Better Data Augmentation is All You Need to Improve Autoregressive Modeling

Cuong Dao cuongdd@kth.se
Fynn van Westen fynn@kth.se
Sam Yousefzadegan Hedin samhedin@kth.se

Abstract

In this work, we reimplement the paper "Improved Autoregressive Modeling with Distribution Smoothing" by [7]. Our work focuses on the reproducibility of the paper. Specifically, an autoregressive model is trained on noisy data and denoising is performed either with a gradient-based approach or another conditional model. We show results that support the main claim of the original paper. By adding randomized noise to training data, one can improve the performance and the quality of generated images of autoregressive models. The influences of initialization with pre-trained weights is also investigated. Moreover, insights are given about the training and evaluation settings and the challenges in the replication of the original paper. Our implementation is available at `https://github.com/samhedin/advdl_project`

1 Introduction

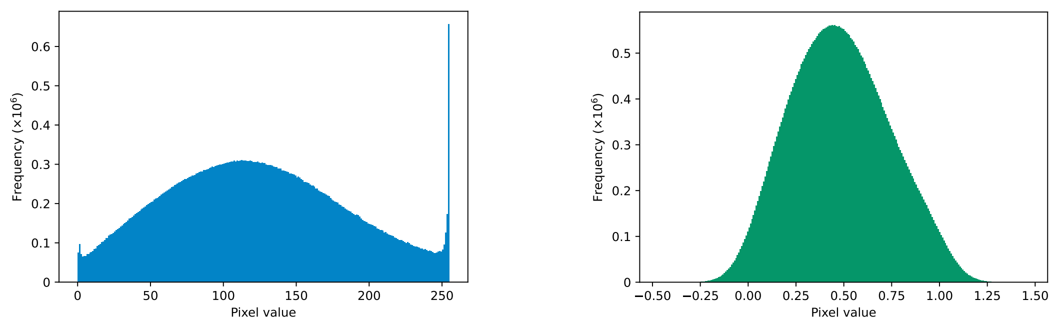
The authors of [7] pinpoint two problems with current autoregressive models. 1) A low-dimensional manifold in the underlying data distribution. 2) the "compounding error" in the generation process of autoregressive models. The authors propose a randomized smoothing approach to autoregressive modeling. Unlike previous works [3], which focused on improving the model's representational power, [7] applies smoothing to the data distribution $p_{\text{data}}(\mathbf{x})$. To that end, [7] perturbs the dataset with specific noise, which smooths the data distribution and makes the images noisy. Subsequently, a reverse smoothing process, either directly through gradient-based denoising or through a second conditional model, is applied to the noisy data to generate realistic images.

In this work, we focus on reproducing the process of learning the smooth distribution and then denoising to generate authentic images with an autoregressive model. We conduct experiments on the CIFAR-10 dataset [6], and evaluate the performance of the models using the Inception score. Overall, we find that the central claim in the paper is correct and that smoothing the distribution of input data indeed helps the autoregressive model learn the distribution of the data.

2 Background

The original paper proposes that autoregressive generative models perform poorly in image generation because of the training data distribution. For example, when plotting the distribution of CIFAR-10, Figure 1 it can be seen that it has a complicated density function with sharp peaks, which is typical for datasets collected from natural observations.

The distribution is due to pixels of images densely accumulating around a small subset of the possible pixel values instead of being distributed over the whole color range. This makes sense since pixels with similar values often make up a large part of pixel space, like the light blue of a sky. Further, some colors are unlikely to be seen in natural images, which is the type of image that makes up CIFAR-10. As mentioned in the paper [7] it has been generally observed that some likelihood models, such as



(a) Regular distribution (b) Smoothed with Gaussian noise with $\sigma = 0.1$

Figure 1: Marginal distribution of greyscale-level dataset. *Left:* original CIFAR-10 dataset. Two spikes including on for the edge value of 0 and another for value of 255. *Right:* Distribution of CIFAR-10 smoothed by a Gaussian noise with $\sigma = 0.1$, the distribution resembles a Gaussian, which is smoother and easier to model.

normalizing flows, show a pathological behavior on this kind of distribution. Since a conditional of a autoregressive model is a 1-d normalizing flow, given a fixed context, it is most probable that distributions with complicated densities affect or hinder the learning of autoregressive models.

The paper also points out another problem with autoregressive models, namely, "compounding errors". The model relies on the previously generated context to generate the following data (e.g., pixel in image). Thus, if a mistake happens in previous steps, the model will extrapolate from those errors during the generation process.

