Does Distribution Smoothing Aid Autoregressive Sample Quality?

Adithya Raju Ganeshan **Ilona Toikka** **Serkan Arda Yilal**

Abstract

In this project we worked on reproducing the results from the paper "Improved autoregressive modeling with distribution smoothing". Although we did not manage to reproduce the exact quantitative results due to time and computation constraints, we still saw the same trend in the results. From a visual perspective we confirmed that distribution smoothing helps in achieving better sample quality. Additionally, we provided quantitative results for the MNIST data set and a time analysis for training the models and sampling. Our implementation can be found in this Github repository.

1 Introduction

So far, generative models have been quite successful on tasks such as semi-supervised learning, generating missing samples and detecting out of distribution samples. While it is the case, the density estimation of the data distribution we achieve with the models like GANs and VAEs have been implicit for GANs and intractable for VAEs. On the other hand, autoregressive models compute the density estimation process with an explicit and tractable way which makes them more understandable [9]. Although autoregressive models achieve good success on tasks such as compression, speech synthesisation [8] while providing a likelihood value, it is not as successful as GANs on generating samples qualitatively [11]. The reason for this can be associated with the problems caused by its sequential nature such as manifold hypothesis and compounding error.

In order to solve that problem, the paper we have chosen “Improved Autoregressive Modeling with Distribution Smoothing” proposed the solution of convolving noise with the existing data distribution to smooth it and make it easier to learn this smoothed distribution with an autoregressive model. This approach continues with the denoising process to derive the original data distribution by either gradient-based denoising (single-step) or by using a second conditional autoregressive model(two-step) to derive the original distribution from the smoothed distribution [8].

In this project, we tried to show whether the proposed solution really helps us on sampling more qualitative images from smoothed distributions on datasets such as MNIST and CIFAR10. While doing that, we used different noise levels and calculated FID, Inception, and BPD scores to compare the quality of samples. Due to time and computational source limitations, we had to keep number of epochs low compared to original paper and it prevented us to achieve similar scores considering autoregressive models require longer time to train compared to other generative models. Although that is the case, we have observed a similar trend in the improvement of scores and samples qualities as in the original paper. Additionally, we calculated scores for MNIST database which was not provided in [8]. Also we trained models with different noise levels to find the optimal value for each dataset. Lastly, we did time analysis for training and sampling processes which showed us that it takes a lot of time to accomplish these processes in autoregressive models.

The remaining part of this report is structured such that it will cover to project in a more detailed way. In section 2, we will talk about the related work that led the authors of this paper to propose a solution like that. In section 3, we will get into details of the method that we used in order to produce

the results of this paper. Throughout the section 4, we will mention the content of the datasets we have used along with the state of art score values for these datasets. As we continue with Section 5, we will include the experiments we implemented to validate the results of the paper. Since we didn't have the same resources and environments to implement the paper results completely in the same way, we will mention the challenges that we have faced in section 6 regarding these constraints. As we conclude our insights that we got from the project in section 7, we will continue to discuss about the solution in an ethical aspect and mention its possible societal effects in section 8. With section 9, we will end our report with a self-assessment by discussing what grade we deserve with the provided reasons.

2 Related work

The concept of using tractable conditional distributions for estimation of original data distribution on autoregressive modelling started to get attention with The Neural Autoregressive Distribution Estimator (NADE) [6]. By using the approach of minimizing the average negative log-likelihood, they implemented a Bayesian network for estimation of data distribution.

This approach also influenced usage of autoregressive models on image sampling within PixelRNN and PixelCNN models [14]. The image sampling is achieved by modelling the pixels with discrete values by using a distribution implemented with softmax layer. With the sequential predictions of pixels in an image, the samples are generated with log-likelihood scores while keeping the models scalable and tractable which is not possible in comparably faster and more qualitative sampling models like GANs. It must be noted that despite their tractability, the process of sampling was still slow for these models.

In order to improve the lacking points of PixelCNN such as slow training and complex model structure, a modified PixelCNN model called PixelCNN++ [10] is implemented. By providing changes in the PixelCNN such as using discretized mixture likelihood for the pixels instead of softmax layer for faster training, downsampling of input for multi-resolution processing, and short-cut connections for securing information throughout sub-sampling and up-sampling processes in ResNet layers, the new model achieved higher sample quality and likelihood while speeding up the whole process.

