Improved Text-Conditioned Image Generation

Justin Wong
Department of Statistics
Stanford University
Stanford, CA 94305
`juswong@stanford.edu`

Abstract

While simpler conditional models are able to create good quality images on simple conditions, extensions to text-embedding conditions require more stable design. Notably, the ACGAN architecture is able to generate good samples, but seems to learn a biased model. An extension to the usage of Wasserstein distance to achieve ACWGAN gives higher sample quality as well as more generalizeable results to different conditioning than those present during training. We can also attempt to further improve the sample quality by extending an importance sampling metric to conditional classes. We find that this helps with generalizing to different conditioning but does not necessarily help with increasing quality of samples as evaluated by quantitative metrics.

1 Introduction

Image generation is a rich field with applications and extensions in various areas. Classically, the generative adversarial network (GAN) solution to this problem has involved using a pair of networks trained adversarially to create images. The generator network synthesizes images after sampling noise from some latent distribution and using it as input. The discriminator attempts to classify if input images are from the true data distribution or generated by the generator. The two have opposing goals so during training, they improve with each other until the generator reaches a satisfactory level of proficiency. However, this approach has limitations as there are no clear way of conditioning the generated data to sample desired features or details. This has been improved by the introduction of conditional models trained on inputs of a latent distribution sample and a conditioning element (i.e. a one-hot label or text embedding) that would allow for more control in the generation process and possibly more structured learned distribution.

The branch of text-conditioned image synthesis has applications in many areas such as computer aided design, result fill-in, and art generation. Improving upon current methods will allow for an increase in quality of these applications as well as possible use cases in other similar methods. However, this task is in general very difficult since it takes a much lower dimensional latent vector and decodes it into a sample image of the desired class. The true distribution of images is of much higher dimensionality and is often much richer than the simple latent distribution used. Then the GAN has the challenging task of finding as best an estimate of this unknown and complex distribution using the comparatively sparse samples space given during training. This is further exacerbated by the need to learn some good conditional distribution for each of the text-embedded conditions. The GAN now cannot just approximate a sub-domain of the true data distribution, but has to demonstrate proficiency in at least the sub-domains specified by the conditions present in training.

The baseline task that purports to solve this problem are auxiliary classifier GANs (ACGANs). This attempts to solve the previous problem by mandating that the samples generated can be classified accurately by the discriminator [1]. However, we will demonstrate that ACGAN may be learning a biased representative distribution and does not generalize well to different conditionings that may be present outside of training. We will study modifications of the ACGAN to improve its performance and generalizability. The main improvement will be through changing the distance metric to use Wasserstein distance to create an AC Wasserstein GAN (ACWGAN) which will have desireable properties. We also study the effects of importance weight re-sampling on creating a new data distribution in the hopes that it can filter out poor samples and leave only good samples as outputs.

2 Related Work

A key challenge in transitioning from image generation to conditional image generation is in ensuring that the samples synthesized are recognizable for each class. This means learning a good representation for every class used as a condition is necessary. ACGAN modifies the original discriminator architecture to add an output that determines the class of the input image. This introduces an additional term in both the discriminator and generator loss function to minimize over - the auxiliary classifier's loss. [1]. For this to make sense, we also pass in information about the image's class to the generator as part of the latent sample. This can be done as a one hot class vector, but we use text embeddings of the class names for this purpose instead.

However, training a GAN is often difficult and unstable. A proposed solution to this is to use Wasserstein distance in the optimization step instead of the vanilla Jenson Shannon divergence [2]. This allows for the usage of a distance metric that more closely relates to the sample quality (correlating to human preferences), the Wasserstein (Earth-Mover) distance. That is, given two distributions, the Wasserstein distance is the lowest cost to convert a given distribution to a different one that is desired. However, the Wasserstein distance is intractable in original form. Fortunately, we can optimize an equivalent problem under the constraint that we use a 1-Lipschitz function. This has shown great effects in increasing the stability of GAN training and sample quality.

Wasserstein GANs have been used in conditional image synthesis before. However, this was in context of a conditional GAN used in image reconstruction [3] which uses a much stronger condition. Also, the generator is given much more image data, so there is less to synthesize. In intermediary work, we also found that the conditional GAN had weaker results than the ACGAN model.

Finally, there are recent results for importance weight re-sampling and their usage in post-hoc sample correction[4]. This looks to reduce bias learned in the generator by finding importance weights relative to each generated image. We can then re-sample to gain a less biased distribution and higher sample quality.

Our main contribution is to build upon prior work to merge WGAN and ACGAN to create a better architecture that current standards. This improves upon the previous ideas since it allows for conditional image synthesis built on unrelated prior distribution and less conditional information than other condition WGAN models. This will also improve upon the baseline ACGAN model and offer future work directions for further modifications that are possible. Finally, we go on to create a natural extension to the importance weight re-sampling methods fitted to conditional class models. Demonstrating that a natural extension applies to conditional models allows for us to fit an importance weight classifier to help in re-sampling to get higher quality images than the baseline.