3 Related work

3.1 Autoregressive modeling

Autoregressive generative models over high-dimensional data $\mathbf{x} = (x_1, \ldots, x_n)$, based on chain rules probability, factor the joint distribution as a product of conditional probabilities:

$$p(\mathbf{x}) = p\left(x_1, \ldots, x_n\right) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

The model is then trained to model $p(x_i | x_{1:i-1})$. Research in autoregressive modeling often focuses on improving the capability of the model, i.e., increasing the receptive field and memory [2]. A natural architecture for this task is a recurrent neural network such as PixelRNN [12] which is composed of multiple layers of two-dimensional LSTM and generates images pixel-by-pixel. The major limit of PixelRNN lies in its sequential nature, which is too slow for many applications. An alternative is PixelCNN [12, 13] which uses masked or shifted convolutions. This ensures that the model only sees pixels before the currently generated pixel. PixelCNN++ [10] improves on PixelCNN with several changes, including 1) discretized logistic mixture likelihood instead of 256-way softmax for output distribution, which reduces the computational power needed significantly. 2) conditioning on the whole pixel instead of sub-pixels as in PixelCNN [13]. The paper we implement in this work [7] utilizes PixelCNN++ as the backbone model for all image generation tasks.

3.2 Evaluation metrics for image generation

The results in this report are evaluated quantitatively and with inception scores [9]. Inception scores are given to a generative network based on the quality and diversity of the images it produces. The Inception V3 is a network for image classification trained on ImageNet [8]. Image quality and image diversity are estimated by classifying images with the inception network. Specifically, a network generates high-quality images if the inception network is certain about its classifications – if the inception network has classified banana, it should do so with high confidence. Further, a network generates diverse images if the inception network makes many different classifications for the generated images – a network that only generates a very specific type of dog is considered a bad network. The inception score is widely used, but not without its issues. As noted in [1], while the classification accuracy of the inception v3 network tends to be robust against the randomness involved in training neural networks, its certainty about classifications is not. The paper demonstrates inception score differences up to 11.5The paper notes that using Inception Score for generative models trained on anything but ImageNet is inappropriate. It is partly inappropriate because the classes in ImageNet are very different from the classes in CIFAR-10. In our case, we use inception scores since they are used in the original paper [7], however, we will not place much weight on it.

4 Method

4.1 Modeling

The authors propose to circumvent the problem of complex distributions by decomposing the density estimation problem into a smoothed data modeling problem and an inverse smoothing problem – referred to as denoising. This has the benefit of bringing the initial distribution in line with a well-known distribution favorable for likelihood models, typically a Gaussian distribution. The smoothing is done by adding random Gaussian noise to the data. The change of distribution of this process for the CIFAR-10 dataset, is illustrated in Figure 1.

In the first subproblem of this decomposition, the autoregressive network attempts to model the smoothed data distribution $p_\theta(\tilde{\mathbf{x}})$. Image generation is a two-step process that begins with sampling from the learned, smoothed distribution, followed by a denoising method to invert the data smoothing. The paper evaluates two methods for this purpose.

The first one is single-step denoising (SSD) that uses the gradients of the original network $p_\theta(\tilde{\mathbf{x}})$ and the fact that the distribution of the noise is known. If the noise is Gaussian, SSD is described by Equation 1.

$$\bar{\mathbf{x}} = \tilde{\mathbf{x}} + \sigma^2 \nabla_{\tilde{\mathbf{x}}} \log(p_\theta(\tilde{\mathbf{x}})) \tag{1}$$

SSD is computationally efficient, but is lacking as it only provides a point estimate of $\bar{\mathbf{x}} = \mathbb{E}\left[\mathbf{x}|\tilde{\mathbf{x}}\right]$.

The second one is to model the conditional distribution $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ using a neural network of the same architecture. An autoregressive denoising network is trained by giving it clean input $\mathbf{x}$ and its noisy counterpart $\tilde{\mathbf{x}}$: $\mathbf{x}$ and $\tilde{\mathbf{x}}$ can be stacked together as $[\mathbf{x}, \tilde{\mathbf{x}}]$ and used as input to the network. The output of the network $\hat{\mathbf{x}}$ will now be half noisy and half clean $\bar{\mathbf{x}}$. The loss is calculated between x and the clean part $\bar{\mathbf{x}}$ of the output. To denoise an image with the network, $\tilde{\mathbf{x}}$, the image is stacked together with itself, creating $[\tilde{\mathbf{x}}, \tilde{\mathbf{x}}]$ and the output can be treated similarly as before.

In the paper and our reproduction, a PixelCNN++ with 5 ResNet blocks, 160 filters and 10 mixtures of logistics to model the output, is used for both $p_\theta(\tilde{\mathbf{x}})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$.

