An Application of the Auxiliary Classifier GAN on the CIFAR-10 Dataset

Eric Tay
ECE Department
Duke University
et102@duke.edu

Abstract

In this paper, we implement the Auxiliary Classifier Generative Adversarial Network (AC-GAN) [1] and train our AC-GAN on the CIFAR-10 Dataset. In our project proposal, we set out to generate a few samples from each class, calculate an "ResNet Score" (RS) [2] for the model, and perform latent space interpolations between samples. In this work, we extend these requirements, training 3 additional models, additionally calculating the "ResNet Accuracy" (RA) for the images each model generates, and plotting other visualizations found in Figure 9 of [1]. Ultimately, we propose several directions for future work in order to generate more realistic images when training AC-GANs on the CIFAR-10 Dataset. Code to reproduce the results in this paper can be found at our GitHub repository.

1 Background

Generative adversarial networks (GANs) offer a promising approach for training an image synthesis model. While GANs can produce convincing image samples on datasets with low variability and low resolution, GANs struggle to generate globally coherent, high resolution samples, particularly from datasets with high variability [1].

To combat this, the basic GAN framework can be augmented using side information. One strategy is to supply both the generator and discriminator with class labels in order to produce class conditional samples [3]. While the AC-GAN also supplies the generator with class labels, it instead tasks the discriminator with reconstructing the class label of input images, on top of discriminating between real and fake images [1].

Ultimately, this new algorithm is useful in stabilizing training, and in generating high quality and high resolution images with significant diversity [1].

2 Method

We now rigorously define the method, as it pertains to our implementation (CIFAR-10). In the AC-GAN, every generated sample has a corresponding class label $c \in \mathbb{N}, 1 \leq c \leq 10$, in addition to a noise vector $z \in \mathbb{R}^{100}$. The generator $G : \mathbb{R}^{110} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^{32} \times \mathbb{R}^{32} \times \mathbb{R}^3$ uses both to generate images $X_{fake} = G(c, z)$. The discriminator $D : \mathbb{R}^{32} \times \mathbb{R}^{32} \times \mathbb{R}^3 \rightarrow \mathbb{R}^{11}$, then gives both a probability distribution over sources and a probability distribution over the class labels, $P(S|X), P(C|X) = D(X), P(S|X) \in \mathbb{R}, P(C|X) \in \mathbb{R}^{10}$. The objective function then has two parts: the log-likelihood of the correct source, L_S, and the log-likelihood of the correct class, L_C.

$$L_S = E[\log P(S = real|X_{real})] + E[\log P(S = fake|X_{fake})] \tag{1}$$

ECE 590-07 (2021 Spring) final report.

$$L_C = E[\log P(C = c|X_{real})] + E[\log P(C = c|X_{fake})] \quad (2)$$

D is trained to maximize $L_S + L_C$ while G is trained to maximize $L_C - L_S$.

2.1 ResNet

In Section 3, we evaluate the generated images quantitatively with a ResNet-20 model, which was trained using the architecture and training code generated in Homework 3 of the class. We trained this model on the CIFAR-10 train data for 100 epochs, with a batch size of 128, initial learning rate of 0.1 and regularization value of $1e-4$ (L2 penalty applied on the SGD optimizer), finally saving the model with the highest test accuracy as our ResNet.

To obtain an RS, we use the ResNet to make predictions on generated images to get the conditional label distributions, $p(C|X)$. Meaningful images should have a conditional label distribution $p(C|X)$ with low entropy. Moreover, we expect the model to generate varied images, so the marginal $\int\int p(C|X = G(c, z))dcdz$ should have high entropy. Combining these two requirements, the RS is defined as $\exp(E_X(\mathbb{KL}(p(C|X)||p(C))))$ [2], which we compute according to code the authors of [2] provide at [4].

More directly, the RA is simply the classification accuracy of our RN on the generated images. This is a more straightforward metric that is instead used in [1].