3 ACGAN & ACWGAN

3.1 Problem Statement

Given some image set data distribution p_{data}, we want to be able to create a function G that models that data distribution. We often choose that the input is some latent space vector z sampled from a trivial distribution such a multivariate unit Gaussian. Let us now assume that the latent vector $z \in \mathbb{R}^{128}$ and the data distribution is CIFAR-10 so real and generated samples $x, G(z) \in \mathbb{R}^{3\times 32\times 32}$.

We want G to learn a joint distribution over that latent variable as opposed to just the conditional distributions. Let us denote $E(\cdot)$ as some embedding function (in this work we will use the BERT encoder) and c is the text condition. Supposing $E(c) \in \mathbb{R}^k$, we can form our input through concatenation $z = z'|E(c), z' \sim \Phi^{128-k}$ and let us say that this process is denoted by $z \sim \Psi$. The generator function is difficult to find in isolation, so we introduce a discriminator that will train adversarially with the generator. In the ACGAN and ACWGAN paradigm, the discriminator takes an image as input and has two outputs. The first output C_1 distinguishes if the sample is real or fake, so it should output high when real and low when fake. The second output C_2 distinguishes the class of the image, so it should be high on the ith entry and low on the others when the image is from class i. Then we want to find

$$C : \mathbb{R}^{3\times32\times32} \to (\mathbb{R}, \mathbb{R}^{10})$$

Decomposing C into its C_1, C_2 and denoting b, y as the target labels for C_1, C_2 and using the error metrics ϵ_1, ϵ_2, we want to get

$$C_1^*, C_2^* = \arg\min_{C_1,C_2} \mathbb{E}_{x\sim p_{data},G(z)}[\epsilon_1(b, C_1(x)) + \lambda\epsilon_2(y, C_2(x))]$$

Returning to the generator, we want to take as input the latent vector z and output an image. Recalling that we want this generation process to be close in some metric to the true data, we denote $K_{C_1}(\cdot,\cdot)$ to be some distance metric parameterized by the discriminator. We then want to find the optimal generator by optimizing

$$G^* = \arg\min_{G} \mathbb{E}_{z\sim\Psi}[K_{C_1}(p_{data}, G(z)) + \lambda_G(\epsilon_2(y, C_2(G(z))))], G : \mathbb{R}^{128} \to \mathbb{R}^{3\times32\times32}$$

Noting that the main difference in the two setups is a change in function choice, we can model the input and output flow of both of these models with the same abstract diagram.

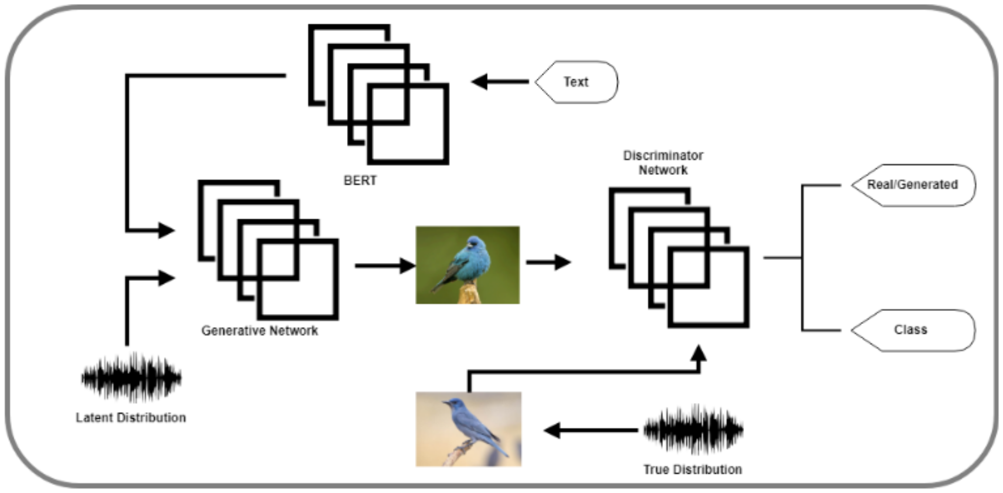

Figure 1: Flow diagram for ACGAN and ACWGAN

3.2 Approach

The ACGAN is formulated using $\epsilon_1(\cdot)$ to be a binary cross entropy loss and $\epsilon_2(\cdot)$ to be a cross entropy loss using a sigmoid and a softmax layer as the final activations to get normalize the outputs to become probabilities. The distance metric is then simply the accuracy of C_1 on the outputs of the generator. This leads to the more familiar ACGAN optimization for the generator as

