Class Balancing GAN with a Classifier in the Loop (Supplementary material)

Harsh Rangwani[1] **Konda Reddy Mopuri**[2] **R. Venkatesh Babu**[1]

[1]Indian Institute of Science, Bengaluru
[2]Indian Institute of Technology, Tirupati

1 APPENDIX

1.1 PROOF OF PROPOSITION 2

Proposition 2 [Guiaşu, 1971]: If each $\hat{p_k}$ satisfies Eq. 11, i.e. $\hat{p_k} = e^{\lambda N_k^t - 1}$, then the regularizer objective in Eq. 7 attains the optimal minimum value of $\lambda - \sum_k \frac{e^{\lambda N_k^t - 1}}{N_k^t}$.
Proof: We wish to optimize the following objective:

$$\min_{\hat{p}} \sum_k \frac{\hat{p_k}\log(\hat{p_k})}{N_k^t} \quad \text{such that} \quad \sum_k \hat{p_k} = 1 \tag{1}$$

Introducing the probability constraint via the Lagrange multiplier λ:

$$L(\hat{p}, \lambda) = \sum_k \frac{\hat{p_k}\log(\hat{p_k})}{N_k^t} - \lambda(\sum \hat{p_k} - 1) \tag{2}$$

$$L(\hat{p}, \lambda) - \lambda = \sum_k \frac{\hat{p_k}\log(\hat{p_k})}{N_k^t} - \lambda(\sum \hat{p_k}) \tag{3}$$

$$L(\hat{p}, \lambda) - \lambda = \sum_k \frac{\hat{p_k}\log(\hat{p_k}e^{-\lambda N_k^t})}{N_k^t} \tag{4}$$

$$L(\hat{p}, \lambda) - \lambda = \sum_k \frac{\hat{p_k}\log(\hat{p_k}e^{-\lambda N_k^t})}{N_k^t} \tag{5}$$

$$L(\hat{p}, \lambda) - \lambda = \sum_k \frac{e^{\lambda N_k^t}}{N_k^t}(\hat{p_k}e^{-\lambda N_k^t}\log(\hat{p_k}e^{-\lambda N_k^t})) \tag{6}$$

Now, for $x > 0$, we know that $x\log(x) \geq -\frac{1}{e}$:

$$L(\hat{p}, \lambda) \geq \lambda - \sum_k \frac{e^{\lambda N_k^t - 1}}{N_k^t} \tag{7}$$

The $xlog(x)$ attains the minimal value at only $x = \frac{1}{e}$, which is the point corresponding to $\hat{p_k}e^{-\lambda N_k^t} = 1/e$, for every k. This shows that at $\hat{p_k} = e^{\lambda N_k^t - 1}$ is the optimal point where the objective attains it's minimum value. This result is derived from Theorem 2 in Guiaşu [1971].

1.2 DATASETS

We use CIFAR-10 [Krizhevsky et al., 2009] dataset for our experiments which has 50K training images and 10K validation images. For the LSUN [Yu et al., 2015] dataset we use a fixed subset of 50K training images for each of bedroom, conference room, dining room, kitchen and living room classes. In total we have 250K training images and 1.5K validation set of images for LSUN dataset. The imbalanced versions of the datasets are created by removing images from the training set. For the large dataset experiments, we make use of CIFAR-100 [Krizhevsky, 2009] and iNaturalist-2019 [iNaturalist, 2019]. The CIFAR-100 dataset is composed of the 500 training images and 100 testing images for each class. The iNaturalist-2019 is a long-tailed dataset composed of the 268,243 images present across 1010 classes in the training set, the validation set is composed of 3030 images balanced across classes.

1.3 ARCHITECTURE DETAILS FOR GAN

We use the SNDCGAN architecture for experiments on CIFAR-10 and SNResGAN architecture for experiments on LSUN, CIFAR-100 and iNaturalist-2019 datasets [Gulrajani et al., 2017, Miyato et al., 2018]. The notation for the architecture tables are as follows: m is the batch size, FC(dim_in, dim_out) is a fully connected Layer, CONV(channels_in, channels_out, kernel_size, stride) is convolution layer, TCONV(channels_in, channel_out, kernel_size, stride) is the transpose convolution layer, BN is BatchNorm [Ioffe and Szegedy, 2015] Layer in case of unconditonal GANs and conditional BatchNorm in case of conditional GANs. LRelu is the leaky relu activation function and GSP is the Global Sum Pooling Layer. The DIS_BLOCK(channels_in, channels_out, downsampling) and GEN_BLOCK(channels_in, channels_out, upsampling) correspond to the Discriminator and Generator block used in Gulrajani et al. [2017]. The SNResGAN architecture for CIFAR-100 differs by a small amount as it has 32×32 image size, for which we use the exact same architecture described in Miyato et al. [2018]. The architectures are presented in detail in Tables 2, 3, 4 and 5.