2.2 Model 1

Table 1: Hyperparameters and architecture used for model 1 and 3

Operation	Kernel	Strides	Padding	Feature maps	BN?	Dropout	Nonlinearity
$G_x^{(1)}(z)$							
Linear	N/A	N/A	N/A	384	×	0.0	ReLU
Transposed Convolution	6×6	2×2	0	192	✓	0.0	ReLU
Transposed Convolution	5×5	2×2	0	96	✓	0.0	ReLU
Transposed Convolution	4×4	2×2	0	3	×	0.0	Tanh
$G_x^{(3)}(z)$							
Linear	N/A	N/A	N/A	768	×	0.0	ReLU
Transposed Convolution	5×5	2×2	0	384	✓	0.0	ReLU
Transposed Convolution	5×5	2×2	0	192	✓	0.0	ReLU
Transposed Convolution	4×4	2×2	0	96	✓	0.0	ReLU
Transposed Convolution	4×4	1×1	0	48	✓	0.0	ReLU
Transposed Convolution	4×4	2×2	0	3	×	0.0	Tanh
$D_x^{(1)}(z)/D_x^{(3)}(z)$							
Convolution	3×3	2×2	1	16	×	0.5	Leaky ReLU
Convolution	3×3	1×1	1	32	✓	0.5	Leaky ReLU
Convolution	3×3	2×2	1	64	✓	0.5	Leaky ReLU
Convolution	3×3	1×1	1	128	✓	0.5	Leaky ReLU
Convolution	3×3	2×2	1	256	✓	0.5	Leaky ReLU
Convolution	3×3	1×1	1	512	✓	0.5	Leaky ReLU
Linear	N/A	N/A	0	11	×	0.0	Soft-Sigmoid
Generator Optimizer	Adam ($\alpha = 0.0002, \beta_1 = 0.5, \beta_2 = 0.999$)						
Discriminator Optimizer	Adam ($\alpha = 0.0002, \beta_1 = 0.5, \beta_2 = 0.999$)						
Batch size	100						
Epochs	100						
Leaky ReLU slope	0.2						
Activation noise standard deviation	0.1						
Weight initialization	Isotropic Gaussian (μ = 0, σ = 0.02) for Linear/Conv layers, Constant(1) for BN layers						
Bias initialization	Constant(0)						

A Soft-Sigmoid refers to an operation over 11 output units where we apply a Softmax activation to 10 of the units and a Sigmoid activation to the remaining unit. “Activation noise standard deviation" applies to the noise added to the output of each layer of the discriminators, as done in [1] and [2].

For model 1, we seeked to reproduce the AC-GAN used for CIFAR-10 data in [1] with high fidelity. To do so, we employ the hyperparameters and architecture for $G^{(1)}$ and $D^{(1)}$ in Table 1, which are largely the same as those in Table 2 of [1], with the following differences:

- Instead of kernel sizes of 5, we employ kernel sizes of 6, 5, and 4. The former setting would lead to output images of size $29 \times 29 \times 3$, and it is unclear how [1] employs padding.
- We add a padding of 1 to all convolution layers of $D^{(1)}$, since padding was not specified for [1], and doing so is necessary to prevent images from being too small as they pass through $D^{(1)}$.
- Instead of using a grid of 3 values for the learning rates (α) of $G^{(1)}$ and $D^{(1)}$, and activation noise standard deviation, we fix these values to be the middle value of the proposed grid in [1] due to computational reasons. The other values were also tried but these values seemed to generate the best images qualitatively.
- We initialize the weights for BN layers to be 1, since this was unspecified in [1].

The pseudocode in Algorithm 1 describes the training procedure we employed.

Algorithm 1

1: **procedure** AC-GAN TRAINING
2: Load CIFAR-10 train data
3: Subtract 0.5 from the pixel values in each channel, and divide by 0.5.
4: **for** epoch i in $1, 2, \ldots, 100$ **do**
5: **for** batch index j in $1, 2, \ldots, 500$ **do**
6: Zero the gradient for Optimizer$_D$
7: Loss$_{\text{real data}} = -(E[\log P(S = real|X_{real})] + 2 * E[\log P(C = c|X_{real})])/2$
8: Loss$_{\text{fake data}} = -(E[\log P(S = fake|X_{fake})] + E[\log P(C = c|X_{fake})])/2$
9: Loss$_D$ = Loss$_{\text{real data}}$ + Loss$_{\text{fake data}}$
10: Compute gradients and update weights for D
11: Zero the gradient for Optimizer$_G$
12: Loss$_G$ = $E[\log P(S = fake|X_{fake})] - E[\log P(C = c|X_{fake})]$
13: Compute gradients and update weights for G

With reference to Algorithm 1, we note that the following discretionary decisions were made during training:

- Line 3: Images were normalized to have pixel values between -1 and 1. This was because the provided generator architecture in Table 2 of [1] had a final $tanh$ activation layer.
- Lines 7 and 8: Loss was divided by 2 for D. This was to ensure that G and D were approximately updated by the same amount, such that no one would overpower the other.
- Line 7: Class loss for D ($-E[\log P(C = c|X_{real})]/2$) was multiplied by 2. This was because $D^{(1)}$ faced significant difficulties in accurately determining the class of real images. Even after this multiplication, after 100 epochs, the final classification accuracy for class in Model 1 was 57.95%.