$$\max_{G} \lambda_G \mathbb{E}_{z\sim\Psi}[log(y|C_2(G(z))) - \mathbb{E}_{z\sim\Psi} log(1 - C_1(G(z)))]$$

and for the discriminator as

$$\max_{D} \mathbb{E}_{x\sim p_{data}} log(C_1(x)) + \mathbb{E}_{z\sim\Psi} log(1 - C_1(G(z))) + \lambda_1 \mathbb{E}_{x\sim p_{data}} log(y|C_2(x)) + \lambda_2 \mathbb{E}_{z\sim\Psi} log(y|C_2(G(z)))$$

In contrast, Wasserstein GAN limits C_1, C_2 to be 1-Lipschitz. Instead of a binary cross entropy loss, we just take $\epsilon_1(x) = x$, but we continue using $\epsilon_2(\cdot)$ as a cross entropy loss. Notably, we do not use since we are no longer restricting the function to output probabilities. The distance metric in this case is the score given by C_1 to the outputs of the generator. Then we get our ACWGAN formulation for the generator as

$$\min_G \lambda_G \mathbb{E}_{z\sim\Psi}[log(y|C_2(G(z))) - \mathbb{E}_{z\sim\Psi} C_1(G(z))]$$

and for the discriminator as

$$\min_D \mathbb{E}_{z\sim\Psi} C_1(G(z)) - \mathbb{E}_{x\sim p_{data}} C_1(x) + \lambda_1 \mathbb{E}_{x\sim p_{data}} log(y|C_2(x)) + \lambda_2 \mathbb{E}_{z\sim\Psi} log(y|C_2(G(z)))$$

We enforce that C_1, C_2 to be 1-Lipschitz by two methods. The first is by weight clipping to limit it to a bounding box where the functions remain 1-Lipschitz. Then there is no modification needed to the loss function. The other method is by adding a regularization term that penalizes the discriminator by being further from 1-Lipschitz. Using the gradient penalty method, we get an optimization on

$$\min_D \mathbb{E}_{z\sim\Psi} C_1(G(z)) - \mathbb{E}_{x\sim p_{data}} C_1(x) + \lambda_1 \mathbb{E}_{x\sim p_{data}} log(y|C_2(x)) + \lambda_2 \mathbb{E}_{z\sim\Psi} log(y|C_2(G(z)))$$
$$+\lambda \mathbb{E}_{\hat{x}\sim \alpha x + (1-\alpha)G(z)}(||\nabla C(\hat{x})||_2 - 1)^2$$

which we can recognize as the classical Wasserstein GAN-GP loss functions plus regularization terms on using the auxiliary classifier. This is useful since it allows us to learn more complex distributions as compared to the former method [5].

We choose to use a larger model architecture inspired by that used by the gradient penalty WGAN method to ensure that all the models have enough parameters to learn. In the ACGAN model, we allow for the usage of batch normalization as well as sigmoid and softmax activations. We have simply given it a larger parameter space to train over by changing from a shallow convolutional network to a deep residual network. We remove the sigmoid and softmax activation when training the ACWGAN and remove batch normalization when training the gradient penalty version. This makes it easier to preserve the 1-Lipschitz property and gives better training results. This also ensures that the only difference between the performances of the two models is the loss functions and has nothing to do with the architecture. In this respect, we are isolating the effect of changes in results to the nature of the models so as not to confound other variables.

We have a few hyperparameters to optimize over such as the gradient penalty coefficient in the discriminator (for ACWGAN) and the auxiliary classifier coefficients present in the generator and the discriminator for both ACGAN and ACWGAN. We perform a small parameter sweep of [0, 1, 10] and find that the general gradient penalty coefficient of 10 works well in this use case as well. We also find that the auxiliary classifier coefficients are best at either 0 or 1. In terms of the loss functions written above, we use $\lambda_G = 1, \lambda_1 = 1, \lambda_2 = 0$, so the training is best when the auxiliary classifier error is restricted to either the error or the true data distribution or the generated distribution, depending on the network. We did not find a significant change when experimenting with the learning rate, so it is left to 10^{-4}.

4 Importance Weighting

4.1 Problem Statement

In addition to creating the ACWGAN model, we also want to create a method to do importance sampling of the outputs that are created by the generator. Specifically, this should take an image and output the probability of it being from the real data distribution versus data synthesized by the generator. Moreover, we want a classifier that learns a joint distribution of the class label condition and the image itself. We model this by introducing a third neural network we will call the classifier w that takes inputs of the image and the class label and outputs the probability of it being an image samples from the true data distribution or from the generated distribution.

$$W : (\mathbb{R}^{3\times32\times32}, \mathbb{R}^{10}) \to \mathbb{R}$$

. Given input image x and one hot class label vector c, we want to optimize this classifier over some error function ϵ

$$W^* = \arg\min_W \mathbb{E}\epsilon(W(x, c))$$

We will use this classifier to compute the importance weights for samples synthesized by the generator.

