Top-k Training of GANs: Improving GAN Performance by Throwing Away Bad Samples (Reproduced)

Ajay Arokiasamy[1], **Abhilash Pulickal Scaria**[2] , **Aditya Gowrish Menti,**[3]
aea1n21@soton.ac.uk[1], aps1n21@soton.ac.uk[2], agm2g21@soton.ac.uk[3]
32677359[1],33124639[2],33168482[3]

ABSTRACT

This is a reproduced paper of the "Top-k Training of GANs: Improving GAN Performance by Throwing Away Bad Samples" paper that is published by Samarth Sinha et.al, on 14 Feb 2020. In this paper we are trying to analyse and reproduce the results that are obtained in the original paper. Generative adversarial networks (GANs) can improve their performance with no additional computational cost just by modifying a simple code in the generator parameters of GAN. Using the available resources, we reproduced the experiments and observed the results. Both vanilla GAN and state of the art models like DCGAN showed improvement in performance when top-k training was used.[1]

1 INTRODUCTION

Generative adversarial networks (GANs) are generative models with two neural networks that train and update alternatively using unsupervised learning with implicit density estimation. In an unsupervised learning task, Generative modelling involves automatically discovering and learning the patterns in input data. Then the model is used to generate new outputs that plausibly could have been an original dataset example. GANs can be divided into two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real or fake. The two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples. GANs are an interesting topic in the research community, They can be used in multiple problem domains to create realistic outputs for the problems.

In this paper we are trying to reproduce the results of the main paper "Top-k Training of GANs: Improving GAN Performance by Throwing Away Bad Samples" paper that is published by Samarth Sinha et.al. We successfully implemented the Top-K training method for GANs. It is a one line code modification of the generator algorithm during its update time. With this modification we can discard the samples that are far from the original distribution. The experiments on toy dataset(Gaussian Mixture) which was fully explained were re-implemented. The FID scores and loss curves were calculated to observe the performance improvement on CIFAR 10 when top-k training was used.

2 WORKING OF GANS

During training, the generator eventually becomes better at creating images that look real, while the discriminator gets better at telling them apart. The process reaches equilibrium when the discriminator can no longer distinguish real from fake images. Thus, if the discriminator is well trained and the generator manages to generate real-looking images that fool the discriminator, then we have a good generative model: we are generating images that look like the training set. After this training phase, we only need the generator to sample new (false) realistic data. We no longer need the discriminator. Note that the random noise guarantees that the generator does not always produce the same image (which can deceive the discriminator). The generator G and the discriminator D are jointly trained in a two-player minimax game formulation. The minimax objective function is:

[1]https://github.com/COMP6248-Reproducability-Challenge/COMP6248-Top-k-Training-of-GANs-Reproduced-

$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{\text{data}}}\log D_{\theta_d}(x)+\mathbb{E}_{z\sim p(z)}\log\left(1-D_{\theta_d}\left(G_{\theta_g}(z)\right)\right)\right]$$

3 METHODOLOGY

The top-K Training method is a simple modification of the code of the GAN generator. The parameters of the generator in a GAN are updated on every mini-batch of generated outputs by using the Top-K samples of the discriminator's prediction output. So, we simply zero out the gradients from the elements of the mini-batch corresponding to the lowest discriminator outputs. More formally, we modify the generator update step from Equation:

$$\theta_G \leftarrow \theta_G - \alpha_G \sum_{\max_k\{D(Z)\}} \nabla_{\theta_G} V(D,G)$$

where D(Z) is the discriminator's output for all entries in the mini-batch Z. Therefore, as training progresses, the discriminator, D, can be seen as a scoring function for the generated samples: The higher the score the closer generated data is to the real data and a sample that is far from the real data distribution will have a lower score. By performing the top-k operation on the discriminator predictions, we are only updating the generator on the 'best' generated samples in a given batch, as scored by the discriminator.