4.2 Evaluation

The original paper does not have adequate details on the settings to compute the Inception score. A common practice is to have $50,000$ images, running over 10 splits to compute Inception score for image generation in CIFAR-10. However, due to the limited computational resources, that setting is not feasible for us. Therefore, in all of our experiments, we evaluated the Inception score with $7,500$ images over 10 splits.

5 Data

In our work, we use the standard dataset CIFAR-10 [6] that contains 60K RGB images (50K for training and 10K for validation). The images are of size 32×32 and distributed over 10 classes. At the time of our work, the state-of-the-art results for image generation task on CIFAR-10 w.r.t Inception score is 10.11 by [4], and FID score is 2.10 by [11].

Version	Inception score
PixelCNN++ baseline	2.78 ± 0.08
Sampling from $p_\theta(\tilde{\mathbf{x}})$	1.44 ± 0.02
Single-step denoising	1.87 ± 0.03
Sampling from $p_\theta(\mathbf{x}\|\tilde{\mathbf{x}})$	2.36 ± 0.07

Table 1: The inception scores for generated images with different scheme. *First row:* baseline – PixelCNN++ trained on clean CIFAR-10; *Second row:* samples from PixelCNN++ trained on smooth data; *Third row:* samples from single-step denoising; *Forth row:* samples from the conditional model $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$.

6 Experiments

6.1 Training on smooth data and denoising

The first experiment focuses on the effect of smoothing when training from scratch. For learning the smoothing distribution, we train a PixelCNN++ from scratch on smooth data for 100 epochs. For reverse smoothing, i.e., denoising, we either apply gradient-based approach as specified in Equation 1 or train another PixelCNN++ from scratch on smooth data, with stacked input $[\mathbf{x}, \tilde{\mathbf{x}}]$. Moreover, for comparison, we also train a baseline model – a PixelCNN++ from scratch on clean data. All models are trained for 100 epochs. We use Adam optimizer [5] with learning rate $\eta = 0.0002$, and step learning rate scheduler with a step size of 1 and a weight decay factor of 0.999995. Figure 2 shows samples generated from those models.

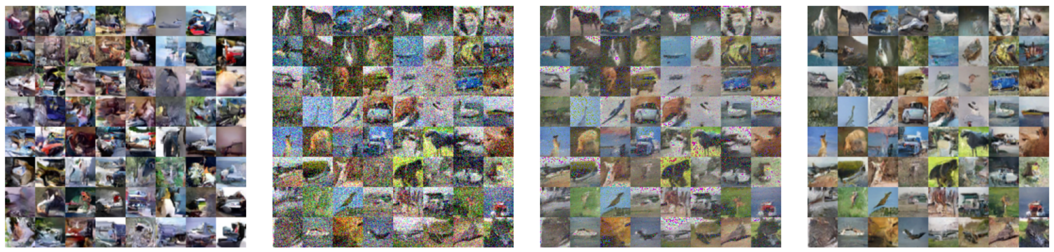


Figure 2: From left to right: **Column 1:** samples from the baseline PixelCNN++ model. **Column 2:** samples from $p_\theta(\tilde{\mathbf{x}})$. **Column 3:** samples from "single-step denoising" from $p_\theta(\tilde{\mathbf{x}})$. **Column 4:** samples from $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$

A substantial improvement that the original paper shows over the default PixelCNN++ model is that the original model often produces nonsense – the images may be sharp and clear but distorted unnaturally. We see a similar trend in this experiment. The images generated from the baseline PixelCNN++ paper are sharp, as shown in the first column of Figure 2, but it is hard to pinpoint what is on many of the images. The images sampled from the network trained on smooth data resemble more realistic and natural things. Single-step denoising does reduce noise but visible artifacts remain, whereas samples from $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ are generally of good quality. We note a slight decrease in sharpness relative to the standard PixelCNN++, but in return, it produces very believable and realistic images.

6.2 Initialization with pretrained PixelCNN++ weights

In this experiment, we investigate whether using pretrained PixelCNN++ would help with learning the smooth distribution. The models were initialized with weights from a PixelCNN++ trained on clean CIFAR-10 for 890 epochs [1]. The first baseline model was created by continuing to train on clean data for 100 epochs. The second model was trained for another 100 epochs on smooth data. Samples were generated from the baseline network as a source of comparison. Samples were also generated from the smoothed network, and the samples were denoised with single-step denoising.

[1]`https://github.com/pclucas14/pixel-cnn-pp`

Model	Inception score
PixelCNN++ trained for 890 epochs	2.94
PixelCNN++ fine-tuned on clean data	1.36
PixelCNN++ fine-tuned on smooth data, followed by SSD	1.98

Table 2: The inception scores for PixelCNN++ in different training schemes.

The generated images can be seen in Figure 3, and the corresponding inception scores are available in Table 2.

