Self-Supervised Learning to Create Image Embeddings for Classification Tasks

Aaron Jimenez, Kali Hale
University of California, Santa Barbara
aaron_jimenez@ucsb.edu, kalihale@ucsb.edu
March 18, 2023

Abstract

In image classification problems which require large datasets, finding labelled datasets can be an issue. The vast majority of the data that exists is unlabelled, making it impossible to use with supervised learning. In cases like this, we turn to self-supervised learning. In this project, we will test the efficacy of a self-supervised ResNet-50 model pretrained using Sim-CLR compared to various other supervised PyTorch models on the Oxford-IIIT Pet Dataset.

1. Introduction

One of the main issues with image classification problems has been the need for a large labeled dataset. Given the massive size of some datasets, such as ImageNet [3], the costs for labeling all of the examples for said dataset can be incredibly expensive in both time and resources (human effort, financially, etc.). As such, a potential solution to these problems are to pretrain an image classification model in an unsupervised manner that would not require an sort of labeling during the training.

Various unsupervised learning methods have been proposed over the years in order to achieve models with performance results close to those of their supervised counterparts.

1.1. Generative Approach

Some approaches have involved using using unsupervised training to generate an image based on an transformed version of said image. From there an intermediate representations can be extracted which can be used for classification tasks.

While this this approach can be effective in generating an image embedding, it is somewhat computationally costly. In contrast to this approach, other proposals involve using discriminative models which are less costly to use.

1.2. Discriminative Approach

In general the training strategy for unsupervised discriminative models is similar to that of supervised models. The only main difference is that the labels are generated using heuristics that are derived from an unlabeled datasets. However, using a heuristic-based approach could negatively affect the generality of the learning feature embeddings generated by the model.

Another approach for training unsupervised models involves a contrastive approach to training.

1.3. Contrastive Learning

As seen in previous works [5, 6], a discriminative approach to training based on contrastive learning can actually be very effective at learning general feature representations for an image, comparable to supervised models. However, these approaches can be complicated, requiring things such as specialized architectures.

As such simpler contrastive learning framework was proposed called *SimCLR* [2]. This framework does not make use of any specialized architecture in order learn and generate image feature representations while generating state-of-the-art results.

1.4. Uses and Project Plan

For this paper, we hope to use a pretrained *SimCLR* model fine-tuned on the Oxford-IIIT Pet Dataset [10]. We hope to then compare this with other models fine-tuned on the same dataset. Our goal is to demonstrate that models pretrained using *SimCLR* can still be give good performance when trained on downstream tasks.

2. *SimCLR* Architectural Design

In order to train a contrastive model, the *SimCLR* framework breaks the process into four parts: stochastic data augmentation, a neural base encoder, a neural projection head, and a contrastive loss function. A visual representation of the architecture and the general algorithm can be seen in Figures 1 and 2.

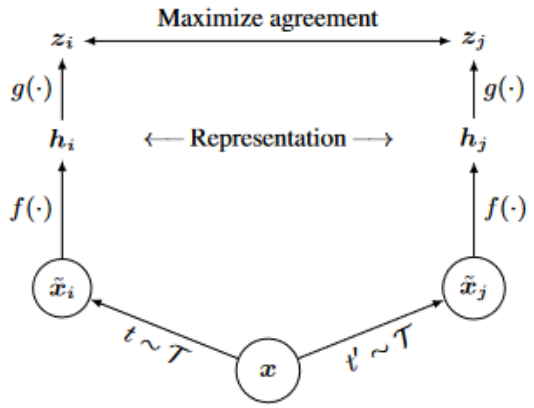

Figure 1. Two different data augmentations from the same family (t and t') are applied to the same input x to get $\tilde{x_i}$ and $\tilde{x_j}$. From here they are both passed through the base encoder $f(\cdot)$ to get h_i and h_j. These are then passed through the projection head $g(\cdot)$ to get x_i and x_j. From here, the contrastive loss is used to train $f(\cdot)$ and $g(\cdot)$ to maximize the agreement of x_i and x_j. [2]

2.1. Data Augmentation

The stochastic data augmentation used by the *SimCLR* framework is composed of a series of transformations on an image on order to to augment the image dataset. In the paper, the authors use a combination of three different types of augmentations: random image cropping followed by resizing the image back to its original size, random color distortions, and random Gaussian blur.