When the training starts the generator and the discriminator are at a starting phase where the data generated is not close to the real data but has a lot of information in it that is useful for further training. So, it won't be helpful to throw out gradients from samples scored poorly by the discriminator at the beginning of training – it would just amount to throwing out random samples, which is roughly equivalent to simply using a smaller batch size. Thus, we set k = B, where B is the full batch size, at the start of training and gradually reduce it over the course of training. In practice, we will decay k by a constant factor, γ , every epoch of training to a minimum of k = ν. We use the minimum value so that training doesn't progress to the point of only having one element in the mini-batch.

The paper proposes a simple method to improve the training of GAN that leverages the critics' outputs to filter out "unrealistic" samples and uses only top-k realistic samples of discriminator output to update the generator. The paper discusses the improvements in mode recovery when the top-k method is used for training on a toy example(Mixture of Gaussian). The paper also shows the observation of toy examples that top "unrealistic" samples, examples tend to push away from the nearest modes. The experiments show that the method can be easily applied to various GAN models and improve over the baseline on CIFAR-10 and MNIST datasets.

4 IMPLEMENTATION AND ANALYSIS

The paper implements the top k training on a toy task in which the target distribution is a mixture of Gaussian. We used the same experimental setup but with a reduced number of iterations due to computational complexity. A 4-layer MLP with 256 hidden layers was used for both generator and discriminator(critic). The loss used was Binary cross-entropy. The model was trained with a batch size of 256 for 10000 iterations. The optimizer used was Adam with a learning rate of 10^{-4}. The k is initially set to the batch size(256) then k is decayed using a decay factor of 0.99 until it reaches 192 (75% of initial batch size). This experiment is done to discuss how top k training can reduce mode dropping. Mode dropping: The model is not able to represent all modes in the actual data in the output. The model generates only a subset of individual mixture components. As the metrics used to evaluate how top k training reduces mode dropping was not fully mentioned in this paper so we referenced[2] and unofficial code implementation of these metrics.
The metrics used are given below:

1. Modes recovered: Indicates the number of modes the model was able to reproduce.
2. High quality samples: Indicates the percent of samples that lies at most 4 standard deviations away from nearest mode.

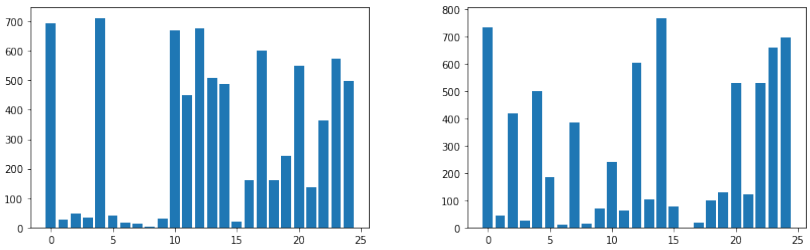

Figure 1: Mode Recovery, With top k (Left) and Without top k (Right)

A histogram of number of modes recovered when the mode is 25 is shown in Figure 1.

Due to the high computational requirements, we were not able to train it on the same number of iterations due to which the results observed are not as good as those observed in the paper. But still, we can observe an increase in the percent of high-quality samples and modes recovered when top-k training is used.

The observations obtained are given in the below table where K = 1 implies top-k training was used and K = 0 implies top-k training was not used.

K	Mode	Modes Recovered	High Quality Sample %
0	25	24	70.27%
1	25	25	77.15%
0	64	21	12.58%
1	64	22	23.38%

In order to explain why throwing away the bottom, k samples cause an improvement in performance the paper does another experiment using the same toy dataset. They draw samples from the generator's prior distribution(z) and use these samples as input to the generator to generate samples. Then the direction of the nearest mode for each of these samples is calculated referred to as oracle direction[2]. After gradient descent using top-k and bottom-k samples. The same z is then given to the generator and the displacement of samples is calculated. The cosine similarity between this displacement and direction is calculated. A value close to 1 indicates that the points are moving towards the mode and a value close to -1 indicates the points moving away from the mode. We were not able to implement the cosine similarity as the method to calculate oracle direction is not mentioned. But we did observe the loss increasing and the model diverging when bottom k samples were used for training.