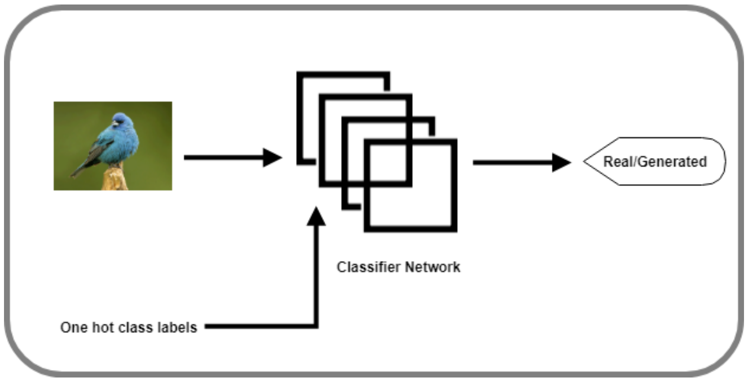

Figure 2: Flow diagram for classifier network

4.2 Approach

This is most naturally optimized using a binary cross entropy loss since it is a binary classifier.

$$W^* = \arg\max_w : \mathbb{E}_{x\sim p_{data}} log(W(x, c)) + \mathbb{E}_{z\sim\Psi} log(1 - W(G(z), c))$$

We want to construct the importance weights from the classifier outputs such that it still maintains the property of being an unbiased estimator. Let $p_\theta(x)$ be the distribution of the generator. Then we can rewrite the expectation to become

$$\begin{aligned}
\mathbb{E}_{p_{data}(x,c)} G(z) &\approx \frac{1}{N}\sum_{i=1}^{N} p_{data}(G(z_i), c) G(z_i) \\
&= \frac{1}{N}\sum_{i=1}^{N} \frac{p_\theta(G(z_i), c)}{p_\theta(G(z_i), c)} p_{data}(G(z_i), c) G(z_i) \\
&= \frac{1}{N}\sum_{i=1}^{N} \frac{p_{data}(G(z_i), c)}{p_\theta(G(z_i), c)} p_\theta(G(z_i), c) G(z_i) \\
&\approx \mathbb{E}_{p_\theta(x,c)} w(G(z), c) G(z)
\end{aligned}$$

where $z_i \sim \Psi$, c is a class labels, and the approximations are made by Monte Carlo estimation. We then find that the importance weights can be constructed from the classifier outputs using $w(G(z), c) = \frac{W(G(z),c)}{1-W(G(z),c)}$ and still be unbiased.

We also want this classifier to have certain properties to see that we are improving the baseline distribution with importance sampling re-weighting. Specifically, let $p_{\theta,w}(x, c) \propto p_\theta(x, c) w(x, c)$ by the unnormalized distribution formed by multiplying importance weights to the original distribution. Let $Z_{\theta,w}(x, c) = \mathbb{E}_{p_\theta} w(x, c)$ be the normalized probability distribution. We want that our new distribution is "closer" to the data distribution that the original one. Choosing the KL divergence to be our metric, we desire $KL(p_{data}, p_{\theta,w}) \leq KL(p_{data}, p_\theta)$. In this computation, we assume that the distribution of the classes are equivalent across the generated distribution and the data distribution to get $C = E_{p_{data}} w(\cdot, c) = E_{p_\theta} w(\cdot, c)$. This makes sense because we are enforcing that the generator learn a joint distribution of images and class labels that mimics the data distribution. Then equivalently

$$
\begin{aligned}
& KL(p_{data}(x,c), P_{\theta,w}(x,c)) \leq KL(p_{data}(x,c), p_\theta(x,c)) \\
\iff & E_{p_{data}} log(\frac{p_{data}(x,c)}{p_{\theta,w}(x,c)}) \leq E_{p_{data}} log(\frac{p_{data}(x,c)}{p_\theta(x,c)}) \\
\iff & E_{p_{data}}[log(p_{data}(x,c)) - log(w(x,c)) - log(p_\theta(x,c)) + log(Z_{\theta,w}(x,c))] \\
& \leq E_{p_{data}(x,c)}[log(p_{data}(x,c)) - log(p_\theta(x,c))] \\
\iff & log(Z_{\theta,w}(x,c)) \leq E_{p_{data}} log(w(x,c)) = E_{p_{data}} log(w(x|c)w(\cdot,c)) = E_{p_{data}} log(w(x|c) + C
\end{aligned}
$$

