Distribution Augmentation for Generative Modeling

Heewoo Jun [*1] **Rewon Child** [*1] **Mark Chen** [1] **John Schulman** [1] **Aditya Ramesh** [1] **Alec Radford** [1]
Ilya Sutskever [1]

Abstract

We present distribution augmentation (DistAug), a simple and powerful method of regularizing generative models. Our approach applies augmentation functions to data and, importantly, conditions the generative model on the specific function used. Unlike typical data augmentation, DistAug allows usage of functions which modify the target density, enabling aggressive augmentations more commonly seen in supervised and self-supervised learning. We demonstrate this is a more effective regularizer than standard methods, and use it to train a 152M parameter autoregressive model on CIFAR-10 to 2.56 bits per dim (relative to the state-of-the-art 2.80). Samples from this model attain FID 12.75 and IS 8.40, outperforming the majority of GANs. We further demonstrate the technique is broadly applicable across model architectures and problem domains.

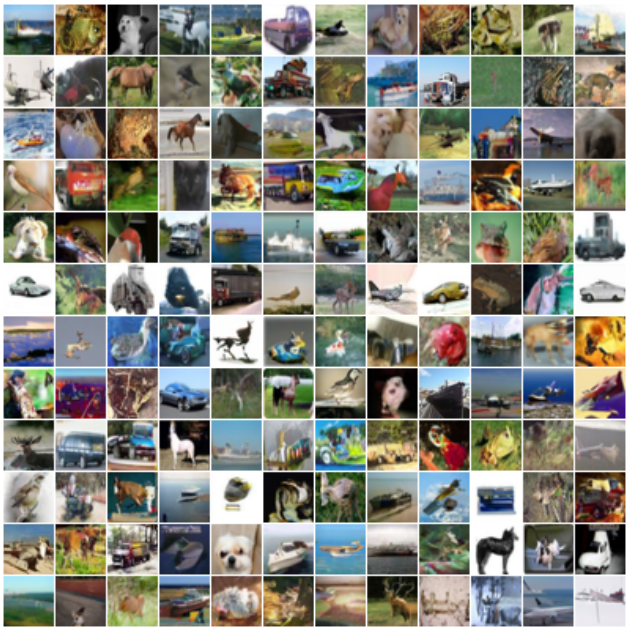


Figure 1. Unconditional samples from a large autoregressive model trained on CIFAR-10 with DistAug. In addition to attaining state-of-the-art likelihoods, at temperature 0.94 this model generates samples competitive with or superior to many GANs (FID 12.75, IS 8.40).

1. Introduction

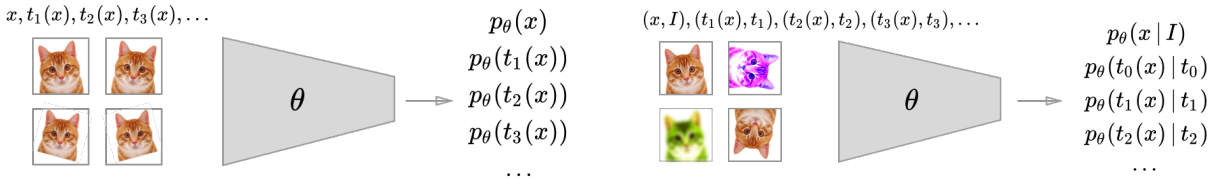
Data augmentation is an indispensable tool in the training of deep neural networks, especially for discriminative (Cubuk et al., 2019), semi-supervised (Xie et al., 2019a;b), and self-supervised (Hénaff et al., 2019; He et al., 2019) vision tasks. However, data augmentation has not played a role in many of the recent advances on pixel-level generative modeling (Salimans et al., 2017; Chen et al., 2018; Menick & Kalchbrenner, 2018; Kingma & Dhariwal, 2018; Ho et al., 2019; Parmar et al., 2018; Child et al., 2019). This is partly caused by the difficulty of applying data augmentation to generative modeling: aggressive augmentation functions can change the data distribution the model learns, distorting its samples and incurring a penalty to its log-likelihood.

To illustrate, consider a supervised learning setting where the distribution of interest is over a small number of class variables y given a high-dimensional image x. In this case, many transformations $t(x)$ can be applied to x such that the conditional distribution $p(y|t(x))$ remains unchanged. Generative models, in contrast, seek to learn the distribution $p(x)$ itself, and as such any $t(x)$ we use must result in images which are also likely under the original distribution. Even mild augmentation functions, however, (like adding Gaussian noise, shifting/rotating and padding with null pixels, or cutting out portions of the image), result in images very unlikely under the original distribution. Models trained with these augmentations may fit the original distribution poorly and generate distorted samples.

This work introduces distribution augmentation (DistAug), a method of regularizing generative models even under augmentations which distort the data distribution. The

[*]Equal contribution [1]OpenAI, San Francisco, California, USA. Correspondence to: Heewoo Jun <heewoo@openai.com>, Rewon Child <rewon@openai.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.



(a) Data augmentation

(b) DistAug

Figure 2. Illustration of the main differences between data augmentation and distribution augmentation (DistAug). DistAug can use more aggressive transformations than data augmentation because it conditions the generative model on the transformation type.

idea is to simply learn a density $p_\theta(t(x) \mid t)$ of the data under some transformation t, conditioned on the transformation itself. The function t can be drawn from a large family of transformations, including state-of-the-art aggressive data augmentations from supervised learning (Cubuk et al., 2019) and self-supervised transformation functions (Gidaris et al., 2018; Kolesnikov et al., 2019). To estimate the un-transformed density and draw samples from it, the transformation t is simply set to the identity function.

This can be interpreted as a unique data-dependent and model-dependent regularizer similar to multi-task learning. We find it substantially outperforms standard methods, allowing us to train an autoregressive CIFAR-10 model to 2.56 bits per dim (versus the previous 2.80 state-of-the-art) and attain sample quality on par with most GANs. We show evidence that this increased performance is both due to the *scalability* of the method relative to standard regularization, and also to the beneficial *inductive biases* received from well-chosen augmentation functions.

We also examine the applicability of DistAug to other scenarios by testing different domains (language modeling and neural machine translation), different models (autoregressive models and invertible flows) and different architectures (self-attention based and convolution-based). We find our technique, with proper dropout tuning, yields improvements in most cases we test. We release our model weights and code at `https://github.com/openai/distribution_augmentation`.

