Exploring GAN Image Data Augmentation in Visual Self-Supervised Learning

Roberto Halpin-Gregorio

rgh224@cornell.edu

1 Introduction

Self-supervised visual representation learning algorithms aim to learning useful visual representations without human supervision. The success of many machine learning algorithms relies on the representation of the input data, thus learning informative representations of images without human knowledge is quite useful. Recent progress in this field have greatly improved the quality of these visual representations, however it is clear from the results of these methods, that there is room for improvement. Additionally, there is the question of how these algorithms are affected by the quantity and quality of their input data.

The self-supervised representation learning model we are interested in examining is a contrastive learning approach called SimCLR [2]. This is due to its simplicity and popularity in the field. SimCLR is a contrastive learning algorithm, which are known to require a substantial amount of data. This is due to their self-supervised nature; these algorithms are trained without labels. The most common method to provide the large amount of data is through data augmentation. The data augmentation used in these algorithms modify the initial dataset with functional operations to produce more input data. The functional operations that SimCLR uses is random crop and resize (with random flip), color distortions, and Gaussian blur. Using multiple different compositions of these data augmentations, with different hyperparameter settings, they can create a fairly large dataset out of a small or moderate dataset. Data augmentation in this manner is not new in machine learning; it has been heavily used in many supervised learning settings. However, SimCLR argues and experimentally shows that data augmentation is considerably more important in contrastive self-supervised learners than in supervised learners.

This need for a lot of data and the reliance of functional data augmentation raises an interesting question: can we use another form of data augmentation to improve our SimCLR model? Additionally, even if an alternative form of data augmentation does not strictly improve the representations of our self-supervised learner, there is bound to be some insight to gain from observing the results.

To be more specific, this project will investigate a new form of data augmentation for self-supervised visual representation learning algorithms. The idea is to use data generated from a deep generative model, specifically a Generative Adversarial Network (GAN) [5], to augment our input dataset. Thus, we aim to improve the representations generated by these self-supervised algorithms by focusing on the input data utilized by these visual representation learning algorithms. We argue that this is of great interest because if GAN samples improve the representations of SimCLR then it could completely transform the landscape of self-supervised learning due to a strong additional data augmentation method. The focus will be in the low-data regime as this will benefit the most from data augmentation. Additionally, due to the computational resources and time constraints, it was important to limit the image dataset size.

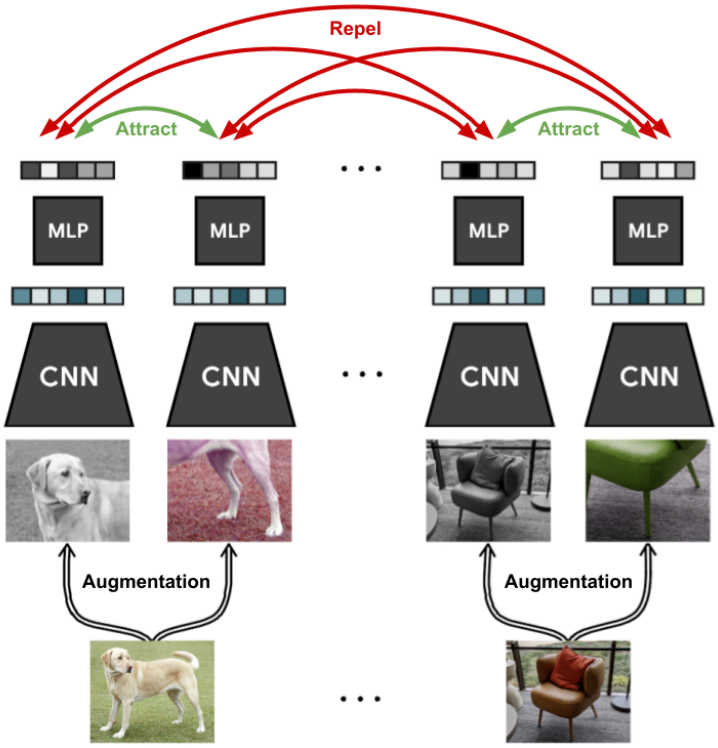


Figure 1: Overview of SimCLR (from SimCLR webpage).