However, computing the joint distributions of either $Z_{\theta,w}(x,c), w(x,c)$ is intractable. The best we can do it to compute the conditional probability $w(x|c)$ using a Monte Carlo estimation. Since we are able to decompose the previous terms relating to the conditional probability, we have necessary conditions that we can check though we do not have sufficient conditions to demonstrate that we will definitely improve the distribution. Specifically, we find that we require at least two conditions. First, we need the expectation of the log probabilities to fulfill the correct ordering.

$$
\begin{aligned}
E_{p_{data}} log(w(x|c)) &\geq log(Z_{\theta,w}(x,c)) - C \\
&= log(E_{p_\theta} w(x,c)) - C \\
&\geq E_{p_\theta} log(w(x,c)) - C = E_{p_\theta} log(w(x|c))
\end{aligned}
$$

Second, we need that the log of the expected probabilities achieve the correct ordering.

$$
\begin{aligned}
log(E_{p_{data}} w(x|c)) &\geq E_{p_{data}} log(w(x|c)) \\
&\geq log(Z_{\theta,w}(x,c)) - C \\
&= log(E_{p_\theta} w(x,c)) - C = log(E_{p_\theta} w(x|c))
\end{aligned}
$$

However, this is not sufficient and can still fail under certain circumstances[4].

Shifting to class conditioned weights is important since we want to be able to perform importance weighted re-sampling on a conditional basis. Specifically, we want to be able to identify good samples with respect to condition as opposed to good samples overall in order to maintain the robustness of the conditional image generation scheme.

This allows us to shift from a naive sampling method to a new re-sampling scheme which may return better samples over qualitative and quantitative metrics.

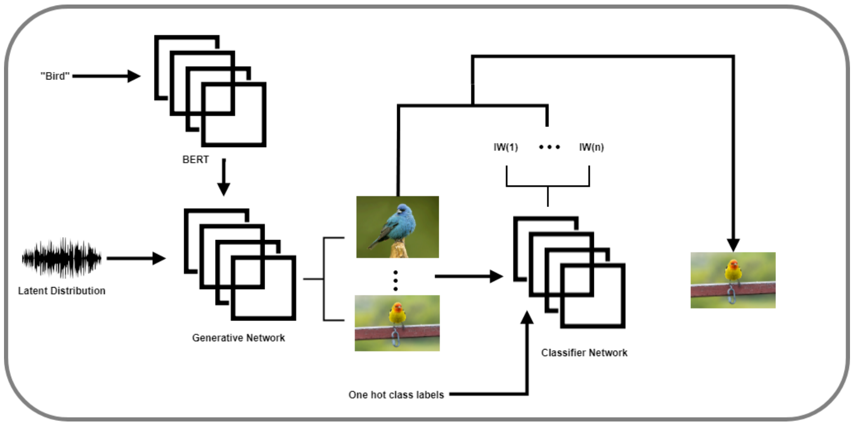

Figure 3: Importance weight re-sampling flow diagram

We choose to use a similar residual network as the classifier with modified architecture to allow for an additional input to be fully connected to the final activation layer. This is because we hope that the discriminator model is able to retrieve much of the class information from the image since a class output was originally one of the desired outputs. We should only need a small effect from the conditional class input plus the final flattened layer to achieve a good classifier.

We first attempt to use importance weights to re-weight the samples generated during training in order to optimize over a weighted loss function. The idea behind this is to train the generator over influential samples such as the worst quality samples as opposed to assuming all samples are equal. This is a technique that has shown improved results in variational autoencoders, but in the implementation of ACWGAN, the importance weights led to unstable behavior and a divergence of the Wasserstein distance[1]. This defeats the purpose of the WGAN so we shift to only using importance weights in post-hoc re-sampling of the generator output.

5 Results

5.1 Experimental Procedure

The models ACGAN and ACWGAN[2] are both trained on the same data - the CIFAR-10 data in the training split. After training for a pre-specified amount of time (a little more than a day), they are stopped and evaluated for performance. As long as the model is able to produce recognizable results on the training text embedding conditions, we accept the parameters. We run another test and train with another seed to gain confidence that the model performance is not related to a lucky seed. However, we will use the first model in evaluation of metric scores.

The classifier model used in importance weighted re-sampling is trained alternatively on the generated images from an accepted ACWGAN model and real images sampled from the CIFAR-10 dataset. We check that the desired inequalities (proven above) hold on the training CIFAR-10 data versus the generated images as well as on the testing CIFAR-10 data versus the generated images. If both of these pass, we accept the model to use in the computation of importance weight re-sampling (IWR) metric scores.