2. Background

2.1. Density estimation

We consider the task of estimating a parametric density $p_\theta(x)$ under the negative log-likelihood objective:

$$L(p_\theta, D_n) := \mathop{\mathbb{E}}_{x \sim D_n} [-\log p_\theta(x)] \tag{1}$$

The model θ can be any likelihood-based generative model, including autoregressive models, invertible flows, variational autoencoders, and more.

2.2. Data augmentation

Standard data augmentation introduces a family of transformation functions T where each $t \sim T$ transforms the data in a particular way, except for the identity transformation I. The objective is to maximize the likelihood over transformed samples:

$$L_{\text{data}}(p_\theta, D_n, T) := \mathop{\mathbb{E}}_{x \sim D_n} \left[\mathop{\mathbb{E}}_{t \sim T} [-\log p_\theta(t(x))] \right] \tag{2}$$

The specifics of the distribution over T are determined on a case-by-case basis, typically with regard to the performance of the model when evaluating non-transformed data on a held-out validation set. In the case of images, the types of transformations are usually significantly more conservative than those used in supervised learning, and are commonly limited to just horizontal flipping. In the case of natural language, transformations might include mild augmentations like replacing words with their synonyms.

This is because using more aggressive transforms will distort the distribution the model learns, often resulting in higher loss on the validation set. Additionally, model samples will include the augmentations, which is usually not desired.

3. DistAug

Distribution augmentation (DistAug), similar to data augmentation, introduces a family of transformations T to the training process. Unlike data augmentation, however, the model learns the density of a transformed data sample, conditioned on the specific transformation:

$$L_{\text{dist}}(p_\theta, D_n, T) := \mathop{\mathbb{E}}_{x \sim D_n} \left[\mathop{\mathbb{E}}_{t \sim T} [-\log p_\theta(t(x)|t)] \right] \tag{3}$$

A graphical depiction of this process appears in Figure 2. DistAug allows the model to adapt its density estimate of the sample based off which transformation is being applied. In practice, this means that regardless of how radically a transformation distorts a sample, adding it to the training process need not affect the model's estimate of the original samples as long as the model has enough capacity.

In principle, any number and type of transformations can be applied, regardless of whether they preserve the information in the data and regardless of how similar the transformed samples are to the original. However, if $|\det(\partial t/\partial x)| > 0$, it is possible to invert the transformed density. For permutation-motivated transforms like rotation, jigsaw, and reordering of words, it is possible to estimate and sample from the density in non-canonical order.

As a varied and wide family of transformations can be used with DistAug, we hypothesize that it can be used to train more powerful models without overfitting than is possible with standard regularization methods. In section 4.1, we provide experimental evidence supporting this claim.

3.1. DistAug as a data-dependent regularizer

A helpful way of rewriting the objective of a generative model trained with DistAug is as follows:

$$L_{\text{dist}}(p_\theta, D_n, T) = \mathop{\mathbb{E}}_{x\sim D_n}[-\log p_\theta(I(x)|I)] + \mathop{\mathbb{E}}_{x\sim D_n}\left[\mathop{\mathbb{E}}_{t\sim T}\left[-\log\left(\frac{p_\theta(t(x)|t)}{p_\theta(I(x)|I)}\right)\right]\right] \quad (4)$$

$$= L(p_\theta, D_n) + \Omega(p_\theta, D_n, T). \quad (5)$$

In other words, the objective under DistAug is equivalent to the original objective, plus some data and model-dependent regularizer Ω whose difficulty is dependent on T. The model incurs a penalty if it models the original distribution much better than any of the transformed distributions.

3.2. DistAug as multi-task learning

The objective in Eq. 3 can also be interpreted in the framework of multi-task learning, where the separate tasks in this case are to learn the transformed distributions. We speculate that, in a similar nature to multi-task learning, when the family of transformations T is chosen with knowledge of the original distribution and model class, DistAug may lead to better generalization than techniques which penalize complexity uniformly.

In section 4.2, we provide evidence that even *small* models trained with DistAug achieve gains in validation performance relative to standard training methods, suggesting that gains are not only due to the scale of augmentations applied, but also due to the helpful inductive bias of certain transformations.

4. Experiments

In this section, we provide evidence supporting our main claims that 1) DistAug scales to larger model sizes than standard regularization techniques, 2) specific choices of distributions can lead to improved performance even for small models, and 3) the conditioning signal leads to lower error and improved samples.

In this section, we primarily study an autoregressive model (the Sparse Transformer (Child et al., 2019)) and its performance on the natural image benchmark datasets CIFAR-10 and ImageNet-64. In section 5, we test whether it applies to other data types, model architectures, and generative objectives.

In all image experiments, models using DistAug were tuned to find the best subset of augmentation functions from rotation (Gidaris et al., 2018), spatial transposition, color swapping, jigsaw shuffling (Noroozi & Favaro, 2016), and an ensemble of data augmentations from supervised learning called RandAugment (Cubuk et al., 2019). To condition the model on these transformations, we replace the start of sequence embedding vector with the sum of augmentation embeddings. For example, if we ask the model to generate images with rotation and transposition, we choose one each from four rotation embeddings and two transposition embeddings. Each option is selected uniformly at random. For rotation, spatial transposition, color swapping, and jigsaw recordering, we also permute the positional embeddings supplied to the Transformer, which results in a smoother learning and an increase of around 0.01 BPD at convergence.

Baseline regularization methods include just model regularization (dropout and weight decay), as well as model regularization and limited data augmentation (horizontal flipping). Also, all DistAug runs were tuned for the best value of dropout, which was typically much lower than the corresponding baseline run.

Detailed hyperparameter settings for experiments are available in the Supplementary Material.

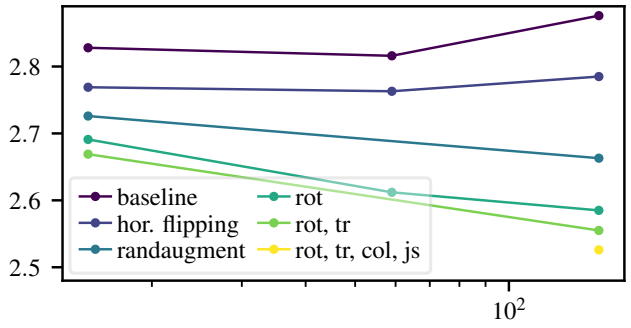
4.1. Scalability of DistAug

Here, we assessed the performance of DistAug relative to standard regularization methods when both increasing model size and data size.