Upon completing the model training, we generated sample images, and computed the RS and RA on these images. Both qualitatively and quantitatively, we found that the images were not very desirable, and therefore explored with different AC-GAN models.

2.3 Model 2

For Model 2, we trained the completed Model 1 for 60 more epochs, this time using a learning rate of 0.0001 for both $G^{(1)}$ and $D^{(1)}$. We also multiply the class loss by 10, instead of 2, to enable $D^{(1)}$ to better discern the class of real images. We note that doing so significantly aided training, bringing the discriminator class accuracy up to a peak of 65.03%. Interestingly, the generator source accuracy (percentage of images that the fake images fool the discriminator) also increases from 30.49% to a peak of 44.40%. At the end of each epoch, we additionally generate 10,000 images and calculate the RS for these images with our ResNet, eventually storing the model with the highest RS as Model 2.

2.4 Model 3

For Model 3, inspired by the ability of AC-GANs to generate high-quality, high-resolution images [1], we explored using a higher capacity generator and discriminator, $G^{(3)}$ and $D^{(3)}$, whose architectures are given in Table 1. In this case, $G^{(3)}$ would produce $\mathbb{R}^{64} \times \mathbb{R}^{64} \times \mathbb{R}^3$ images as output, which $D^{(3)}$ subsequently takes as input. We train the model with the same parameters as Model 1 (Table 1), with the extra step of upsampling the real CIFAR-10 images with the nearest neighbor algorithm to make them compatible with $D^{(3)}$.

We note that for Model 3, the larger discriminator is, not surprisingly, better able to classify real images, with a final class accuracy of 67.38% by the end of 100 epochs.

2.5 Model 4

As done in Model 2, we trained the completed Model 3 for 20 more epochs (60 was not chosen due to the high computational cost of training the higher capacity $G^{(3)}$ and $D^{(3)}$), this time using a learning rate of 0.0001 for both $G^{(3)}$ and $D^{(3)}$. We also multiply the class loss by 10, instead of 2, to enable $D^{(3)}$ to better discern the class of real images. We note that doing so again significantly aided training, bringing the discriminator class accuracy up to a peak of 73.29%. At the end of each epoch, we additionally generate 10,000 images. We then downsample each image with bilinear interpolation and calculate the RS for these images with our ResNet, eventually storing the model with the highest RS as Model 4.

3 Experiment results

3.1 Qualitative Evaluation of Images

For each of the 10 CIFAR-10 classes, we generate 16 images, 4 from each model, and plot the images in Figure 1.