The quantitative metrics are computed by sampling 10,000 samples from each of the models. First they are conditioned on the same text embeddings as in the training procedure. ACGAN and ACWGAN produce 1,000 samples per class for evaluation using random noise concatenated with the text embeddings as inputs to create dataset A1 and W1. ACWGAN then produces 10 samples for a given class. We compute the importance weights using the classifier outputs on those images and sample one image to add to dataset I1. This continues until I1 has 10,000 samples. Next, they are conditioned on coarse text embeddings which are just more generalized categories of the original text embeddings. For reference, we have that the fine texts are ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"] and the coarse texts are ["plane", "car", "bird", "pet", "animal", "pet", "amphibian", "equine", "boat", "vehicle"]. The sampling procedure repeats using these new embeddings to create datasets A2, W2, I2. We then initialize Inception network to compute the Inception Score (IS) and the Fréchet Inception Distance (FID) as compared to 10,000 CIFAR-10 samples.

5.2 Data

IS and FID are human-correlated metrics that give a quantitative score to image quality. Utilizing factors such as image sharpness and diversity, IS computes a score computing a scores on sharpness and diversity. The sharpness score is determined by how precisely a classifier network (in this case Inception) can label the image. Then it computes a negative entropy term since we desire $p(y|x)$ to be low entropy. The diversity term is computed by looking at the distribution of classes generated

[1]To accomplish this, we retrain the classifier every iteration for some number of epochs. However, when implemented from the beginning of training, the classifier is far from optimal and causes the loss to diverge. When implemented on a trained ACWGAN model, the loss does not improve whereas the ACWGAN model continues to slowly improve.

[2]ACWGAN was trained with both procedures - the original weight clipping method and gradient penalty. However, the clipping method has been shown to lack the ability to learn more complex distributions [5]. In the training plots in Appendix A, we see that the weight clipping is much less stable. We also notice that the images qualitatively are not good enough, so we restrict analysis to gradient penalty method to avoid wasting compute resources.

and taking the KL divergence between it and a uniform density. We compute a positive entropy term since we desire $p(y)$ to be high entropy. These two scores are multiplied to get the IS so a higher IS is akin to higher sampling quality.

FID computes a metric by looking at network layer weights when images from different distributions are passed though. Specifically, the means and variances of the layer nodes are computed when evaluating the different datasets and fitted to a Gaussian distribution $N_{data}(\mu_1, \Sigma_1), N_\theta(\mu_2, \Sigma_2)$. The layer activation distributions are then compared using the Fréchet distance $FID = ||\mu_1 - \mu_2||^2 + Tr(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^{1/2})$ get an FID score. A lower difference of the activation distributions means that the distributions are more similar and so the samples should be similar in quality as well. Since we are comparing each dataset to CIFAR-10 samples (real data) a lower FID score is indicative of higher sample distribution quality.

Image Generator	**IS**	**FID**
ACGAN fine conditioning	5.88	81.13
ACGAN coarse conditioning	3.98	140.41
ACWGAN-GP fine conditioning	5.89	49.10
ACWGAN-GP coarse conditioning	4.85	87.11
ACWGAN-GP-IWR fine conditioning	4.78	70.52
ACWGAN-GP-IWR coarse conditioning	4.33	95.64
CIFAR-10 images	8.97	-

Numerically, we can see that ACGAN is the least generalizable model and has the worst performance on coarse text conditioning. Our new model ACWGAN has the best sample quality for fine text conditions and coarse text conditions and our extension of conditional importance weight re-sampling achieves the highest generalizability with the smallest change in scores when changing from fine text conditions to coarse text conditions. The IS suggests that there is a difference between ACWGAN and the true CIFAR-10 data. Other non conditional models may achieve at least $IS > 8$. However, this is a limitation of the compute and time provided and if ACWGAN was trained for as long as 100,000 iterations, we would expect to see a similar IS. Moreover $FID < 40$ is would be good and $FID < 20$ is very good. This is the baseline that other non-conditional models look for and ACWGAN is already very close without nearly the same amount of training time. Again, this score should greatly improve by extending the training time by some ten times to 100,000 iterations.

We now take a look at samples generated by each of the models. We will examine 100 samples from each model, 10 samples per text condition and see if the qualitative observations match with the quantitative score assigned to it.