4.1.1. CIFAR-10

We took a CIFAR-10 model with three different capacities: 15M parameters (smaller than typically used), 58M parameters (the previous state-of-art), and 152M parameters (more than is typically used), and assessed their performance when trained with DistAug and with standard regularization methods.

The results are presented in Figure 3. Without DistAug, the larger 152M parameter models perform worse than their 58M parameter counterparts, suggesting that a good regularization technique would be to simply reduce the number of



(a) CIFAR-10 validation bits per dim of different augmentation strategies across model sizes (in millions of parameters). Baseline and horizontal flipping do not use DistAug.

DistAug type	Dropout	BPD
-	60%	2.88
horizontal flipping	40%	2.79
randaugment	25%	2.66
rot	40%	2.59
rot, tr	40%	2.56
rot, tr, col	10%	2.58
rot, tr, col, js	5%	2.53

(b) CIFAR-10 test bits per dim using a 152M Sparse Transformer. Increasing levels of DistAug and less dropout yields increased performance.

Figure 3. Using large models with increasing DistAug strength can improve generalization to an extent that is not possible with standard regularization methods. Dropout and traditional data augmentation plateau at lower parameter count.

parameters in the model. With DistAug, on the other hand, performance continues to improve as we increase model capacity. The model with the most augmentations and the lowest dropout rate performs the best, attaining 2.53 bits per dim on the test set, compared to 2.79 for a similarly-sized model with tuned dropout and standard data augmentation.

It should be noted that this improved performance is not just with models with greater numbers of parameters–in fact, the 15M model trained with rotation and transposition, at 2.67 bits per dim, *also* improves upon the previous 58M state-of-the-art model by 0.13 bits per dim. We attribute this gain to a useful inductive bias from DistAug, and further elaborate on this phenomenon in section 4.2.

4.1.2. IMAGENET-64

We also evaluated models trained with DistAug on the larger and more complex ImageNet-64 dataset. We trained 152M and 303M parameter models for comparison with previous works. DistAug with rotation showed modest gains in performance, as seen in Table 1. We suspect that, as the baseline models do not require training with dropout, that their capacity is still insufficient to benefit fully from DistAug. Larger models will be needed for more substantial gains.

4.2. DistAug on small models

We observe that even relatively small models can benefit from DistAug, which is difficult to explain if it acts purely as a regularizer on the model's capacity. Drawing from DistAug's connections to multi-task learning explored in section 3.2, we hypothesize that the specific inductive bias of the transformations may aid in learning a generalizable solution as opposed to merely regularizing the model.

To test this, we apply single augmentation types to a 15M parameter CIFAR-10 model, reporting results in Table 2. Although each variation provides exactly one auxiliary task for the model to complete, the validation performance varies widely, from an increase of 0.13 bits per dim over the baseline (rotation) to a decrease of 0.06 bits per dim (with a set of fixed random orderings). Thus, we conclude that the task type is important and can lead to wide variations in performance.

Table 1. Validation bits per dim on the ImageNet-64 dataset. We found gains to be modest relative to CIFAR-10, suggesting that model capacity may still be too limited to fully take advantage of DistAug. DistAug still shows some improvement, however, which we discuss in section 4.2

Parameters	DistAug type	Validation BPD
152M	-	3.432
152M	rot	3.427
303M	-	3.425
303M	rot	3.419

It is unclear *why* certain augmentation types are so much stronger than others in improving density estimation. Take, for example, jigsaw, which effectively amplifies the amount of training data by a factor of $(2 \times 2)! = 24$ – larger than the factor of $4\times$ for rotation. In line with results from self-supervision literature (Kolesnikov et al., 2019), however, jigsaw turns out to be weaker than rotation. We speculate that rotation may just happen to provide a useful inductive bias for the specific model and datasets we use, and posit this as the reason it helps even for models which are not prone to overfitting.

Another interpretation is that the model is encouraged to learn the original data distribution and a neural circuit to generate augmented distributions from it. It is possible that certain augmentation types such as rotation are a simple

Table 2. CIFAR-10 performance for a relatively small 15M parameter models, trained with varying choices of a single augmentation function. Rotation and RandAugment (Cubuk et al., 2019) perform remarkably well, even outperforming larger state-of-the-art models, whereas random orderings or excess data decrease performance. The specific inductive biases of transformations thus can aid model learning, independent of their regularizing effect.

Augmentation	Validation BPD
Baseline	2.82
RandAugment	2.73
Jigsaw	2.78
Color	2.75
Transposition	2.73
Rotation	2.69
75% Fixed random order	2.88
75% ImageNet data	2.80

enough function to implement, and facilitate in learning the original data distribution in the process of jointly compressing the data and the corruption process. We found tiny models with 2M parameters plateaued to a worse likelihood than the baseline when DistAug was used. Because the model does not have enough capacity to learn this joint representation let alone all distributions, it does not learn any one distribution well. This is contrasted by sufficiently larger models that could eventually learn a more compact representation.

Such mechanics may lead the model to avoid rote memorization and improve generalization. We leave detailed examination of this mechanism to future work.

4.3. Importance of the conditioning signal

For a 32×32 image, 1 bit of conditioning signal provides only about 0.0003 BPD of information. But, the presence of this bit allows the model to estimate and sample from the original distribution much more effectively, as shown in Figure 4. While models trained without conditioning can still put significant probability mass on unaugmented data, during sampling augmented samples are drawn frequently and appear visually distorted, a phenomenon introduced in (Theis et al., 2015).

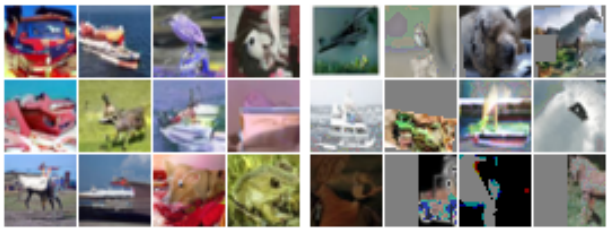
We provide quantitative evaluations of this behavior in Table 3 where heavy RandAugment normally used for larger images is applied 99% of the time. In the case of a RandAugment-conditioned model, the conditioning signal provides a gain of only 0.007 bits per dim likelihood yet improves the FID evaluation by over 20 points. We find that the model can still sample plausible objects and obtains a comparable FID to GAN models, whereas the model trained with traditional data augmentation rarely produces unaugmented samples, as shown in Figure 8b.