This allows the authors to generate far for training examples than what is actually provided by the original dataset. This is needed considering that contrastive learning strategies generally tend to require larger amounts of training data than supervised learning.

2.2. Neural Base Encoder

The neural base encoder, $f(\cdot)$, is used to extract feature information from the augmented dataset. For the *SimCLR* the authors used various different ResNet [7] models for various different tests. We chose to use a pretrained ResNet-50 model provided by the authors.

2.3. Neural Projection Head

The neural projection head, $g(\cdot)$, is used by the framework to map the output of $f(\cdot)$ to a space that can be used by the contrastive loss function. In the paper, the authors use one MLP hidden layer with a ReLU activation function.

2.4. Contrastive Loss

In the paper, they then use a contrastive loss function for a contrastive prediction task.

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of $f, g, \mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

Figure 2. Summary of the *SimCLR* algorithm. [2]

As an example, given set $\{\tilde{x_k}\}$ that includes positive pairs $\tilde{x_i}$ and $\tilde{x_j}$, the contrastive prediction task tries to find the $\tilde{x_j}$ in $\{\tilde{x_k}\}_{k \neq i}$ for a given $\tilde{x_i}$.

The contrastive loss function is given by the equation:

$$\ell_{i,j} = \frac{\exp\left(\text{sim}(z_i, x_j)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{\{k \neq i\}} \exp\left(\text{sim}(z_i, x_k)/\tau\right)}, \qquad (1)$$

where $\mathbb{1}_{\{k \neq i\}}$ is 1 iff $k \neq i$ or else 0, τ is a temperature parameter, $\text{sim}(u, v) = u^\top v / ||u|| ||v||$, and N is the batch size. The final loss is calculated across all positive pairs in the batch.

3. Experimental Framework

For our experimental framework we chose to compare a ResNet-50 PyTorch model pretrained using *SimCLR* to various pretrained supervised PyTorch models, both CNN-based and Transformer-based, for image classification tasks.

We ran our tests on two different machines. The specifications of which are:

- Computer 1
  - CPU: AMD Ryzen 5 5600X
  - RAM: 32 GB DDR4

  - GPU: Nvidia RTX 3060 (12 GB vRAM)

- Computer 2
  - CPU: Intel Core i9-9900K
  - RAM: 32 GB DDR4
  - GPU: Nvidia RTX 2070 (8 GB vRAM)

The link to our experimental framework can be found here: https://github.com/MadKingAaron/CS281_Project.

3.1. Training

For our downstream stream training we chose to add an addition projection head with a ReLU activation and train our models using these. These models were trained for an image classification task on the Oxford-IIIT Pet Dataset [10] using the Adam [8] optimizer and PyTorch's *ReduceLROnPlateau* learning rate scheduler.

We conducted various different training sessions were we changing things the training batch sizes, learning rate, and training epochs for all the models. In addition we both trained the models using both fine-tuning and transfer learning, where all layers except for the final added projected head's weights remained frozen.

3.2. Testing and Validation

For testing and validation we decided to use Scikit-Learn to give us a classification report based on the validation and testing dataset. In addition, when validating the model we return the validation loss which is used to adjust automatically the learning rate after each training epoch if deemed necessary.

4. Experimental Results

4.1. Comparing Different Learning Rates

Using a batch size of 64 and comparing on 50 epochs, we can see in Figure 3 that there is a small positive difference to decreasing the learning rate by a factor of 10 for most of the algorithms with the exception of VGG [9] and VIT [4]. VGG and VIT perform very badly with a learning rate of 0.001, but perform similarly to the others with a learning rate of 0.0001. Because of this, we chose to fix our learning rate at 0.0001 for the rest of our comparisons.