Exploring the data efficiency of self-supervised learning is essential as these methods typically require many passes through the data (epochs) to achieve desirable performance. When compared to supervised learning, self-supervised methods typically take 3-5x the epochs to converge. For example, SimCLR achieves best results when trained for at least 1000 epochs. This project aims to explore how generative model data augmentation affects the performance of the SimCLR model when dealing with a small amount of data. Additionally, it is of interest to observe the learning efficiency when using this strategy. How does this data augmentation affect the amount iterations or mini-batches it takes to reach a reasonable accuracy?

2 Background

The main algorithm we are aiming to improve is SimCLR [2], which is a contrastive self-supervised visual representation learning algorithm. A self-supervised visual representation learning algorithm aims to learn the best possible representations of visual data, such as images, without labels. Being self-supervised means that the model itself assigns its own form of labels to its inputs so that it can learn; it is a form of unsupervised learning. This method is how humans learn the majority of the time; we evaluate events without being explicitly told a *label*.

The SimCLR framework applies random transformations to its input image, resulting in a pair of two augmented images x_1, x_2 (different from the dataset augmentation). Then each of these images are passed through an encoder to output representations z_1, z_2. The model is trained by identifying the correct pair of images out of a collection of z_i, and maximizing the similarity of these two representations z_1, z_2 (contrastive loss). Contrasting the correct representations out of the collection is why the model is a contrastive learner. The goal is to create an embedding space that maps similar images close together and dissimilar images far apart. Refer to Figure 1 for an illustration of the pipeline.

There are two main steps when it comes to SimCLR for our purposes: pre-training and linear evaluation. Pre-training is where we train the SimCLR model in a self-supervised approach using the

1. Given an image dataset X without labels.
2. Train a generative model on X to generate a set of fake images Z.
3. Use $X \cup Z$ to pretrain a self-supervised representation learner, such as SimCLR.

Figure 2: Overview of the main method.

contrastive loss. This creates a trained encoder that will output our representations. Then to evaluate the representations we perform the linear evaluation protocol. This involves attaching a linear model head to the output of our encoder, we then fine-tune this linear model on our same dataset that we pre-trained on. Fine-tune here means that we freeze the weights in our encoder and just update our linear model weights. This acts as a linear classifier applied to our representations; it is a simple method to determine the quality of the representations. Finally, we evaluate the encoder with the linear model on our testing data to receive our test accuracy. This test accuracy is the main metric used to evaluate the representations in all recent self-supervised representation learning algorithms.

The generative model of choice to augment our input dataset is the Generative Adversarial Network (GAN) [5]. The GAN is a generative model framework that has two main models, the discriminator and generator. The discriminator aims to correctly distinguish between real images and fake images produced by the generator, whereas the generator wants to trick the discriminator. When both models are finished training, the generator can act as a great fake image generator, as is has been trained to produce images similar to the real image distribution used in training.

3 Method

Our goal is to improve the representations produced by self-supervised representation learning algorithms by targeting the input data. The main idea behind this project, outlined in Figure 2, is to use GANs to generate new images to be used as an augment dataset for self-supervised representation learning algorithms. More specifically, given an image dataset X, a GAN will be trained on X to generate a set of images Z. Then a self-supervised representation learning algorithm will be pre-trained on $X \cup Z$, and we will compare its representations on the augmented data to its representations on just X.

Thus, the main idea is to generate fake images using our real training data and use these fake images as data augmentation. Hopefully, the self-supervised models that we pre-train with this data augmentation can beat the performance on the original, real dataset. One key requirement is that the GAN, or generative model, used can produce good samples with limited training data.