Moreover, the corruption process internalized by an unconditional augmentation model often can not be fixed by finetuning on the original data. For the above example with heavy RandAugment, we further finetune for 10 to 100 epochs with a varying amount of model regularization and find that the sample quality and the rate of unaugmented samples do not change much.

It is interesting, however, that when the unconditional model generates a sample from the original distribution, the sample quality is often quite high. There are clearly object-like images, demonstrating that augmentation *can* still help with sample quality of the original distribution, even without conditioning. But to ensure that the model can reliably sample from the original distribution instead of following its adjusted training distribution, conditioning must be used.

Table 3. Ablation on the conditioning signal

Augmentation	Conditioning?	BPD	FID	IS
RandAugment	no	2.715	42.0	6.36
RandAugment	yes	2.708	21.1	7.52



(a) With conditioning (b) Without conditioning

Figure 4. Even though the log-likelihood of unconditional and conditional augmented models can be similar, samples from DistAug are much improved visually and have a lower incidence of artifacts.

5. Comparison across architectures, domains, and other existing work

We tested whether the same technique applies to different model architectures, problem domains, and generative modeling objectives. We also attempted to contextualize our work in the body of existing generative modeling literature. Our results are summarized in Tables 4, 5, and 6.

5.1. Comparison with existing autoregressive models

We compare the best Sparse Transformer (Child et al., 2019) we could train with DistAug with the previous state-of-the-art self-attention based generative models. With an equal amount of parameters, simply adding rotation leads to a gain of 0.18 bits per dim (BPD) over the state-of-the-art, to 2.62. Larger models benefit even more: a 152M parameter

Table 4. Performance on the CIFAR-10 natural image density estimation benchmark. Well-tuned distribution augmentation leads to significant gains across model architectures, generative objectives, and datasets when compared with standard augmentation or regularization techniques. "rot, tr" corresponds to augmenting by spatially rotating images and transposing them which creates 8 left-right-aware reorientations of the image.

Model	Parameters	Regularization	DistAug	BPD
Autoregressive, Self-Attention				
PixelSNAIL (Chen et al., 2018)			-	2.85
Sparse Transformer (Child et al., 2019)	58M	dropout 25%	-	2.80
Sparse Transformer	58M	dropout 1%	rot	**2.62**
Sparse Transformer	152M	dropout 40%,hflip	-	2.78
Sparse Transformer	152M	dropout 40%	rot,tr	2.56
Sparse Transformer	152M	dropout 5%	rot,tr,js,c	**2.53**
Autoregressive, Convolutional				
PixelCNN (van den Oord et al., 2016b)				3.14
Gated PixelCNN (van den Oord et al., 2016a)				3.03
PixelCNN++ (Salimans et al., 2017)	53M	50%	-	2.93
PixelCNN++	53M	50%	rot, tr	2.88
PixelCNN++	53M	5%	rot, tr, col	**2.84**
Invertible flow				
Glow (Kingma & Dhariwal, 2018)			-	3.35
Flow++ (Ho et al., 2019)	32M	20%	-	3.08
Flow++	32M	1%	rot, tr	**2.98**

Table 5. Frechet Inception Distance (FID) and Inception Scores (IS) for models trained with DistAug on CIFAR-10, relative to existing state-of-the-art. We find that better sample quality tends to correlate with fewer augmentations, and only loosely follows likelihoods. When the sampling temperature is varied, moreover, networks trained with DistAug can generate samples with similar or better quality than GAN-based networks, despite not incorporating any optimization objective for sample quality.

Model	Parameters	Regularization	DistAug	BPD	Temperature	FID	IS
Sparse Transformer	58M	Dropout 1%	rot	2.62	1.0	37.5	6.41
Sparse Transformer	152M	Dropout 5%	rot,tr,js,c	**2.53**	1.0	42.9	6.85
Sparse Transformer	152M	Dropout 40%, hflip	-	2.78	1.0	27.8	7.16
Sparse Transformer	152M	Dropout 25%	randaugment	2.66	1.0	**14.7**	**8.18**
Sparse Transformer	152M	Dropout 40%	rot,tr	2.56	1.0	21.8	7.81
Sparse Transformer	152M	Dropout 40%	rot,tr		0.98	17.0	8.00
Sparse Transformer	152M	Dropout 40%	rot,tr		0.94	12.7	8.40
Sparse Transformer	152M	Dropout 40%	rot,tr		0.90	15.2	8.36
Sparse Transformer	152M	Dropout 25%	randaugment	-	0.94	**10.57**	**8.93**
PGGAN (Karras et al., 2017)							**8.80**
AutoGAN (Gong et al., 2019)						**12.42**	8.55
CR-GAN (Tian et al., 2018)						14.56	8.40
SN-GAN (Miyato et al., 2018)						21.7	8.22
WGAN-GP (Gulrajani et al., 2017)						29.3	7.86

Table 6. Performance on natural language generation benchmarks. Enwik8 is unconditional generation, whereas Neural Machine Translation (NMT) is conditional generation. DistAug increases performance on the tasks we studied.

Model	Parameters	Dropout	DistAug	
Enwik8				**Bits per byte**
Sparse Transformer (Child et al., 2019)	95M	20%	-	0.99
Sparse Transformer	95M	10%	document, sentence, word reversal	$\mathbf{0.968 \pm 0.001}$
En-De Neural Machine Translation				**BLEU**
Transformer (Vaswani et al., 2017)	210M	30%	-	26.23 ± 0.77
Transformer	210M	15%	sentence reversal	$\mathbf{26.82 \pm 0.24}$

Sparse Transformer attains 2.78 BPD using traditional data augmentation, but with aggressive DistAug gets 2.53 BPD, a relative gain of 0.25 BPD.

5.2. Comparison across architectures

5.2.1. PixelCNN++

We also compare with a model architecture variant which incorporates no self-attention, the PixelCNN++ (Salimans et al., 2017). The relative gain from introducing DistAug appears to be more limited than in the self-attention case, gaining 0.09 bits per dim to a total of 2.84. The reason for this difference is unclear, although we speculate that self-attention may be more capable of learning flexible routing patterns than a purely convolutional approach.