Another improvement on PixelCNN has been implemented by using causal convolutions and self-attention to have a high bandwidth access over a huge amount of information in order to utilize long-term dependencies in past experience. With that improvement, PixelSNAIL [1] achieved better BPD (Bits per dimension) scores compared to other autoregressive models including PixelCNN++.

When it comes to the improvement that the paper we have chosen proposed, it is based on the usage of "randomized smoothing" that was proposed in [2] to keep ImageNet classifiers robust against adversarial perturbations. These smoothed models were used for outputting the most likelihood classes for the inputs under isotropic Gaussian Noise. Although the approach is similar in our paper, the "randomized smoothing" was applied on the data distribution instead of the model itself that we are training which is PixelCNN++. By deploying this smoothing operation, it is aimed to keep the learned distribution as close as possible to the original data distribution which is achieved by the robustness that we gain against aforementioned problems like manifold hypothesis and compounding error.

3 Methods

In this section we first discuss how distribution smoothing is done and how it is incorporated into modeling the data distributions. Then we give a short overview of PixelCNN++, the autoregressive model used to model the distributions.

3.1 Modeling with distribution smoothing

The authors of [8] proposed distribution smoothing to improve quality of the generated samples as it has shown good results in adversarial defense. Additionally, autoregressive models suffer from two problems. One of them is the manifold hypothesis: data lies in a low-dimensional manifold and thus the distribution has sharp transitions making it hard to model. The second issue is compounding error:

if the model makes a mistake predicting a pixel value it is likely that it will make more mistakes but nevertheless high likelihood is assigned to a resulting non-realistic sample.

The idea behind distribution smoothing is convolving the original data distribution with a noise distribution which more simply means adding noise to the original images $\mathbf{x}$ before passing them to the model. The Gaussian distribution with a zero mean and a small variance is chosen as a smoothing distribution for this task. In this way we get the smoothed or noisy data distribution $p_\theta(\tilde{\mathbf{x}})$. The next step is to obtain a denoised distribution for which there are two options: gradient-based approach (one-step) and training a second conditional autoregressive model (two-step). For the one-step process, we can denoise the samples $\tilde{\mathbf{x}}$ directly by using the gradient of $p_\theta(\tilde{\mathbf{x}})$. To compute the denoised samples $\bar{\mathbf{x}}$ we apply the following equation: $\bar{\mathbf{x}} = \tilde{\mathbf{x}} + \sigma^2 \nabla_{\tilde{\mathbf{x}}} \log p_\theta(\tilde{\mathbf{x}})$. However, this approach does not provide a likelihood estimate making it harder to compare to other methods. The two-step denoising involves modeling the denoised distribution $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ with a conditional model. To achieve this a concatenation of a noisy image and a clean image is fed into the model instead. Thus, instead of the regular $N \times N \times C$ input size it becomes $N \times 2 * N \times C$ so in simple terms the clean image is appended under the noisy image.

3.2 PixelCNN++

The PixelCNN++ model introduced by [10] is used to model the two distributions $p_\theta(\tilde{\mathbf{x}})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$. PixelCNN++ predicts each pixel value based on all the previous pixel values, therefore factorising the probability density function of an image as $p(\mathbf{x}) = \prod_i p(x_i|x_{<i})$. The main difference between the original PixelCNN++ implementation and the version used together with distribution smoothing is that each individual pixel density $p(x_i|x_{<i})$ is modeled by a discretised mixture of logistics in the original implementation. However, as distribution smoothing implies having a continuous distribution, a continuous mixture of logistics is used instead to model the smooth and denoised distributions $p_\theta(\tilde{\mathbf{x}})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ respectively.

4 Data

We used two well known data sets in our project: MNIST [7] and CIFAR-10 [4]. The MNIST data set contains 28×28 pixel grayscale images of handwritten digits with 60K training and 10K test samples. The CIFAR-10 data set consists of 32×32 pixel colour images of 10 classes with 50K training and 10K test samples. Before training the models both data sets are mapped to be in the range $[-1, 1]$.