The paper implements top k training on different GANs to check their effectiveness. We used the MNIST dataset for training the MLP GAN and CIFAR-10 dataset for training the DCGAN. The MNIST dataset contains 60,000 training images and 10,000 testing images of handwritten digits while the CIFAR-10 dataset contains 60000 32x32 color images in 10 different classes. The GANs were modified such that the top k outputs of the discriminator are used to train the generator. The 'k' value decays by a factor over each batch in the epoch until a specified minimum batch size. The performance of these modified GANs was then compared against their primitive counterparts. There is a reduction in the generator loss due to this. The generator loss observed when training DCGAN for 40K iterations on the CIFAR-10 dataset with and without top-k training is given in Figure 2.

There are different ways by which we can measure the performance of GANs. Inception Score is a metric that estimates the quality of the fake images based on how well the Inception model classifies them as belonging to one of the 1000 known classes. But there is no comparison between the fake images and real images. Fretchlet Inception Distance(FID) is another metric that is used to calculate the distance between the feature vectors of the real images usually generated by Inception Net and the feature vectors generated by the Generator in GAN. A lower FID score means that the GAN is producing fake images which are very close to the real images. While training, we aim to minimise the FID score. It is desirable for the GAN to have high fidelity and diversity in generating images.

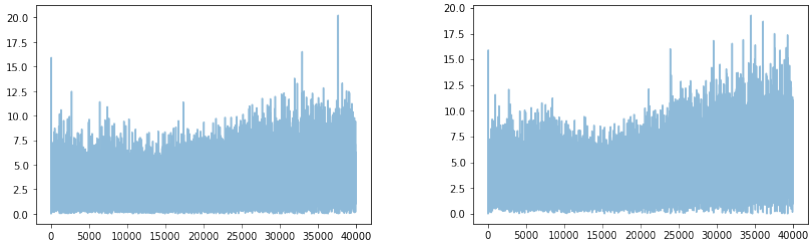

Figure 2: Generator Loss, With k (Left) and Without k (Right)

The FID is considered to be better than the Inception Score as it is robust to noise and distortions. It is also able to detect intra-class mode dropping.

Due to computational limitations, instead of FID 50k used in the paper, we were able to implement the same but at a smaller scale. We ran the DCGAN for 70 epochs on the CIFAR-10 dataset with and without the top k training and compared the performance.

DCGAN	**FID Score**
Without Top-K training	114
With Top-K training	104

5 Conclusion and Future Scope

We re-implemented the "Top-k Training of GANs: Improving GAN Performance by Throwing Away Bad Samples" paper and observed that the Top-K training model will improve the performance of the GANs without much difference in computational cost. Consecutively, we observed improvement in mode recovery when the Top-K method is used. We tested this method with CIFAR-10 and MNIST datasets on DCGAN and observed improvement in FID score and a reduction in generator losses. Due to the computational limitations, we were only able to reproduce the results on a smaller scale. Yet, we were able to observe a performance improvement. The paper discusses how it can improve recovery of long-tail distribution but the actual experimental setup or observations were not provided. Experimental setup to show how top-k training improves mode recovery was provided but whether it reduces mode collapse or vanishing gradient problem was not mentioned. In future work we can focus on these aspects and check whether these problems can be solved.

References

[1] Sinha, Samarth, Zhengli Zhao, Anirudh Goyal, Colin Raffel and Augustus Odena. "Top-k Training of GANs: Improving GAN Performance by Throwing Away Bad Samples." arXiv: Machine Learning (2020): n. pag.

[2] S. Azadi, C. Olsson, T. Darrell, I. Goodfellow, and A. Odena. Discriminator rejection sampling. arXiv preprint arXiv:1810.06758, 2018.

[3] S. Sinha, H. Zhang, A. Goyal, Y. Bengio, H. Larochelle, and A. Odena. Small-gan: Speeding up gan training using core-sets. arXiv preprint arXiv:1910.13540, 2019.

[4] Eric J. Nunn, Pejman Khadivi, Shadrokh Samavi. Compound Frechet Inception Distance for Quality Assessment of GAN Created Images.arXiv preprint arXiv:2106.08575,2021.

[5] Yu, Yu Zhang, Weibin Deng, Yun. (2021). Frechet Inception Distance (FID) for Evaluating GANs.