As mentioned previously, the self-supervised representation learning model we choose for this project is a contrastive learning approach called SimCLR [2]. The generative model we will use to augment our real training data is the GAN model. This is because GANs tend to have high quality generated outputs, compared to other generative approaches, and recent work on improving their data efficiency [14]. We specifically are using the StyleGAN2 model [8] under the DiffAugment framework [14]. We are using the DiffAugment framework due to their open-sourced code, and the data efficient GAN training framework. GAN are notorious for requiring large amounts of data to achieve good results, especially realistic image generation. However, this recent data efficient GAN training framework can use tailor-made data augmentation for the GAN framework to allow GANs to train well in the low-data regime. Given that the self-supervised pre-training framework is based on the idea that we have no labeled data, we require an unconditional GAN. An unconditional GAN is a GAN that specifically trains with unlabeled data, opposed to

conditional GANs which use labels in the training process. The unconditional GAN of choice is the SyleGAN2 for two main reasons. The StyleGAN2 was chosen because DiffAugment has wrapper code for this model, and the model achieves great distribution quality metrics and perceived image quality. The specifics of the StyleGAN2 are not too important for our methods, since we primarily concerned with just a generative model that produces good quality images to test our idea.

Intuitively the area that would benefit the most from this form of data augmentation would be a low-data environment. If there is a lack of real data, a supply of fake, but realistic, images may benefit learning better representations in a self-supervised network. Therefore, the amount of real data available will be constrained to explore this hypothesis. Critically, it will also depend on the capabilities of the data-efficient GAN, since we can only train the GAN with the limited amount of real data available.

4 Implementation

As mentioned previously, the DiffAugment framework code is used with the StyleGAN2 model. Default StyleGAN2 CIFAR-10 hyper-parameters were used for training. Scripts were made to generate the fake dataset using the trained Generator. The SimCLR codebase used for this project is an unofficial implementation on Github. The reason this codebase was chosen was because it is written in PyTorch [12] whereas the official code is written in Tensorflow [1], where is much less preferable. A custom dataset and data loader were implemented for this project as well as some slight changes to the SimCLR model and optimizer code.

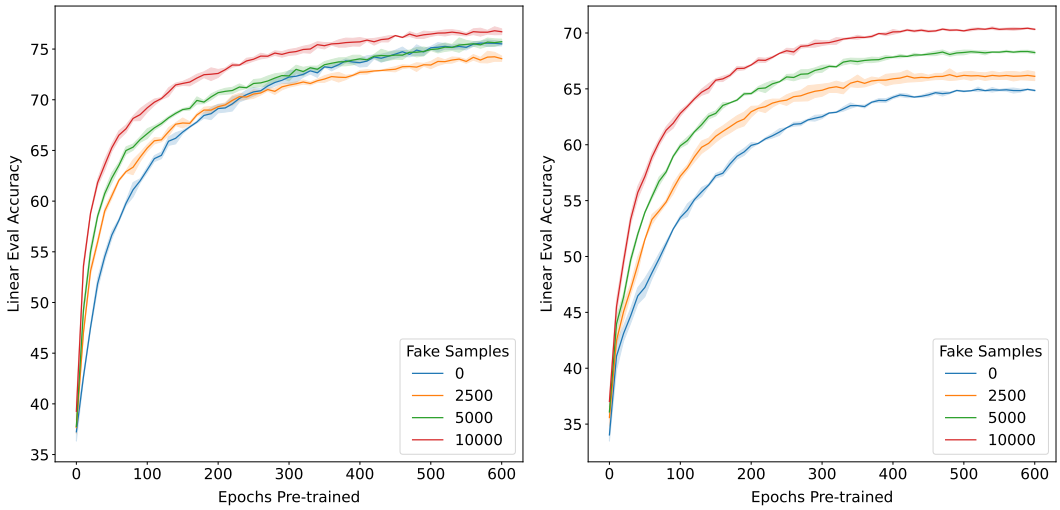
An important implementation choice was the mini-batch sampler during the pre-training process. By default, if the real and fake datasets were combined, a random batch would be selected from the combined dataset. This is fine when dealing with same size real and fake datasets, but in the case where we have discrepancy between the dataset sizes this could be an issue. This stems from the assumption that a consistent, highly imbalanced mini-batches between real and fake data may cause training issues. Thus, it was decided to make the first set of mini-batches have an equal probability distribution between real and fake. However, once the smaller dataset is completely used in the given epoch, this leads to the last set of mini-batches being either completely real or fake. It is not clear what is the superior sampling method, unfortunately there was not enough time to have a satisfactory comparison of the sampling methods, but it is a potential avenue to investigate in the future.