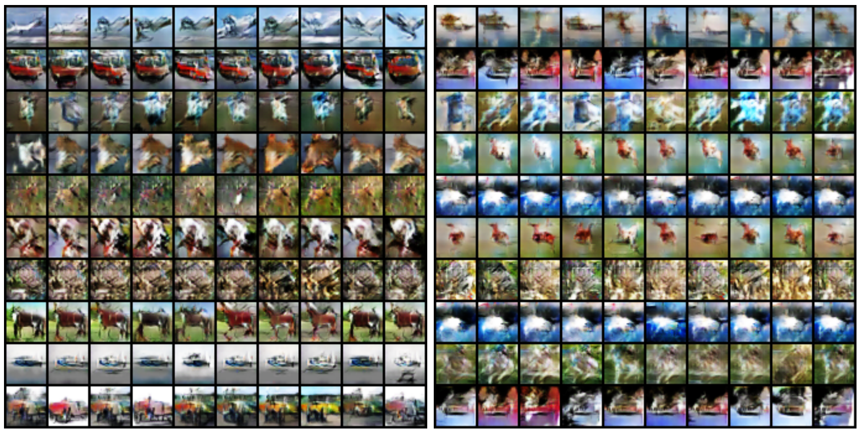

Figure 4: Left: One row per class sampled from ACGAN with fine text embeddings. Right: One row per class sampled from ACGAN with coarse text embeddings.

As can be seen, ACGAN samples on the fine embeddings are high quality and sharpness, but they look near mode collapsed as the images have lots of similarity with each other. We consider that in the loss specified for ACGAN, the generator is punished when the discriminator has difficulty distinguishing the class of the generated sample. Then the generator is possibly biased towards generating the most "class-like" image for each class with some perturbations. This effects is clear once we shift to coarse labels and see that the images generated make no sense. While they are still sharp, they do not capture the classes at all and demonstrate that ACGAN has failed to generalize. They have in fact overfit to the most "class-like" images of each class and have learned a crutch for estimating the conditional probabilities. However, ACGAN seems to have no real notion of the underlying joint distribution. It seems to have high IS due to the fact that it is not mode collapsed, but just centered around a particular sample with small variation. These samples are qualitatively the worst out of the rest that we will examine.

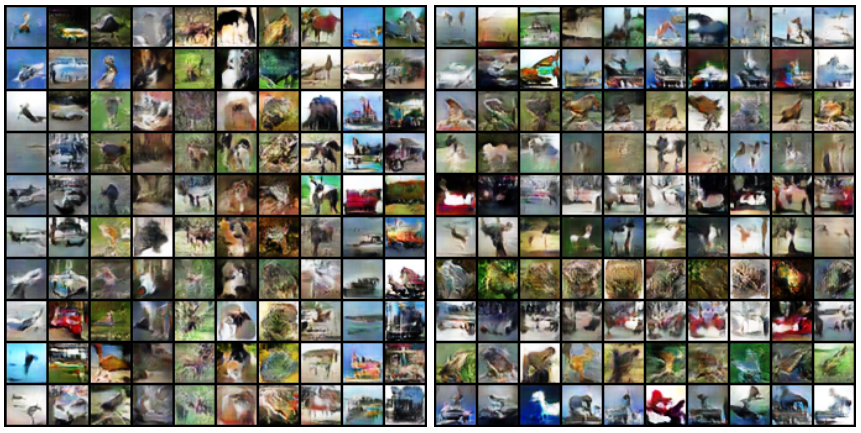

Figure 5: Left: One column per class sampled from ACWGAN with fine text embeddings. Right: One column per class sampled from ACWGAN with coarse text embeddings.

ACWGAN samples have high quality and sharpness and there is a lot of image diversity within the classes. This is reflected in a lower FID and a higher IS. Noting that the change in IS is much less than the change in FID, we suggest that the sharpness of samples is actually less than those of ACGAN. However, WGANs are often trained for some hundreds of thousands of iterations, which is ten times more than the training time that our compute allowed for. We expect that the score discrepancy between ACGAN and ACWGAN will deviate more as training time increases. Moreover, we expect that the sharpness will increase without loss of sample diversity. Here, the IS and FID seem to underestimate the high quality of images sampled.

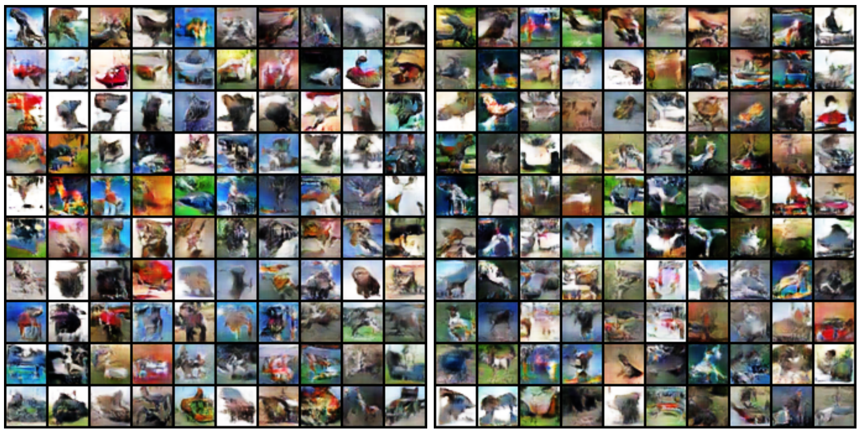

Figure 6: Left: One column per class sampled from ACWGAN with importance sample and fine text embeddings. Right: One column per class sampled from ACWGAN with importance sample and coarse text embeddings.