The state-of-the-art for likelihood based generative models is achieved by the PixelSNAIL model [1] on CIFAR-10 reaching a pbd (bits per dimension) score of 2.85. In comparison the regular PixelCNN++ model [10] achieves a score of 2.92 on the same data set. It is worth mentioning that adding distribution smoothing as done by the authors in [8] does not lead to a better bpd score which could be explained by the fact that good sample quality does not mean a good likelihood score [13]. In general the state-of-the-art models can reach an FID score of around 2 while a PixelCNN++ with distribution smoothing achieves a score of 29.83. For the MNIST data set the state-of-the-art result based on FID comes from a method called sliced iterative normalising flows [3] which achieves a score of 4.5.

5 Experiments and findings

5.1 Reproduce main results

Our first experiment was to try and reproduce the quantitative results from [8]. Thus, we set out to calculate the scores for one-step and two-step denoising on the CIFAR-10 data set. Additionally, we also set out to compute the scores for the MNIST data set. We trained two models per data set, one smoothed distribution model and one conditional denoised distribution model. Both models were trained using the same hyperparameters as mentioned in the repository of [8] but due to time and resource constraints we only trained each model for 100 epochs instead of 1000. The noise level was 0.3 for CIFAR-10 and 0.5 for MNIST.

The quantitaive results can be seen in Table 1. As we were not able to train the models for as long as the original paper our results are not as good as can be seen by the differences in scores reported

in the CIFAR column. Nevertheless, we can see a similar trend between the one-step and two-step denoising methods. In all cases the two-step denoising method leads to a better score. We can also see that both in the original paper and our reproduction the two-step method's FID score is roughly half of the one-step method. Interestingly, our BPD score ended up lower than the upper boundary that was mentioned in [8]. The reason for this can be the loss values converge earlier to the lowest possible values while the sample quality continues to increase as we train more which causes FID and Inception scores to require more training time to have better values.

We also wanted to find out whether we can reproduce the results visually and therefore we present samples drawn from the smoothed distribution that were denoised by the gradient method (one-step) and from the denoised distribution (two-step) in Figure 1 and Figure 2 respectively. As we can see in Figure 1 and Figure 2, the images sampled from both MNIST and CIFAR10 databases look similar to the images included in [8].

	CIFAR10			MNIST		
Method	**Inception**	**FID**	**BPD**	**Inception**	**FID**	**BPD**
One-step	4.58 (7.50)	115.75 (57.53)	-	2.11	17.83	-
Two-step	5.47 (7.84)	56.72 (29.83)	3.17 ($\leq$ 3.53)	2.36	16.61	4.1

Table 1: Inception, FID and BPD scores for one-step and two-step denoising methods. Higher score is better for inception and lower score is better for FID and BPD.

Figure 1: Samples from one-step denoising on CIFAR10 with noise level = 0.3 and MNIST with noise level = 0.5.

Figure 2: Samples from two-step denoising on CIFAR10 with noise level = 0.3 and MNIST with noise level = 0.5.

5.2 Time analysis

In this section we provide the results for training and generation times for smoothing and two step de-noising experiments. All the experiments made use of NVIDIA v100 cards except for the image generation for MNIST two-step de-noising which made used of NVIDIA Quadro K6000. The results in 2 and 1 provide an initial insight into the long training and image generation times for autoregressive modelling. We consider this a meaningful contribution in addition to the reproduction of the results. Furthermore, the time taken for gradient based single step de-noising is negligible as the methods does nor require a model for image generation.

DATA SET	TASK	TIME (Hrs)/2000 images
CIFAR10	smooth	8
	2 step	10
MNIST	smooth	4
	2 step	8 *

Table 2: Image generation time for smoothing and 2 step de-noising

DATA SET	TASK	TIME (Hrs)/100 epochs
CIFAR10	smooth	60
	2 step	80
MNIST	smooth	12
	2 step	20

Table 3: Training time for smoothing and 2 step de noising

6 Challenges

The code available made available by [8] was inspired or based on [5] [12]. The Pytorch code [5] used the discretized version of the log likelihood mixture of logistics loss function while the [8] removed the discretization of the loss function due to the addition of noise. [10] reported this removal of discretization while adding noise. Given this, a considerable amount of time was spent analysing and implementing the loss functions. Results regarding the training time's and resources (GPU's) were missing in the actual which further proved that we had under estimated the training and image generation times. For the entire pipeline of learning the smoothed distribution and sampling images proved to be a time consuming process. Moreover, training a second model for de-noising and sampling images added to the time taken for one single experimental parameter. Given the availability of just one NVIDIA v100 card limited the number of parallel experiments that could be conducted. The authors [8] trained the improved model for 1000 epochs, given this we were limited on the number of epochs we could train our models for leading to just 100 epochs of training.