The two optimizers used in the SimCLR codebase are Adam [9] and Layer-wise Adaptive Rate Scaling (LARS) [13]. The default optimizer used in the SimCLR paper and the official codebase is LARS with dynamic hyper-parameters based on batch size. However, in this unofficial codebase the LARS model is experimental and not fully tested. Indeed, the implementation used for this project slightly modified the LARS code as a bug was found. The Adam optimizer is an official PyTorch implementation, but it is not commonly used for SimCLR and requires fixing an initial learning rate. Since it is unclear which optimizer to use, both optimizers were used in the experiments performed for this project.

5 Experimental Analysis

Since we are most interested in the small-data regime our methods will be tested on a small image dataset, specifically a 5000-sample subset of CIFAR-10, which from now on will be called small CIFAR-10. This will be paired with varying amounts of generated data. As stated earlier, we are comparing our method to SimCLR pre-trained on small CIFAR-10.

We first must generate a fake dataset by using a GAN trained on the small CIFAR-10 dataset. As mentioned previously, a data-efficient StyleGAN2 is trained on the data, producing a trained Generator. The fake dataset is created by sampling many images from the trained Generator. To gain a robust understanding we will experiment with multiple sizes of our generated image set Z that we augment to our original image set X. This will allow us to observe the impact of the amount of augmented GAN samples on the overall performance. We compare a SimCLR model pre-trained



(a) Adam Optimizer

(b) LARS Optimizer

Figure 3: Fake samples improve accuracy for pre-trained models in the low epoch regime. Given our epoch setup, the 10000 fake sample setup performs best across the board. The only difference in the experimental setup in these subplots is the optimizer used, Adam or LARS. When using the Adam optimizer, as the models pre-train for more epochs, the baseline model overtakes the 2500 fake samples setup and matches the 5000 fake samples setup. In the case of the LARS optimizer, the number of fake samples has a clear positive association with the linear eval accuracy.

on just small CIFAR-10 to a SimCLR model pre-trained on small CIFAR-10 and our augmented GAN datasets. Small CIFAR-10 has 5k training images, so we choose to make our augmented GAN datasets have 2.5k, 5k, and 10k images.

SimCLR is pre-trained for 600 epochs in every model setup for three trials and with a batch size of 128, a ResNet-18 [7] backbone, and the following default settings: temperature: 0.5, use blur: False, color jitter strength: 0.5. The optimizer for pre-training was either Adam with a learning rate of $3 \cdot 10^{-4}$, or LARS with the default SimCLR hyper-parameters. The linear evaluation is performed by fine-tuning the linear head using cross-entropy loss. This is done for 100 epochs with the Adam optimizer using a learning rate of $3 \cdot 10^{-4}$. We record the top-1 accuracy from the CIFAR-10 test set, where top-1 accuracy is our conventional accuracy metric: proportion of examples where the prediction matches the label.

Note that no hyper-parameter tuning was done in any experiment. Ideally, using a larger batch size and more pre-training epochs would be desired. Additionally, the LARS optimizer is experimental, and the Adam optimizer is not normally used for SimCLR. Overall, the compute budget and time was limited for this project, and so it was decided that this setup was sufficient.

5.1 Main Results

At every 10 pre-training epochs, the pre-trained model undergoes the linear evaluation protocol, producing a test accuracy. This is done for all model setups and the results can be found in Figure 3. The results show that the 10000 fake sample setup performs the best across all epochs. When using the Adam optimizer, the two other fake sample setups initially perform better than the baseline, but around epoch 250 the baseline outperforms the 2500 fake sample setup and ends up matching the 5000 fake sample setup. In the LARS setting, we have a consistent trend that more fake data leads to better performance. However, these accuracies are lower than the ones obtained when using Adam, even though LARS is known to perform better in SimCLR. As mentioned previously, this may be due to the codebase used for the SimCLR model.