5.2.2. Flow++

Additionally, to test whether it may be just the quirks of the autoregressive objective which lead to gains from DistAug, we also test on a state-of-the-art invertible flow with variational dequantization, Flow++ (Ho et al., 2019). We added a conditioning embedding to every coupling network in order to allow the model to incorporate transformation information.

We find that the model trained with rotation and tranposition gains 0.1 bits per dim over the baseline, to a total of 2.98. When we inspected the learned conditioning embeddings, however, we found that many of them had similar values, and samples generated with different orientations looked visually similar. Thus, we conclude that the network did not learn to use our conditioning, and that similar gains would be seen with an unconditional model.

5.3. Comparison of sample quality relative to other generative models

We also compare the sample quality relative to existing work by calculating Frechet Inception Distance (FID) and Inception Scores (IS). We draw 10,000 samples and report scores in Table 5. Inception scores are calculated in 10 batches, and we report the mean.

We notice only a loose correlation between bits per dim and sample quality metrics, a phenomenon observed to be possible in (Theis et al., 2015). Instead, we notice that applying fewer augmentations tends to result in better sample quality for the model size considered in this work, with both conditional RandAugment and rot,tr performing similarly well.

The model trained with rot,tr,js,col, on the other hand, achieves high FID and low IS despite achieving the lowest bits-per-dim of any model. Samples of this model are shown in 8a. The reason for this disparity is unknown, but may be due to the limited capacity of the model to condition on a wide number of transformations.

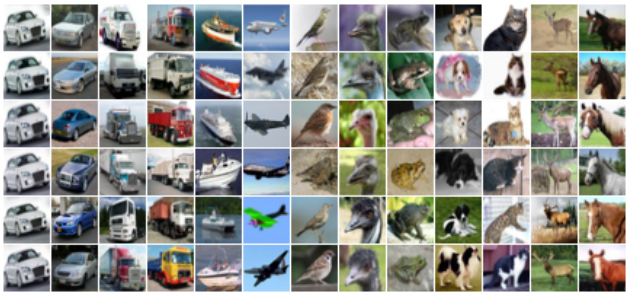


Figure 5. The top row is cherry-picked samples from a DistAug model where the augmentation is RandAugment (Cubuk et al., 2019). More representative samples are shown in Figure 7b. Subsequent rows are the nearest training data in the Inception embedding space. We found none of the generated samples were clearly copied pixel-by-pixel, although numerous, near identical copies of the white sedan are found in the training set.

When samples are generated by tempering the output distribution, as in common practice with autoregressive models, we find that sample quality matches or exceeds the best GANs in the literature. This is despite the fact that ours is a likelihood-based model that must cover all modes of the data, and incorporates no objective specifically tailored to promote sample quality. In Figure 5, we cherry-pick some of the best samples and show the nearest neighbors in the Inception embedding space. We found little evidence to suggest the model is merely memorizing data. One case of a white sedan did appear to be very similar to training examples, but the same car appears numerous times in the training data. Additionally, the sampled image has many differences on a pixel-by-pixel level.

5.4. Comparison on text domain

In principle, the benefits of DistAug need not be limited to images. Here, we explore augmentation for generative modeling of text on two tasks: neural language modeling and neural machine translation.

For language modeling, we factorize the prose probability in various ways. Namely, we consider independently reversing the order of sentences, words, and letters within a word. For example, for the last two options, `"the cat sat on the mat"` becomes `"mat the on sat cat the"` and `"eht tac tas no eht tam"` respectively. In general, this idea is equivalent to progressively factorizing a string x into (s)entences, (w)ords, and (c)haracters: $p_\theta(x) = \prod_i p_\theta(s_{t_i}|s_{t_{<i}})$, $p_\theta(s) = \prod_i p_\theta(w_{u_i}|w_{u_{<i}})$, $p_\theta(w) = \prod_i p_\theta(c_{v_i}|c_{v_{<i}})$ with shared orderings t, u, v at each level. On Enwik8, these augmentations lead to a gain of 0.022 bits per dim. We note that because fp16 training

and augmentation introduces a bit of variance in training, the reported result is an average over the best three runs out of ten selected based on validation error.

For Neural Machine Translation (NMT), we took the same Transformer as used by Vaswani et al. (2017) for the En-De translation task. As byte-pair encoding (BPE) (Gage, 1994) is used in these models, we only experimented with reversing the word order of a sentence (and not the characters within words) as to avoid changing the tokenization of normal sentences. Also, as NMT is a conditional generation task, augmentations can be applied to either the source or target sentence. We found augmenting the target with reversals increased BLEU from 26.2 to 26.9, as measured by the average of 5 runs. Augmenting the source we found to provide no additional value over augmenting the target.

While a number of strong augmentations exist for images, it is unclear how to best augment text, especially with the more recent frontends like BPE. Admittedly, improvement on text is modest compared to the image domain. Because DistAug depends on the availability of relevant transformations for the task, the effectiveness of this algorithm is limited to those data-limited domains where prior knowledge can be encoded as augmentation functions.

6. Related work

Distribution augmentation is situated in a large number of existing works on self-supervision, order-agnostic autoregressive training, and multi-task learning.

Self-supervised learning in computer vision is a relatively recent paradigm in representation learning where the supervision signal comes from the data and a pretext task. In this class of approaches, a classifier is trained to predict the transformation applied to the input data. Past works introduced rotation prediction (Gidaris et al., 2018), jigsaw puzzle solving (Noroozi & Favaro, 2016), color prediction (Zhang et al., 2016), and context prediction (Doersch et al., 2015; van den Oord et al., 2018) to build up a useful representation for downstream tasks. DistAug can be thought of as learning the inverse generative processes of those different tasks. It is perhaps unsurprising then that we see the same relative rank ordering of pretasks in effectiveness, for example, between rotation and Jigsaw in this work.

Many transformations in our work can be interpreted as some reordering of the raw data, such as rotation and reversal of words. In this case, the task t becomes a permutation operator, which has a rich history in the literature, starting with the very general orderless NADE (Uria et al., 2014), whose objective looks identical to Eq. (3). XLNet (Yang et al., 2019) is a modern reincarnation of this idea that uses the inherently orderless Transformer architecture to learn the density in different orderings. This work also introduced the idea of swapping positional embeddings which we found to be crucial when the model is asked to generate a large number of permutations. Both approaches differ from our method in that they attempt to learn over randomly sampled permutations. We saw that some orderings, however, impart much more useful inductive biases than others. In spirit, this is closer to matching the densities of fixed orderings, for example, of forward and backward models for better sequence generation (Serdyuk et al., 2017), although we propose to train one model for all of them.