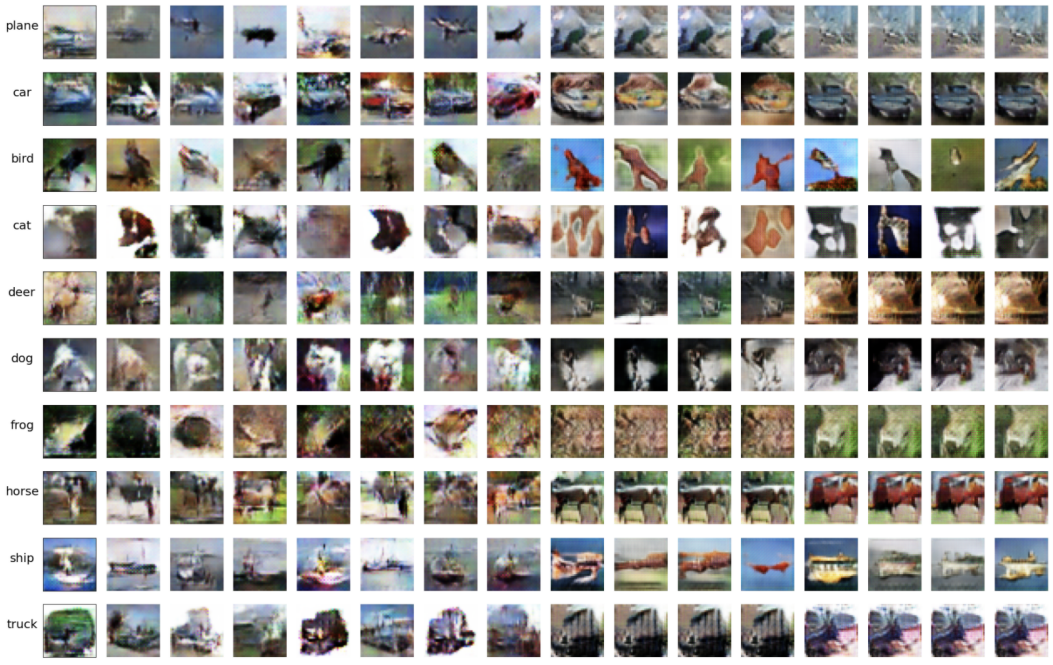

Figure 1: Images generated from all 4 models, across all 10 CIFAR-10 classes. For each row, the same 4 110-dimensional vectors are used to create all 16 images, for better comparison. From left to right, the first 4 images are generated by Model 1, then Model 2, then Model 3, and finally Model 4.

We note that in general, finetuning our models (going from Model 1 to 2, and from 3 to 4) led to higher quality images qualitatively. In addition, it seems like the images generated from the higher-capacity models (Models 3 and 4) are more realistic than those generated from Models 1 and 2. However, it should be noted that Models 3 and 4 are more susceptible to mode collapse within a single class.

3.2 Quantitative Evaluation of Images

To quantitatively evaluate our models and their generated images, we generate 25,000 images for each model (as is done in [1]), and calculate the RS and RA for these images. We summarize our results in Table 2.

Table 2: RS and RA values across different images

Images	ResNet Score	ResNet Accuracy
Model 1	4.12 ± 0.02	46.27%
Model 2	4.99 ± 0.04	63.52%
Model 3	4.98 ± 0.05	59.70%
Model 4	7.11 ± 0.04	90.82%
CIFAR-10 Test Data	8.49 ± 0.10	92.06%
CIFAR-10 Train Data	9.05 ± 0.03	98.53%

Our quantitative results reflect the same findings as our qualitative evaluations. Indeed, Model 2 and Model 4 have higher RS and RA values than Model 1 and Model 3 respectively, providing evidence for the effectiveness of finetuning with a greater emphasis placed on the class loss of the discriminator. In addition, Models 3 and Model 4 also have higher RS and RA values than Model 1 and Model 2 respectively, which unsurprisingly point to the success of higher-capacity models.

Table 2 also indicates the high success of Model 4, which achieves RS and RA values close to that of actual CIFAR-10 data. While this could be partially attributed to the training alterations we propose in Section 2, we acknowledge that this is also due to the mode collapse experienced in some classes, i.e. for some classes, Model 4 (and Model 3) tends to generate very similar images. This behavior is not reflected in Models 1 and 2.

3.3 Latent Space Interpolations of Images

Since Model 4 generate the most realistic images qualitatively and performed the best quantitatively, we elected to plot latent space interpolations of images using Model 4 (Figure 2).

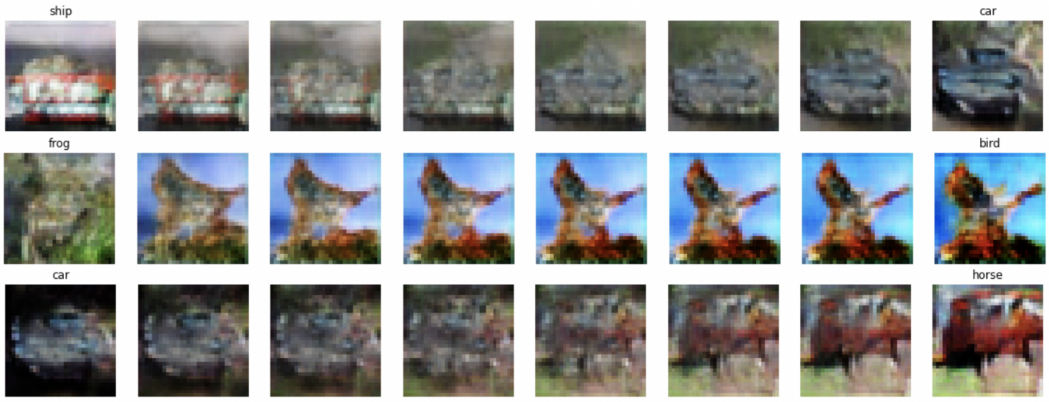

Figure 2: Each row features images generateed from the latent space interpolation between latent vectors corresponding to different classes.