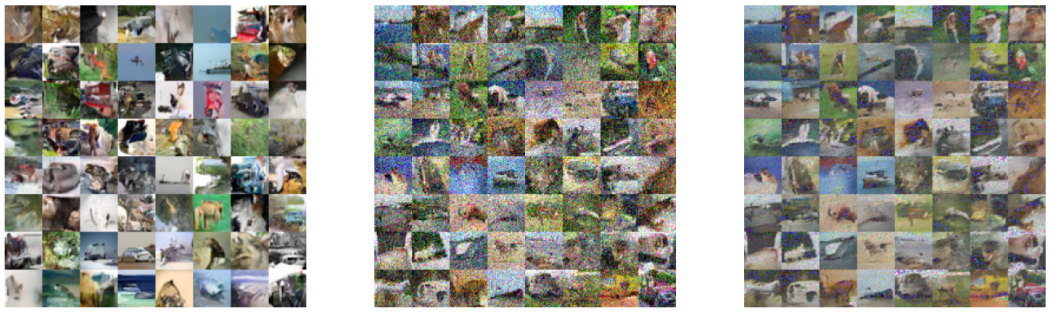


Figure 3: Generated images on models fine-tuned on a pretrained PixelCNN++. From left to right: **Column 1:** samples from baseline PixelCNN++ fine-tuned on clean data for 100 epochs; **Column 2:** samples from $p_\theta \tilde{\mathbf{x}}$ trained on smooth data for 100 epochs with initialization from pretrained weights; **Column 3:** samples from single-step denoising for images in Column 2.

The experiment's goal was to see if a pre-trained network could easily be improved by training on smooth images for a shorter period. The experiment shows that single-step denoising works in that it improves the score of images generated by a network trained on a smooth distribution. However, the original PixelCNN++ network performs better than the single-step denoised version. This might be because the network did not have time to learn the gaussian noise in the short time it was trained.

7 Challenges

The original paper [7] lacked information about the actual number of epochs trained. Only a constant on the maximum epochs of $1,000$ was found in their Github repository, which was unrealistic for us due to our restrictions of computational resources. Therefore, we can not determine how our results would hold up on a similar training time. Furthermore, there is no reference to a specific implementation of Inception score in the paper. We use a publicly available version [2] which might be different from the original paper's implementation. As pointed out in [1], Inception score depends on the implementation details, the number of samples used when computing the score; therefore, the results are unreliable.

8 Conclusion

Since objective measurements of the quality of generated images are so hard to come by, arguing that we have definitely or definitely not reproduced the original paper seems impossible. Further, it is impossible to say how much more training would have affected the results.

However, we have successfully trained an autoregressive network on noisy images and seen that both single-step denoising and using another network for denoising improve the quality of the generated images. We believe that both our results and the results of the original paper show improvement over

[2]`https://github.com/sbarratt/inception-score-pytorch`

the baseline PixelCNN++ model. For example, the baseline PixelCNN++ model generates images that look like abstract shapes, where it is hard to say what is on the image. Meanwhile, our and the original papers' images look like more realistic things.

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

The most pressing SDG target is ending poverty. While we cannot see a specific way for the paper's results to be used to reduce poverty, historically, the best way to do so is through cumulative technological development and trade. This paper can be regarded as a small addition to the technological knowledge of humanity, and therefore it might help reduce poverty, even if we do not know how. It is hard to evaluate the ethics of machine learning without discussing the incredible energy and hardware costs. Especially autoregressive models require immense amounts of computational resources, which might make some people claim that they should not be used. However, not exploring machine learning properly might lead to many missed ideas and inventions. Placing a ban on it because of its environmental impact might hurt the environment in the long run.

10 Self-Assessment

We believe our project qualifies for an A for the following reasons:

- We have successfully implemented the methods presented in [7]. We have been able to produce results that support the main claim of the original paper in terms of Inception score on CIFAR-10 dataset.
- We have managed to conduct an extra experiment using pretrained PixelCNN++ model trained on clean data. This helps bring more aspects when training the model on smooth data.
- We have provided more details that are not mentioned in the paper regarding the experiments and evaluation. This supports further attempt to reproduce the original paper and produce sensible results.

References

[1] Shane Barratt and Rishi Sharma. A note on the inception score, 2018.

[2] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *CoRR*, abs/2103.04922, 2021.

[3] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. *CoRR*, abs/1902.02918, 2019.

[4] Dongjun Kim, Seungjae Shin, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. Score matching model for unbounded data score, 2021.

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

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

[8] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[9] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

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

[11] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space, 2021.

[12] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *CoRR*, abs/1601.06759, 2016.

[13] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016.