Sharing of weights to learn multiple tasks is a common strategy used to better regularize the model. Training on multiple tasks in NLP (Collobert & Weston, 2008), computer vision (Torralba et al., 2007), and across multiple modalities or datasets (Ngiam et al., 2011) is known to provide helpful inductive biases and improve generalization. More similar in spirit to our approach is priming the generator network to control what it generates. For example, (Eslami et al., 2018) shows the model images of a scene and asks it to generate the scene from a different view. (Johnson et al., 2016) trains a neural machine translation model with one decoder that, when conditioned on a certain language token translates to that target language. Although we did not try in this work, we could also use fill-in-the-blank style pretext task like BERT (Devlin et al., 2019) as distributions to augment. This works for any arbitrary data types including text and is related to cutout (Devries & Taylor, 2017) used in RandAugment (Cubuk et al., 2019).

7. Conclusion

We present DistAug, a simple and effective method for regularizing generative models. We show that it improves upon data augmentation by allowing transformations that modify the target density, by scaling to larger model sizes, and by providing useful inductive biases to aid learning. We train autoregressive models that set a new state-of-the-art for density estimation of CIFAR-10 and beat the sample quality of most GANs without explicitly training to do so. In general, the effectiveness of our method depends on the strength of inductive bias encoded by the augmentation function as well as the capacity and expressivity of the model. With less powerful models and in domains like text, where the choice of augmentation is less clear, the gains are modest.

8. Acknowledgements

We would like to thank Jakub Pachocki, Johannes Otterbach, Prafulla Dhariwal, Pranav Shyam, and Nick Ryder for helpful discussions and feedback on early drafts of this work.

References

Chen, X., Mishra, N., Rohaninejad, M., and Abbeel, P. PixelSNAIL: An improved autoregressive generative model. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 864–872, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/chen18h.html.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509, 2019. URL http://arxiv.org/abs/1904.10509.

Collobert, R. and Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, 2008.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Devries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *ArXiv*, abs/1708.04552, 2017.

Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1422–1430, 2015.

Eslami, S. M. A., Jimenez Rezende, D., Besse, F., Viola, F., Morcos, A. S., Garnelo, M., Ruderman, A., Rusu, A. A., Danihelka, I., Gregor, K., Reichert, D. P., Buesing, L., Weber, T., Vinyals, O., Rosenbaum, D., Rabinowitz, N., King, H., Hillier, C., Botvinick, M., Wierstra, D., Kavukcuoglu, K., and Hassabis, D. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018. ISSN 0036-8075. doi: 10.1126/science.aar6170. URL https://science.sciencemag.org/content/360/6394/1204.

Gage, P. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994. ISSN 0898-9788.

Gidaris, S., Singh, P., and Komodakis, N. Unsupervised representation learning by predicting image rotations. *CoRR*, abs/1803.07728, 2018. URL http://arxiv.org/abs/1803.07728.

Gong, X., Chang, S., Jiang, Y., and Wang, Z. Autogan: Neural architecture search for generative adversarial networks. *ArXiv*, abs/1908.03835, 2019.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017. URL http://arxiv.org/abs/1704.00028.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. B. Momentum contrast for unsupervised visual representation learning. *ArXiv*, abs/1911.05722, 2019.

Hénaff, O. J., Razavi, A., Doersch, C., Eslami, S. M. A., and van den Oord, A. Data-efficient image recognition with contrastive predictive coding. *CoRR*, abs/1905.09272, 2019. URL http://arxiv.org/abs/1905.09272.

Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. Flow++: Improving flow-based generative models with variational dequantization and architecture design. *CoRR*, abs/1902.00275, 2019. URL http://arxiv.org/abs/1902.00275.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F. B., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558, 2016. URL http://arxiv.org/abs/1611.04558.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *ArXiv*, abs/1710.10196, 2017.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10215–10224. 2018.

Kolesnikov, A., Zhai, X., and Beyer, L. Revisiting self-supervised visual representation learning. *CoRR*, abs/1901.09005, 2019. URL http://arxiv.org/abs/1901.09005.

Menick, J. and Kalchbrenner, N. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. *CoRR*, abs/1812.01608, 2018. URL http://arxiv.org/abs/1812.01608.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *ArXiv*, abs/1802.05957, 2018.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. Multimodal deep learning. 2011.

Noroozi, M. and Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pp. 69–84. Springer, 2016.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., and Ku, A. Image transformer. *CoRR*, abs/1802.05751, 2018. URL http://arxiv.org/abs/1802.05751.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. 2018.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR*, abs/1701.05517, 2017. URL http://arxiv.org/abs/1701.05517.

Serdyuk, D., Ke, N. R., Sordoni, A., Pal, C., and Bengio, Y. Twin networks: Using the future as a regularizer. *CoRR*, abs/1708.06742, 2017. URL http://arxiv.org/abs/1708.06742.

Theis, L., van den Oord, A., and Bethge, M. A note on the evaluation of generative models. *CoRR*, abs/1511.01844, 2015.

Tian, Y., Peng, X., Zhao, L., Zhang, S., and Metaxas, D. N. Cr-gan: Learning complete representations for multi-view generation. In *IJCAI*, 2018.

Torralba, A., Murphy, K. P., and Freeman, W. T. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):854–869, 2007.

Uria, B., Murray, I., and Larochelle, H. A deep and tractable density estimator. In *International Conference on Machine Learning*, pp. 467–475, 2014.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016a. URL http://arxiv.org/abs/1606.05328.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016b. URL http://arxiv.org/abs/1606.05328.

van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Xie, Q., Dai, Z., Hovy, E. H., Luong, M., and Le, Q. V. Unsupervised data augmentation. *CoRR*, abs/1904.12848, 2019a. URL http://arxiv.org/abs/1904.12848.

Xie, Q., Hovy, E., Luong, M.-T., and Le, Q. V. Self-training with noisy student improves imagenet classification, 2019b. URL http://arxiv.org/abs/1911.04252. cite arxiv:1911.04252.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019. URL http://arxiv.org/abs/1906.08237.

Zhang, R., Isola, P., and Efros, A. A. Colorful image colorization. In *European conference on computer vision*, pp. 649–666. Springer, 2016.