Imbalance Ratio (ρ)	100	10	1
CIFAR-10	10	7.5	5
LSUN	20	7.5	5

Table 1: Values of λ for different imbalance cases. For LSUN the λ gets divide by 5 and for λ it gets divided by 10 before multiplication to regularizer term.

1.4 HYPERPARAMETER CONFIGURATION (IMAGE GENERATION EXPERIMENTS)

1.4.1 Lambda the Regularizer coeffecient

The λ hyperparameter is the only hyperparameter that we change across different imbalance scenarios. As the overall objective is composed of the two terms:

$$L_g = -E_{(x,z)\sim(P_r,P_z)}[\log(\sigma(D(G(z)) - D(x))] + \lambda L_{reg} \tag{8}$$

As the number of terms in the regularizer objective can increase with number of classes K. For making the regularizer term invariant of K and also keeping the scale of regularizer term similar to GAN loss, we normalize it by K. Then the loss is multiplied by λ. Hence the effective factor that gets multiplied with regularizer term is $\frac{\lambda}{K}$.

The presence of pre-trained classifier which provides labels for generated images makes it easy to determine the value of λ. Although the pre-trained classifier is trained on long-tailed data its label distribution is sufficient to provide a signal for balance in generated distribution. We use the KL Divergence of labels with respect to uniform distribution for 10k samples in validation stage to check for balance in distribution and choose λ accordingly. We use the FID implementation available here [1].

1.4.2 Other Hyperparmeters

We update the effective class distribution periodically after 2k updates (i.e. each cycle defined in section 3 consists of 2k iteration). We find the algorithm performance to be stable for a large range of update frequency depicted in Figure 1. We

[1]https://github.com/mseitzer/pytorch-fid

Layer	Input	Output	Operation
Input Layer	(m, 128)	(m, 8192)	FC(128, 8192)
Reshape Layer	(m, 8192)	(m, 4, 4, 512)	RESHAPE
Hidden Layer	(m, 4, 4, 512)	(m, 8, 8, 256)	TCONV(512, 256, 4, 2),BN,LRELU
Hidden Layer	(m, 8, 8, 256)	(m, 16, 16, 128)	TCONV(256, 128, 4, 2),BN,LRELU
Hidden Layer	(m, 16, 16, 128)	(m, 32, 32, 64)	TCONV(128, 64, 4, 2),BN,LRELU
Hidden Layer	(m, 32, 32, 64)	(m, 32, 32, 3)	CONV(64, 3, 3, 1)
Output Layer	(m, 32, 32, 3)	(m, 32, 32, 3)	TANH

Table 2: Generator of SNDCGAN [Miyato et al., 2018, Radford et al., 2015] used for CIFAR10 image synthesis.

Layer	Input	Output	Operation
Input Layer	(m, 32, 32, 3)	(m, 32, 32, 64)	CONV(3, 64, 3, 1), LRELU
Hidden Layer	(m, 32, 32, 64)	(m, 16, 16, 64)	CONV(64, 64, 4, 2), LRELU
Hidden Layer	(m, 16, 16, 64)	(m, 16, 16, 128)	CONV(64, 128, 3, 1), LRELU
Hidden Layer	(m, 16, 16, 128)	(m, 8, 8, 128)	CONV(128, 128, 4, 2), LRELU
Hidden Layer	(m, 8, 8, 128)	(m, 8, 8, 256)	CONV(128, 256, 3, 1), LRELU
Hidden Layer	(m, 8, 8, 256)	(m, 4, 4, 256)	CONV(256, 256, 4, 2), LRELU
Hidden Layer	(m, 4, 4, 256)	(m, 4, 4, 512)	CONV(256, 512, 3, 1), LRELU
Hidden Layer	(m, 4, 4, 512)	(m, 512)	GSP
Output Layer	(m, 512)	(m, 1)	FC(512, 1)

Table 3: Discriminator of SNDCGAN [Miyato et al., 2018] used for CIFAR10 image synthesis.

Layer	Input	Output	Operation
Input Layer	(m, 128)	(m, 16384)	FC(128, 16384)
Reshape Layer	(m, 16384)	(m, 4, 4, 1024)	RESHAPE
Hidden Layer	(m, 4, 4, 1024)	(m, 8, 8, 512)	GEN_BLOCK(1024, 512, True)
Hidden Layer	(m, 8, 8, 512)	(m, 16, 16, 256)	GEN_BLOCK(512, 256, True)
Hidden Layer	(m, 16, 16, 256)	(m, 32, 32, 128)	GEN_BLOCK(256, 128, True)
Hidden Layer	(m, 32, 32, 128)	(m, 64, 64, 64)	GEN_BLOCK(128, 64, True)
Hidden Layer	(m, 64, 64, 64)	(m, 64, 64, 3)	BN, RELU, CONV(64, 3, 3, 1)
Output Layer	(m, 64, 64, 3)	(m, 64, 64, 3)	TANH

Table 4: Generator of SNResGAN used for LSUN and iNaturalist-2019 image synthesis.