We tested additional learning rates on SimCLR (with 20 epochs and batch size 32) to see if we could improve our average accuracy, as shown in the table below. Even at the best learning rate of 0.00035, we only achieved an accuracy rate of 0.53.

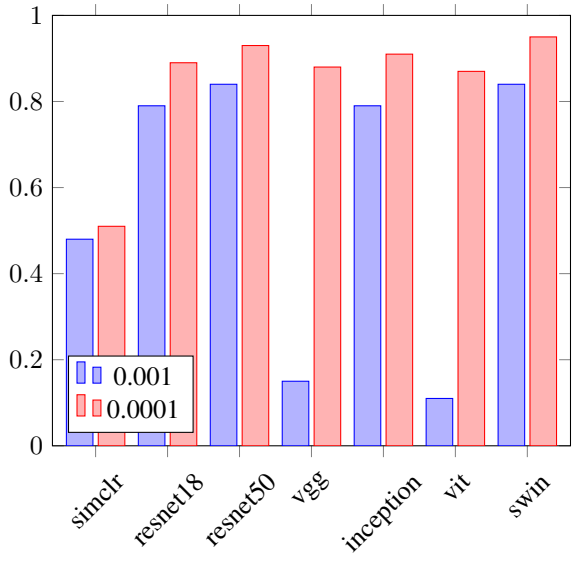

Figure 3. Learning rate difference on finetuned models using 50 epochs and a batch size of 64.

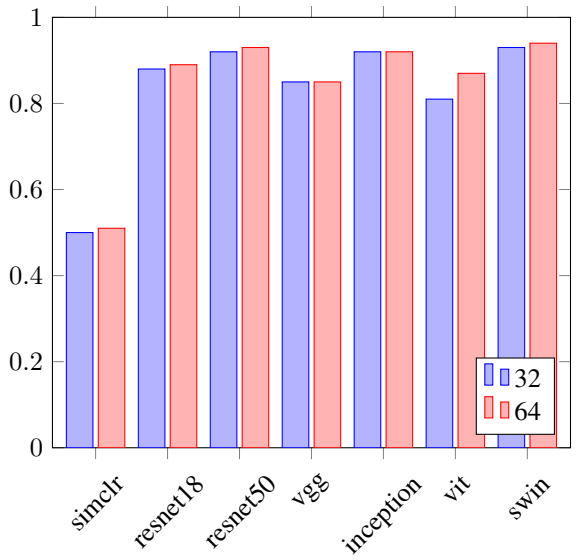

Figure 4. Comparing fine-tuned models using batch size 32 vs 64 using 50 epochs and a learning rate of 0.0001.

learning rate	accuracy
0.001	0.44
0.0008	0.43
0.0005	0.5
0.0004	0.51
0.00035	0.53
0.0003	0.51
0.0001	0.5

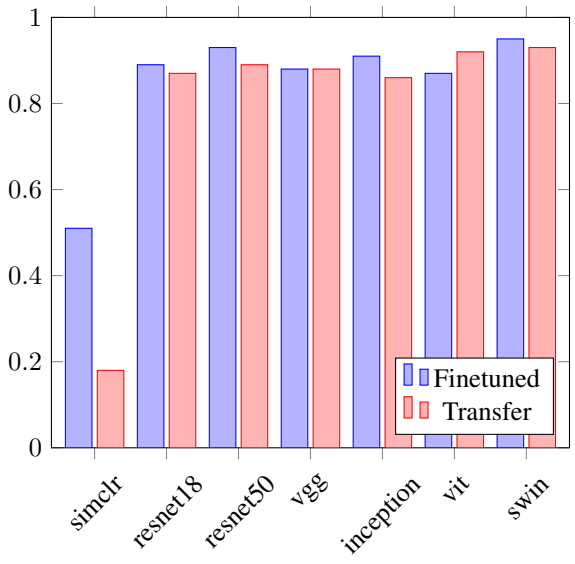

Figure 5. Finetuned models vs transfer learning training on 50 epochs with a learning rate of 0.0001 and a batch size of 64