A. Experimental details

A.1. CIFAR-10 (Sparse Transformer)

For CIFAR-10, 58M and 152M models use the same hyperparameters as the ones in (Child et al., 2019) except they are trained with a learning rate of 0.00015 for 1000 – 1500 epochs with a cosine decay (Radford et al., 2018) over 10000 epochs. When the validation loss stops decreasing, typically around 1000 epochs, we further linearly decay the learning rate to 0 over 300 epochs.

The 15M model has one-fourth the layers of the 58M model. Since it was hard to gauge the optimal learning schedule for this model, we anneal the learning rate by half manually based on the validation loss after 2048 epochs. Some permutation runs at this scale show instability before 2048 epochs when various conditioning information is removed. In these cases, learning rate is annealed manually sooner. We stopped the 15M runs at 4000 epochs, even though they continue to improve marginally after this, as the magnitude of increase is typically less than 0.001 BPD.

Batch size for all CIFAR-10 experiments was 16.

A.2. CIFAR-10 (Flow++)

The Flow++ experiments in the paper use the same hyperparameters and codebase as the state-of-the-art model in Ho et al. (2019). All results are reported for 4000 importance weighted samples. We tested with and without augmentations of rotation and transposition, and tested dropout rates of 20% and 1%. Conditioning is applied by concatenating an embedding to the input of each coupling neural network. We share embedding parameters for all coupling networks of equal input resolution.

The baseline model achieves 3.08 bits per dim with 20% dropout and 3.22 bits per dim with 1% dropout. The augmented model achieves 3.03 bits per dim with dropout of 20% and 2.98 bits per dim with a dropout of 1%.

As noted in the main text, however, the embeddings for the augmented model tended to converge to highly similar values and samples from the model look visually similar regardless of which conditioning signal is supplied. This suggests the network did not learn to condition on the augmentation type. Thus we expect that future work incorporating better conditioning techniques for flows will benefit to a greater extent from our technique.

A.3. CIFAR-10 (PixelCNN++)

Our PixelCNN++ experiments used the codebase from Salimans et al. (2017), and our baseline with dropout 0.5 achieved 2.926 bits per dim, closely matching the 2.92 bits per dim reported in the paper. We tested three configurations consisting of the following augmentations: rotation, rotation followed by transposition, and rotation followed by transposition followed by color swapping.

We re-purposed the class conditioning mechanism from PixelCNN++, which learns a class-dependent bias for each convolutional unit, by changing it into an augmentation-dependent bias. When stacking multiple augmentations, we learned a separate bias for each augmentation type, and added the bias vectors.

We trained for 2000 CIFAR-10 epochs, allowing each configuration to converge. Augmenting with rotation improved bits per dim to 2.89 and adding transposition improved bits per dim to 2.88. Adding color swapping resulted in 2.91 bits per dim, but by lowering the dropout rate to 0.05, we achieved our highest performing 2.84 bits per dim.

The importance of conditioning is evident with the PixelCNN++ architecture as well. When we trained the configurations without conditioning, rotation and rotation + transposition both performed about 0.005 bits per dim worse than their conditional counterparts, and rotation + transposition + color swapping performed 0.02 bits per dim worse than its conditional counterpart.

A.4. Jigsaw shuffling and color swapping

For jigsaw shuffling, an image is broken into 2×2 tiles. The tiles are reordered in $(2 \times 2)! = 24$ ways. For color swapping, all $3! = 6$ permutations of the RGB channels are used. A start-of-sequence embedding is learned for each of the 24 and 6 permutations respectively. When we stack different transformations motivated from self-supervision, we use the sum as the start-of-sequence embedding.

A.5. RandAugment parameters

When applying RandAugment (Cubuk et al., 2019) to 32x32 images, we found reducing the cutout and translation constants to 20 and 50 resulted in lower BPD and less corrupted samples than with the original values of 40 and 100.

A.6. FID and Inception Score

FID was calculated with 10,000 samples. Inception score was calculated in 10 batches and using 10,000 samples.

A.7. ImageNet-64 (Sparse Transformer)

We use the same 152M Sparse Transformer model from (Child et al., 2019), and double the depth for the 303M model. Both models are trained with a batch size of 128 with a learning rate of 0.00015 cosine decayed over 10000 epochs. However, ImageNet runs typically converge within 100 to 200 epochs. Because a high learning rate is used throughout training, the baseline likelihood is slightly bet-

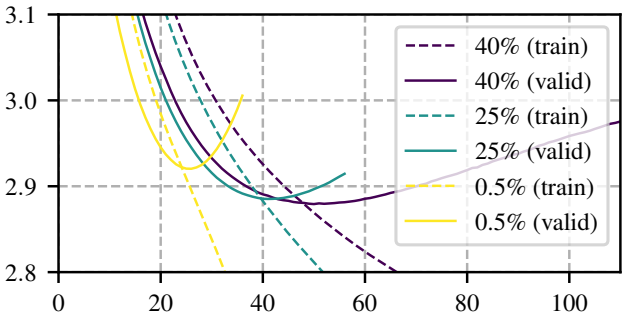
ter than was originally reported in (Child et al., 2019) but exhibits instability around 60 epochs. When training becomes unstable, we first try linearly decaying the learning rate to 0 over 50 epochs. If this still results in divergence, we decrease the learning rate by a constant factor. We found near convergence, the learning rate had to be decreased by a factor of 10-15 when linear decay was not an option.

A.8. Enwik8

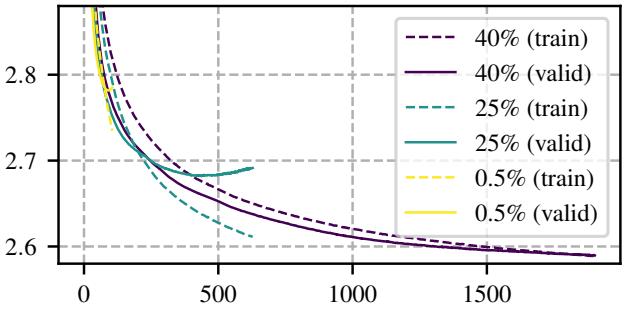
The Enwik8 experiment uses the same hyperparameters as the 95M Sparse Transformer baseline with fixed attention. There are six priming embeddings in total, consisting of two for each decision of whether to apply one of the three reversal augmentations. All options are chosen uniformly at random, so the model is presented with the unaugmented ordering one-eighth of the time. When the model is trained with distribution augmentation, the dropout rate is decreased to 10%. During evaluation, we use the same approach of increasing minimum context length to 12,160 to make our results comparable to (Child et al., 2019). However, it's worth noting that the model gets 0.969 ± 0.001 test bits-per-byte even when evaluated with 6,144 context tokens.