The samples generated through importance weight re-sampling of ACWGAN outputs retain image diversity, but have lost some level of sharpness. This may be because the sharpest samples were more easily distinguishable as generated as compared to fuzzier samples. As a result, we notice that the importance weight re-sampling unfortunately decreased the IS and increased the FID. Even though the classifier fulfilled the necessary conditions, they were not sufficient to guarantee that the new distribution is better than the original. However, we do notice that usage of this new method improves generalizability from fine text embeddings to coarse text embeddings. That is, the difference of IS and FID is less than the other models evaluated. In effect, the re-sampling has still managed to reduced bias present in the generation process.

We believe that importance weight re-sampling does indeed work for this problem, but there are other issues hindering the IS and FID score evaluated. First, the Wasserstein GAN was trained for only 10,000 iterations. However, in the literature, WGANs are often trained for some 100,000 iterations as they tend to continue to improve over time. The training graph of the gradient penalty ACWGAN seems to imply a slow improvement as well [3]. Obtaining a better ACWGAN to train the classifier on would allow for better importance weights to be estimated. Furthermore, assumptions made when creating the classifier architecture may prevent it from being an optimal classifier. Though it satisfies necessary conditions, it says nothing about its optimality and so the importance weights may again be inaccurate. Both of these point to the method being sound, but the evaluation having possible flaws that can be corrected with more compute (either on training the ACWGAN for a longer period of time or for examining more classifier models of possibly higher parameter count).

6 Conclusions

In this work, we were able to utilize and improve upon the existing ACGAN and WGAN to create an ACWGAN model that outperformed the baseline metrics. Since this type of model is often trained for much longer, we expect that the results can be improved simply by increasing the amount of compute used to get even better results. Moreover, we extended the concept of importance weighting re-sampling to introduce the idea of conditional re-sampling by using an additional classifier network added onto ACWGAN. While this did not improve the sample quality (though this may also be an effect of the ACWGAN not being trained to its full potential or the classifier not being optimal), it still managed to demonstrate a reduction in the learned bias of ACWGAN by making its results more generalizable to different conditioning than seen in training.

While the results could have been better given more time and more compute, we still achieved the goals of improving the baseline, implementing ACWGAN, and implementing conditional importance weights to ACWGAN. While there were more ideas that could have been accomplished, this work is

[3]The training graph only goes up to a certain amount since we use Google Colaboratory to do model training. The runtime is reset after some time so the local variables are lost and we are unable to get the full training plot.

already non-trivial and the results are very good, but we have many ideas of how to improve upon them. Future work may entail modifying ACWGAN such that it is able to improve and train with importance weighting. This would be a great benefit to improving the samples and will allow for a better performance of post-hoc re-sampling.

Acknowledgments

We would like to thank Professor Ermon, Professor Grover, and the TA team for their guidance, insight, and support.

References

[1] C. Shlens J. Odena, A. Olah. Conditional image synthesis with auxiliary classifier gans. *Proceedings of Machine Learning Research 70*, 2017.

[2] S. Bottou L. Arjovsky, M. Chintala. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[3] S. Mirza, M. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1610.09585*, 2014.

[4] J. Agarwal A. Tran K. Kapoor A. Horvitz E. Ermon S. Grover, A. Song. Bias correction of learned generative models using likelihood-free importance weighting. *Neurips*, 2019.

[5] F. Arjovsky M. Dumoulin V. Courville A. Gulrajani, I. Ahmed. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

Appendix A Training Plots

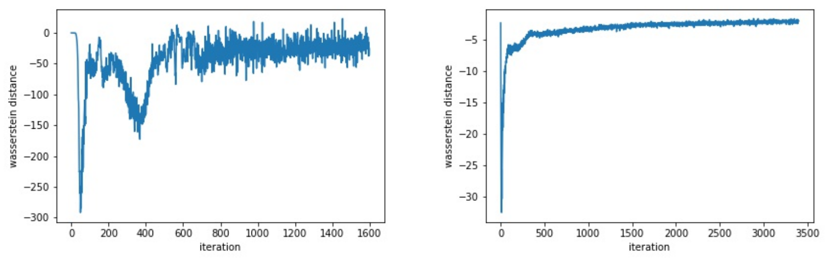

Figure 7: Left: Negative Wasserstein distance during training of ACWGAN implemented with clipping. Right: Negative Wasserstein distance during training of ACWGAN implemented with gradient penalty.

Appendix B Code

https://github.com/JustinWongStanford/ACWGAN/tree/master