A Somewhat Less Simple SimCLR

COLE FRANK

Overview

Chen et al.'s "A Simple Framework for Contrastive Learning of Visual Representations" provides a straightforward method for self-supervised learning of image embeddings and demonstrates the effectiveness of these representations on ImageNet and CIFAR classification. Their method entails constructing two different augmentations of every image in a given batch, passing all of the images through their encoder network and another final feed-forward + non-linear activation layer and maximizing the similarity between the two different augmentations that come from the same image. They then use the thus trained encoder network (without the final feed-forward + non-linear activation layer used for the self-supervised contrastive loss task) to embed images that they then train a linear classifier on for a supervised classification task.

I tested two different extensions of this SimCLR framework:

1) A greyscale augmentation of the input images
2) A modification of the batching/training process and the loss function such that within each batch each image was not duplicated but either triplicated or quadrupled

Methods

Greyscale Image Augmentation

Chen et al. found that composing multiple image augmentations together was more effective for learning image representation than any single augmentation of an image. In their final implementation on the CIFAR images they compose three augmentations for each image: a random resized crop, a random horizontal flip, and a color distortion. Another paper, Cai et al 2020, analyzed the contribution of negative examples in the self-supervised task by the difficulty of the example (where difficulty is measured by how similar the negative embedding is to the target image embedding) and found that only the 5% most difficult negative examples are necessary for most of the learning to occur.

These results seem to suggest that augmentations that make the contrastive loss task more difficult (up to a point) result in better visual representations. So I hypothesized that an image transformation that further obscured information in the original image might yield even better representations. To test this hypothesis, I convert all of the training images to grayscale in addition to the other augmentations in the SimCLR pipeline.

Triplicates, Quadruples

The SimCLR method involves duplicating and then augmenting every image in a given batch, then summing the contrastive loss term for each positive pair (i, j), which takes the form:

$$\ell_{i,j} = -\log \frac{\exp\left(sim\left(z_i, z_j\right)/\tau\right)}{\sum_{k=1}^{2N} 1_{[k\neq i]} \exp\left(sim\left(z_i, z_k\right)/\tau\right)}$$

I modified this method to instead triplicate or quadruple the images in a given batch before augmenting all of them individually and performing the contrastive loss task. My new loss function (in the triplicate case) for each positive triplet (i, j, k) was:

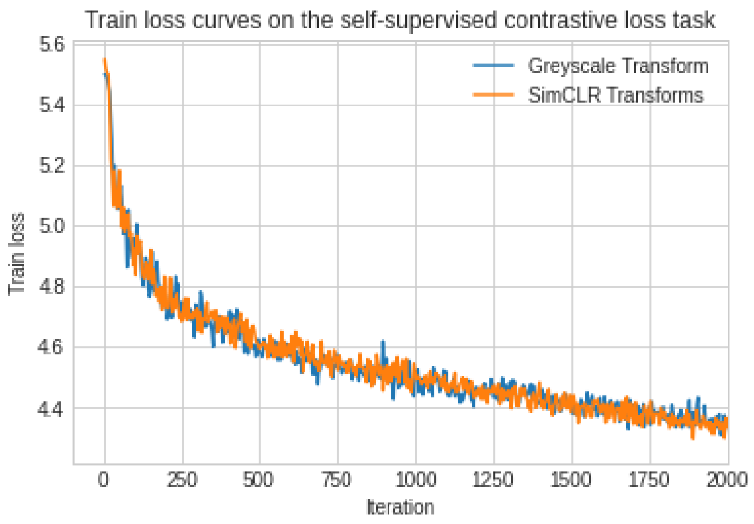
$$\ell_{i,j,k} = -\log \frac{\exp\left(\left(sim\left(z_i, z_j\right) + sim\left(z_i, z_k\right) / \tau\right)\right)}{\sum_{h=1}^{2N} 1_{[h \neq i]} \exp\left(sim\left(z_i, z_h\right) / \tau\right)}$$

