimal MoCo parameters

Because of limited computational resources we tested the parameters only on every other split.

The optimal **number of negatives** grows with the amount of pre-training data. It can possibly be explained this way: on small splits, smaller batches give noisier gradients that act as a regularizer and reduce overfitting, so larger batches only pay off once more data is available.

The best **momentum coefficient** shows no clear pattern.

split size	1k	4k	16k	64k
number of negatives	125	125	500	1000
momentum coefficient	0.99	0.999	0.99	0.9

Table 1. Optimal number of negatives and momentum coefficient for each pre-training split.

Training duration

Figure 2. Example training history: kNN accuracy tracked during pre-training on the split of 64000 images.

Under the kNN early-stopping protocol, the number of epochs needed to converge decreases as the dataset grows: small splits require many more epochs per run, potentially because each epoch contains fewer gradient updates.

split size	1k	2k	4k	8k	16k	32k	64k	100k
number of epochs	4865	3645	3280	1440	1405	840	395	380

Table 2. Number of pre-training epochs required to reach the early-stopping criterion for each split size.

Downstream accuracy on VTAB

Pre-training helps even with very little data: the 1k split already reaches 33.3% versus 21.4% for a randomly initialized backbone. Accuracy then grows approximately log-linearly, by about 1.33% per doubling of the data.

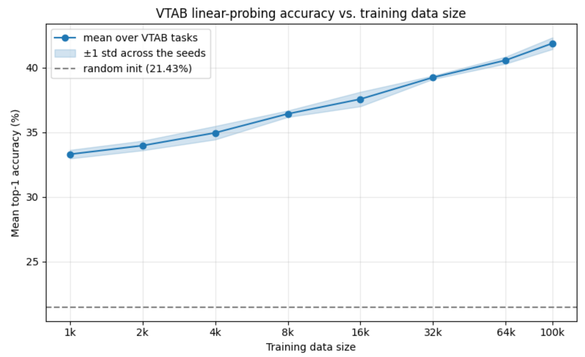

Figure 3. Data-efficiency measured over all tasks

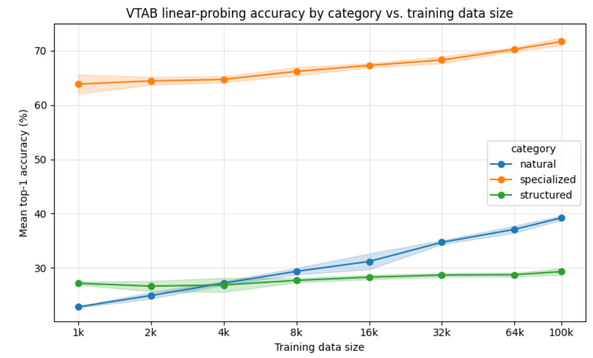

Figure 4. Data-efficiency measured in each category of the tasks

By task category, the gains are uneven: **natural** tasks improve the most (Tiny-ImageNet contains similar images), **specialized** tasks improve moderately, while **structured** tasks stay almost flat.

Conclusions

Choosing Parameters

- In the original MoCo papers, increasing the number of negative keys was always beneficial; in the low-data regime we observe the opposite: smaller batches, with fewer negatives, can give the best accuracy.
- The optimal momentum coefficient also sometimes differs from the 0.99 recommended in the original paper, and shows no clear pattern across the splits.
- Our models needed many more epochs to converge than the 300 that were enough in the original work, especially on the small splits.

These differences show that the MoCo parameters should be adapted when the amount of available data changes.

Data-Efficiency of Downstream Accuracy

- Downstream accuracy grows approximately log-linearly with the dataset size, both for the mean over all tasks and within each category individually. This result extends general scaling observations made previously in the field.
- The scaling rate differs across task categories, which seems to depend on how close each category is to the pre-training data. **Natural** tasks are the closest to Tiny-ImageNet and improve the most; **specialized** tasks are a bit further and improve moderately; **structured** tasks (e.g. object counting, distance estimation) are too challenging, features learned from general images, at only 64×64 resolution, help them little, so they stay almost flat.

Future Work

It is possible to continue this research in several directions:

- Extending the analysis of MoCo. For example, by evaluating a different backbone, such as a convolutional network like a ResNet, to test whether the same data-efficiency behavior holds beyond transformers.
- Comparing MoCo to other contrastive learning methods under the same limited-data conditions, to see whether our conclusions apply more broadly.