A.9. Dropout tuning

Distribution augmentation is quite effective when paired with properly tuned dropout. In Figure 6, we see that model regularization or DistAug alone is not enough to considerably improve the validation loss. However, DistAug with a well-tuned dropout rate induces the model to train much longer without overfitting. Alternatively, we could increase augmentation strength to use less dropout, but as noted in the main text, we sometimes found that more augmentations impaired sample quality even as they improved likelihood. In either case, we found that choosing an appropriate level of dropout was important for good performance, and suggest that other practitioners tune the dropout level for a given set of augmentations they apply.



(a) Model regularization alone.



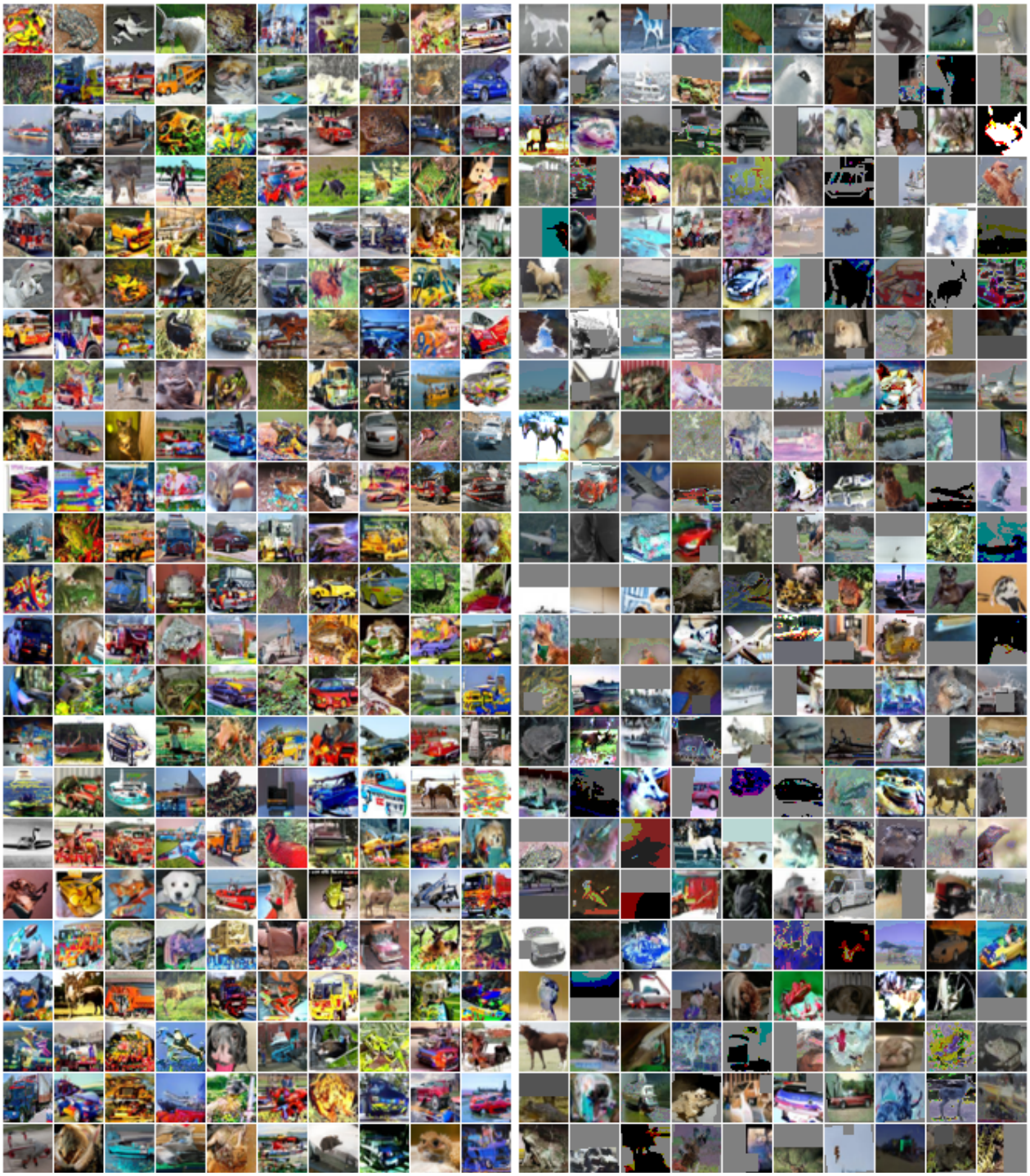
(b) Rotation DistAug.

Figure 6. Tuning dropout is important for distribution augmentation. The x and y axes are epochs and (smoothed) CIFAR-10 negative log-likelihood in BPD, respectively. All experiments are with 152M Sparse Transformer. Here, we see that even 40% dropout with rotation DistAug was not enough to prevent overfitting.

(a) Temperature 1 samples obtain 14.74 FID and 8.18 IS.

(b) Temperature 0.94 samples obtain 10.57 FID and 8.93 IS.

Figure 7. Models trained with heavy distribution augmentation can still generate plausible samples from the original distribution, even though the model has seen unaugmented images a small fraction of the time. We show samples at two different temperatures from the same RandAugment-conditional model trained to 2.66 BPD. When lowering the temperature, we reuse the same initial seeds as those used for temperature 1. While many samples look object-like, there are still some occasional corrupted images.



(a) 2.53 BPD trained with distribution augmentation

(b) 2.72 BPD trained with data augmentation

Figure 8. While the model does assign a higher probability mass on the test examples with augmentation, improved likelihood does not necessarily mean that the model can produce plausible samples (Theis et al., 2015). Samples on the left are drawn from our DistAug model with the best test likelihood of 2.53 BPD. This model is trained with rotation, transposition, color channel swapping, and jigsaw puzzle distribution augmentation. In this setup, the original images are shown once every 1152 times on average. Even with conditioning, we see noticeable artifacts. Samples on the right are from a model trained with regular data augmentation where RandAugment is applied to 99% of the input images. This model obtains 2.72 test BPD compared to 2.80 BPD of Sparse Transformer (Child et al., 2019), but almost all samples are heavily distorted. Finetuning on the original dataset does not improve the sample quality.