Results

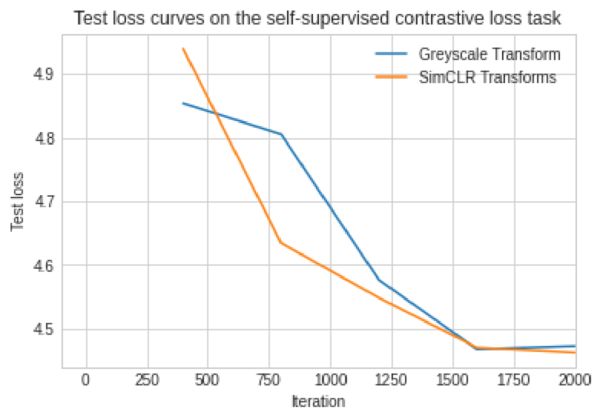
Due to compute constraints I was limited in what experiments I could perform. The original SimCLR paper found that their performance scaled with batch size because the batch size determined the quality and difficulty of the contrastive loss task (a greater batch size translates to more negative examples). They experimented with batch sizes ranging from 256 to 8192 images and found that batch sizes of 2048 consistently outperformed smaller batch sizes. However, the Google Colab GPUs I used to run my experiments did not have enough RAM to handle anything larger than a batch size of 256 when using the original SimCLR implementation. Furthermore, since my triplicate and quadruple extensions effectively entail tripling and quadrupling the batch size, the largest batch size I could run for these modifications was 128. So, for consistency I used batch sizes of 128 for all of the results that follow.

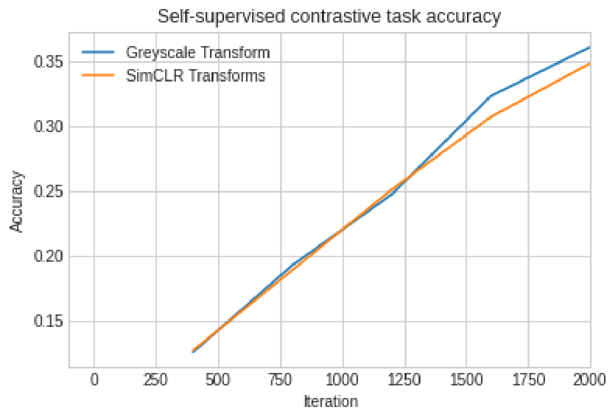
Greyscale Image Augmentation

The greyscale augmented images performed very similarly to the original SimCLR implementation on the self-supervised contrastive loss task:

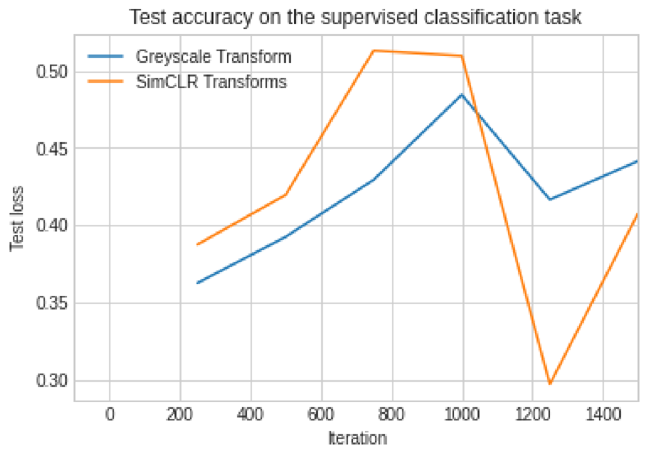


Nearly matching the train loss curve exactly and even slightly outperforming the original implementation in terms of test loss and accuracy on the contrastive task:



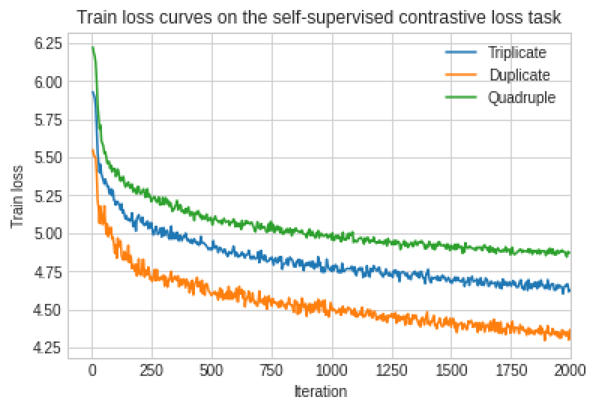


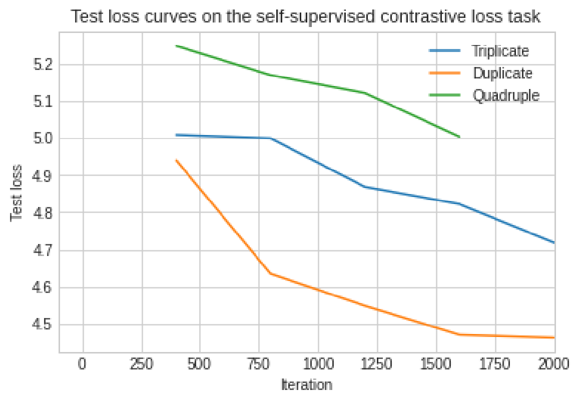
On the linear classification task, the greyscale augmented image modestly outperforms the original implementation, with an accuracy of 0.44 on the test data after 30 epochs, versus an accuracy of 0.41 for the original implementation:



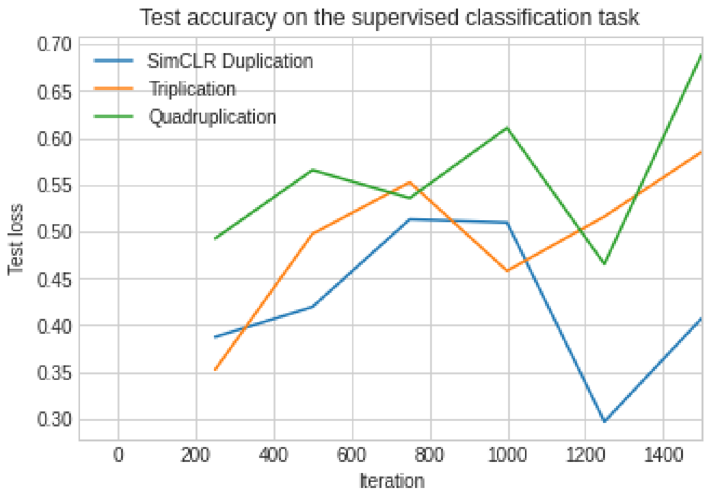
Triplicates, Quadruples

The train and test loss curves for the triplicate and quadruple extensions are visibly worse than for the original duplicate implementation:





On the linear classification task, the triplicate and quadruple methods significantly outperform the original SimCLR duplication:



After 30 epochs of training data on the classification task, the quadruple method achieves 69% accuracy on the classification task, whereas the triple method achieves 58% and the original SimCLR method only 40% accuracy.

Discussion and Conclusion

Both these extensions seem to create better visual representations insofar as they yield better results on the linear classification task. That said, my hypothesis for the greyscale augmentation that reducing the amount of information in the image would make the contrastive task harder was not clearly borne out by the results as the train and test curves of the grayscale and non-greyscale data are very similar. Thus, it's intriguing that the decoder learned on the greyscale augmented data still somewhat outperformed on the classification task. It's worth noting that the grayscale transformation significantly reduces the information and size of the images (a threefold decrease since the 3 color channels are collapsed to 1), so training on greyscale images could be more efficient than training on color images.

Generally, I think a few caveats are in order. I did not have the compute and time to train these models to convergence. It is possible that training to convergence with larger batch sizes would yield different results. Furthermore, the duplicate/triple/quadruple results are arguably not an apples-to-apples comparison. While each approach uses 128 unique images per batch, tripling or quadrupling means more augmentations per batch. Specifically, at a batch size of 128, the duplicate method sees 256 total augmentations, the triplicate 384 and the quadruple method 512. Not only are the triplicate and quadruple methods seeing more augmentations per batch, they are also more computationally intensive as a result. As a next step I would like to control for this discrepancy by scaling the unique image batch size for each method appropriately.

References

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations." In International conference on machine learning, pp. 1597-1607. PMLR, 2020.

Cai, Tiffany Tianhui, Jonathan Frankle, David J. Schwab, and Ari S. Morcos. "Are all negatives created equal in contrastive instance discrimination?." arXiv preprint arXiv:2010.06682 (2020).