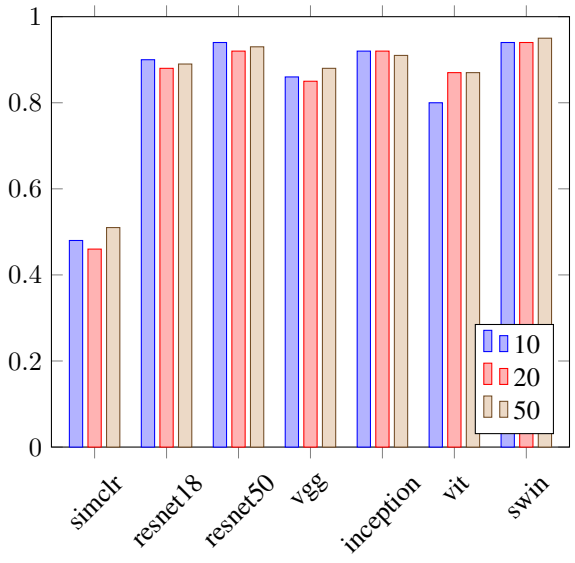

Figure 6. Epoch difference on finetuned models using a learning rate of 0.0001 and a batch size of 64

4.2. Comparing Batch Sizes

As we can see in Figure 4, the difference between training our model in batch size 32 versus batch size 64 is negligible. Even using batch size 256, *SimCLR* only achieved an accuracy of 46%.

This is important considering that according to the authors of the *SimCLR* paper described the model needing to take advantage of larger batch sizes during training. However, given this was explained in the context of contrastive learning and not supervised learning, as we had done during finetuning. Given this, we are not sure exactly what could have been the issue. Our current working theory is that it had to do with the pretrained *SimCLR* that was provided on the authors' GitHub repository.

4.3. Comparing Finetuned vs Transfer Learning

In the next test we compared finetuned models where all model parameters are trained and model trained using transfer learning where all layers except for the last projection layer are frozen during training. Model was trained for 50 epochs with a learning rate of 0.0001.

What we found was that transfer learning actually performed much worse for *SimCLR* and slightly worse for all other models. Even when using transfer learning it is clear that the *SimCLR* model performed far worse than all the models pretrained with supervised learning.

These results can be found in Figure 5.

4.4. Comparing Training Epochs

For our following test we training using 10, 20, and 50 epochs for each model. As can be seen in Figure 6, where *SimCLR* still performs worse than all of the other models. For every other model architecture, the model does not perform much better or worse when the number of epochs is increased, and every other model tested performs significantly better than *SimCLR*

5. Future Work

While the *SimCLR* pretrained model did not give us the results we had expected, other studies such as Bountos et. al. [1] were able to achieve higher accuracy on a large dataset. However, they trained a model from scratch using *SimCLR* in contrast to our approach of using a model already pretrained using the framework. Perhaps in future work, by training a model from scratch, we can achieve better results.

6. Conclusion

In conclusion, recent works into unsupervised learning for image classification has shown results that are comparable to the more traditional supervised models. This is important given the large amount of resources needed to create a labeled dataset that can be used to adequately train a supervised model. As such we decided to train an off-the-shelf model trained using the *SimCLR* framework on a downstream task and compare it to other supervised off-the-shelf models. And while our findings did not show great results, we believe that in the future by tweaking our testing and training a *SimCLR* model from scratch, we can perhaps achieve results comparable to the supervised models.

7. Acknowledgements

Special thanks to Bryce Husserl, who generously ran our code with batch size 256 on his 64 GB of RAM.

References

[1] Nikolaos Ioannis Bountos, Ioannis Papoutsis, Dimitrios Michail, and Nantheera Anantrasirichai. Self-supervised contrastive learning for volcanic unrest detection. *IEEE geoscience and remote sensing letters*, 19:1–5, 2022. 4

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020. 1, 2

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. pages 248–255, 2009. 1

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. 3

[5] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. 1

[6] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. 2:1735–1742, 2006. 1

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 2

[8] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014. 3

[9] Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. pages 730–734, 2015. 3

[10] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 1, 3