In overfit models, we may expect to see discrete transitions in images, or “holes" in the latent space that correspond to noise [1]. We note that both of these circumstances are not seen in Figure 2, which is a “success" metric outlined in the earlier project proposal.

In addition, as done in [1], we investigate the effect of the 100-dimensional noise vector on the generated images (Figure 3).

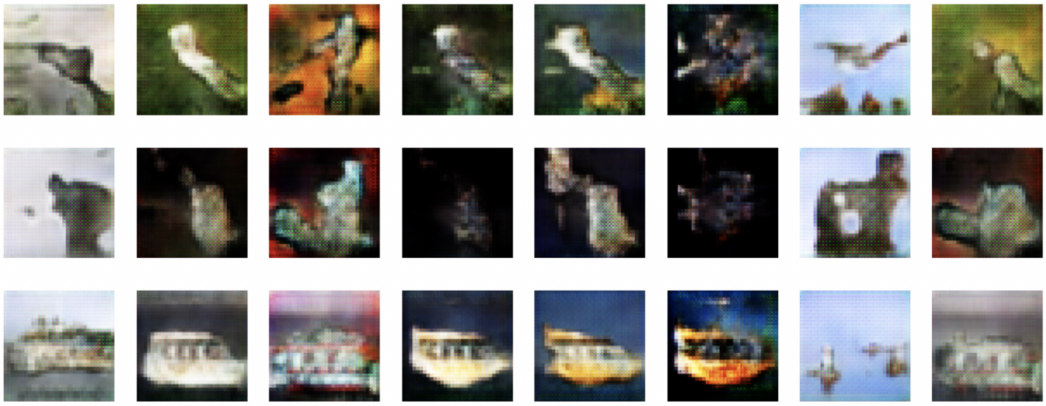

Figure 3: Each row corresponds to a different class label (from top to bottom, bird, cat, ship), while images of the same column are generated with the same noise vector.

Similar to [1], we note that the 100-dimensional noise vector indeed encodes class-independent information, like background color, which indicates that the AC-GAN can represent certain types of “compositionality".

4 Conclusions

In conclusion, we have successfully reproduced [1] to the extent set out in our initial project proposal, implementing and training an AC-GAN on the CIFAR-10 dataset, generating a few samples from each class, calculating an “ResNet Score" (RS) [2] for the model, and performing latent space interpolations between samples. We additionally train 3 more models, calculate the “ResNet Accuracy" (RA) for the images each model generates, and plot other visualizations found in Figure 9 of [1].

Although it is not the focus of this current work, an area of future exploration could be to generate higher quality images with greater diversity while training on CIFAR-10 data. Some potential areas to look into would include:

1. Hyperparameter tuning for multi-task learning. More rigorously, we note that there are 6 terms in lines 7, 8 and 12 of Algorithm 1 that could be tuned to achieve better performance (generating better images), of which we only explored 1 (the class loss of the discriminator).
2. Addressing the mode collapse of higher capacity models by retraining with lower learning rates or higher activation noise standard deviation.
3. Training with many more epochs, which were not done due to the computational time-outs of the VM.

Acknowledgements

We acknowledge the work of other tutorials and implementations of the AC-GAN on the CIFAR-10 dataset [5] [6], which were referenced mainly to ensure that the images generated in this work were of the appropriate quality.

References

[1] A. et al. Odena. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2642–2651, 2017.

[2] T. et al. Salimans. Improved techniques for training gans. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2234–2242, 2016.

[3] M Mirza and S. Osindero. Conditional generative adversarial nets. In *arXiv preprint arXiv:1406.2661*, 2014.

[4] T. et al. Salimans. improved-gan. https://github.com/openai/improved-gan. Accessed: 2021-04-11.

[5] Te-Lin Wu. Conditional image synthesis with auxiliary classifier gans. https://github.com/clvrai/ACGAN-PyTorch. Accessed: 2021-04-11.

[6] Hmrishav Bandyopadhyay. Understanding acgans with code[pytorch]. https://towardsdatascience.com/understanding-acgans-with-code-pytorch-2de35e05d3e4. Accessed: 2021-04-11.

A Timeline and task allocation

- March 29 - April 4: Reading up on GANs, CGANs and AC-GANs.
- April 5 - April 11: Implementing the AC-GAN.
- April 12 - April 19: Implementing and evaluating different models; plotting various figures.
- April 20 - April 28: Finalizing results, preparing for the poster presentation and writing the report.