Further more, [10] mentioned the need to train for longer time in order for the model to converge to 2.92 BPD. We argue that the images generated from our experiments do not reflect the best state of the model, hence, the FID and Inception scores should be considered as a trend rather than actual scores.

7 Conclusion

We successfully adapted the code provided by [5] [12] to reproduce the idea presented by [10]. We observed improvements in the performance metrics especially the Fid and Inception score of PixelCNN++ by adding noise even though we only trained our model for 100 epochs rather than 1000 epochs. In addition to all the knowledge gained, we learned its smart to choose a paper which explains the mathematics behind the idea clearly and provides solid proofs for any assumptions and convergences. Having said this, the difficulty lies in converting the formulas into code that could be deployed efficiently on computational resources such as CPU's and GPUS's. Furthermore, the availability of resources such as time and computation power needs to considered while choosing a paper to reproduce. It's always wise to choose a project which provides a strong code repository, as this serves as a good starting point to validate the overall idea of the paper by running few epochs

of training and noticing the metrics. Finally, we learnt Deep learning takes considerable time and patience for replication of results from papers.

8 Ethical consideration, societal impact, alignment with UN SDG targets

One issue with training generative models is that it consumes a lot of energy. The authors of PixelCNN++ report that it took 10 hours on 8 Maxwell TITAN X GPUs to achieve a bpd score of 3.0 and then an additional 5 days of training for the model to converge to 2.92. It is debatable whether it is really worth it to increase the score by 0.08 if the carbon footprint of training becomes much higher. This arguably excessive use of computational power is also not inline with the UN sustainable development goals which strive for sustainable consumption. Additionally, although we visually see an improvement in image quality by applying the model with distribution smoothing, the quality is still not good enough to deploy the model in an ethically sensitive field like medicine.

Moreover, another possible ethical problem for this model and also for all generative models is the fact that they can be misused to generate fake data. For example, lately there is a huge increase on usage of "deepfakes" which are the images generated by GAN models trained on people's face images to swap with the face of another person. While it has a wide usage in industries such as entertainment and health sector, it has some serious drawbacks for post truth politics and harassment of women in the form of revenge porn, manipulation of the public, intervention to the elections, violations of personal data protection rights and intellectual property rights[15]. Although these images are generated mostly on GANs, with the increasing sampling quality of autoregressive models, there is a danger of involvement of these models in generation of such "deepfake" images which can cause a huge societal impact.

9 Self Assessment

As we proposed in the beginning of our project, we achieved to reproduce the results presented in the chosen paper. Although we didn't achieve similar scores due to the time and computational resource limitations, we showed that the trend observed in our project proves the success of the paper.

When it comes to additional experiments about the paper, we have calculated quantitative results for MNIST in addition CIFAR10. Despite the fact that we couldn't have time to search for optimal hyperparameter values for different noise distributions and number of mixture of logistics due to resource limitations, we made a time analysis on the experiments we implemented to show how much resource these experiments consume.

Given our results and the time dedicated for the experiments conducted in addition to all the knowledge gained, we believe we qualify for minimum grade of C.

References

[1] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. In *International Conference on Machine Learning*, pages 864–872. PMLR, 2018.

[2] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

[3] Biwei Dai and Uros Seljak. Sliced iterative normalizing flows. *arXiv preprint arXiv:2007.00674*, 2020.

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[5] Caccia L. Apixcellcnn++ pytorch implementation.

[6] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 29–37. JMLR Workshop and Conference Proceedings, 2011.

[7] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] Chenlin Meng, Jiaming Song, Yang Song, Shengjia Zhao, and Stefano Ermon. Improved autoregressive modeling with distribution smoothing. *arXiv preprint arXiv:2103.15089*, 2021.

[9] Walter Hugo Lopez Pinaya. Autoregressive models — pixelcnn.

[10] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[11] Harshit Sharma. Auto-regressive generative models (pixelrnn, pixelcnn++).

[12] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[13] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[14] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. PMLR, 2016.

[15] Betül Çolak. Legal issues of deepfakes.