Model	Optimizer	Linear Eval Accuracy (%)
5000 Real	Adam	75.51 ± 0.21
5000 Real + 2500 Fake	Adam	74.05 ± 0.27
5000 Real + 5000 Fake	Adam	75.73 ± 0.25
5000 Real + 10000 Fake	Adam	**76.71 ± 0.29**
5000 Real	LARS	64.86 ± 0.06
5000 Real + 2500 Fake	LARS	66.12 ± 0.19
5000 Real + 5000 Fake	LARS	68.25 ± 0.13
5000 Real + 10000 Fake	LARS	**70.323 ± 0.06**

Table 1: Test accuracy results when using the final pre-trained model after training for 600 epochs. In both optimizer settings, the 10000 fake image setup performs achieves the highest accuracy. In the case of the Adam optimizer, the baseline, 5000 real, matches or beats the other fake image setups, but when using the LARS optimizer, it falls behind all fake image setups.

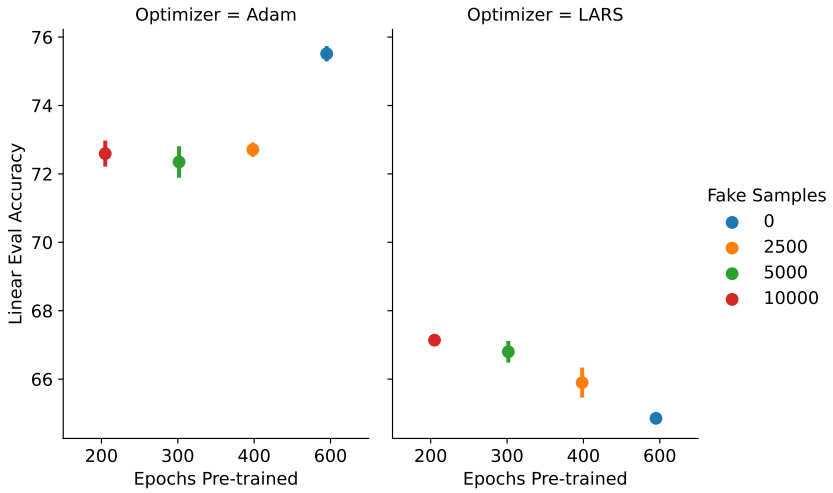


Figure 4: A fair computational comparison on pre-training across the different setups. For example, when using 5000 fake samples the dataset is twice as large, so 300 epochs would be equivalent computation to 600 epochs. When using Adam, having no fake samples gives the best performance, and all the fake sample methods perform worse than the baseline but similar to each other. In the LARS setting, the more fake samples used, the better performance.

The exact numerical results for the final pre-trained model in each setup is found in Table 1. Overall, there is some promise in these results. Given our epoch budget, we do see that some of the fake data augmentation methods outperform the baseline, especially when using a large amount of fake data.

Although it is important to note that under these compute constraints, we cannot assume that any model was *fully* trained to convergence. Typically models like SimCLR are pre-trained for around 1000 epochs and with a larger batch size for best performance, which was not possible here due to compute constraints. Additionally, Adam is not the recommended optimizer for SimCLR, whereas LARS is the preferred optimizer for SimCLR. However, the LARS used is experimental and could possibly have bugs since it is out performed across the board by Adam with arbitrary hyper-parameters versus SimCLR specific hyper-parameters for LARS. Thus, it is important to note that without computing until full convergence we cannot fully claim that the GAN augmentation improves the representations. However, for our compute constraints and optimizers it is clear the 10000 GAN samples as data augmentation improve our representations.

5.2 Learning Efficiency

It is of interest to observe the learning efficiency when using this strategy. How does this data augmentation affect the amount iterations or mini-batches it takes to reach a reasonable accuracy? By having a fair computation comparison, we can examine if any method learns faster. Given the epoch range in the experiments, the following epoch values are used to create a fair comparison: 600 epochs for the baseline or 0 fake sample setup, 400 epochs for 2500 fake samples, 300 epochs for 5000 fake samples, and 200 epochs for 1000 fake samples.

Results with equal computation across both optimizers are found in Figure 4. When using Adam, having no fake samples gives the best performance, and all the fake sample methods perform worse than the baseline but similar to each other. In the LARS setting, the more fake samples used, the better performance. There are some key caveats to take in consideration for this experiment. For one, given that we pre-trained for 600 epochs, we are forced to use at most 200 epochs when comparing the 10000 fake sample method, which is clearly far from convergence. It would be interesting to see how these results would change if we ran this equivalent computation experiment given a higher epoch range. Additionally, even though the baseline achieves the best accuracy for this experiment, it still does not outperform the 10000 fake image setup's raw accuracies at later epochs. Lastly, the LARS setup shows promising results for the fake augmentation method, but due to the substantial accuracy differences compared to Adam and noting that Adam is not normally used for SimCLR, we must take these results with a grain of salt. Overall, finding a good optimizer and training setup with some hyper-parameter tuning could potentially lead to more impactful results once we have confidence in each method's final accuracy and convergence.

6 Discussion and Prior Work

There has been some research that aims to improve self-supervised representation learning algorithms through input data augmentation. However, these methods mainly look at transformations of the original dataset, such as cropping, translations, rotations, blurring, etc. Here we augment the dataset with data not directly transformed for our original real data, but with a generative model trained using that original real data, specifically a GAN. Previous work in this area mainly is based on data augmentations already baked into the small CIFAR-10 baseline. This method applies GAN generated images as a data augmentation and compares with the default baseline.

Our results show that given our compute range and optimizers, good quality generated images from a GAN can improve self-supervised representations. This new form of data augmentation for self-supervised representation learning algorithms can outperform SimCLR pre-trained on the original dataset with typical data augmentations. This is quite interesting because most previous data augmentation work only used simple transformations of the initial data, but here we are basically creating completely new data with the use of a trained Generator. One can argue that this is similar since the GAN is trained from the initial data, and thus is a functional transformation of the initial data, like the other data augmentations. However, I argue that the GAN process is much more involved than cropping or rotating the image because it is noise that is transformed into our generated images and cannot be seen as a simple functional transformation of the initial data.

7 Conclusion

The idea of using fake or generated data from a powerful deep generative model such as the StyleGAN2 to augment the input data for a self-supervised representation learner such as SimCLR seems to hold promise. Throughout our experiments we saw improved accuracy on the CIFAR-10 test set with some of our GAN augmented models. This was especially noticeable when augmenting with 10000 fake images. However, due to the many caveats stated earlier, it is important to view these results as motivation to perform more experiments not as complete answers. As mentioned, there are much needed experimental improvements and hyper-parameter selections to perform.

Even though some of the results were positive, we would still need to perform these experiments with quite a bit more compute to truly conclude that this data augmentation improves the representations

of SimCLR. We know that given our computation range, this technique results in improved representations, however, it would be very interesting to pre-train these models to convergence at higher accuracies and then observe the results.

There are some open directions to take using the overall concept of this method. Thoroughly experimenting with all the different data augmentations commonly used and seeing what combinations of data augmentations, including this method, performs best. This could create a new, definite workflow to augment data for self-supervised learning. Additionally, scaling up these experiments not just to more epochs but using different data such as larger subsets of CIFAR-10, or other data sources such as CIFAR-100 [10], STL-10 [4], and Tiny ImageNet [11]. Using alternative self-supervised models such as Bootstrap Your Own Latent (BYOL) [6] or Momentum Contrast (MoCo) [3] would be an interesting direction as it would push this technique to its limit, demonstrating all its potential effects. Lastly, investigating the different ways of sampling between the real and fake images per mini-batch may be worthwhile.

Overall, this method shows some promise in being a new form of data augmentation and it would be interesting to explore it in more depth.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[3] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[4] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/coates11a.html.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[6] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[8] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[10] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[11] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[13] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv: Computer Vision and Pattern Recognition*, 2017.

[14] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *arXiv preprint arXiv:2006.10738*, 2020.