Layer	Input	Output	Operation
Input Layer	(m, 64, 64, 3)	(m, 32, 32, 64)	DIS_BLOCK(3, 64, True)
Hidden Layer	(m, 32, 32, 64)	(m, 16, 16, 128)	DIS_BLOCK(64, 128, True)
Hidden Layer	(m, 16, 16, 128)	(m, 8, 8, 256)	DIS_BLOCK(128, 256, True)
Hidden Layer	(m, 8, 8, 256)	(m, 4, 4, 512)	DIS_BLOCK(256, 512, True)
Hidden Layer	(m, 4, 4, 512)	(m, 4, 4, 1024)	DIS_BLOCK(512, 1024, False), RELU
Hidden Layer	(m, 4, 4, 1024)	(m, 1024)	GSP
Output Layer	(m, 1024)	(m, 1)	FC(1024, 1)

Table 5: Discriminator of SNResGAN [Miyato et al., 2018, Gulrajani et al., 2017] used for LSUN and iNaturalist-2019 image synthesis.

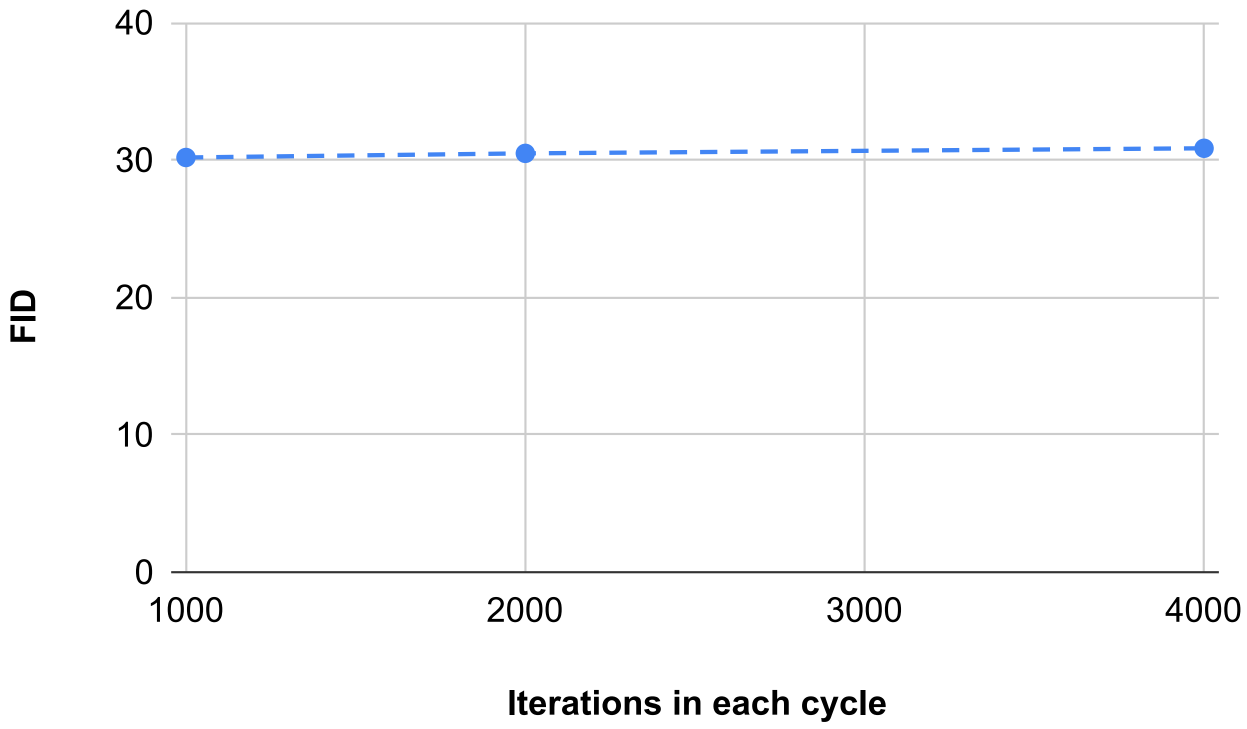


Figure 1: Effect on FID with change in number of steps in each cycle. After each cycle effective class statistics are updated (For CIFAR-10 imbalance ratio $\rho = 10$).

also apply Exponential Moving Average on generator weights after 20k steps for better generalization. The hyperparameters are present in detail in Table 6.

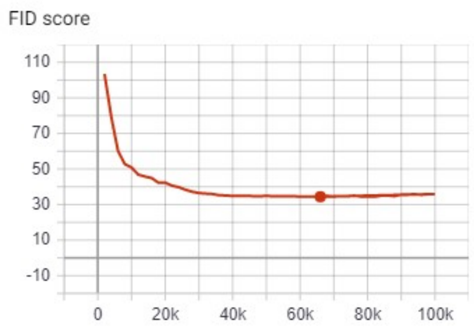
Validation Step: We obtain the FID on 10k generated samples after each 2k iterations and choose the checkpoint with best FID for final sampling and FID calculation present in Table 1.

Convergence of Network: We find that our GAN + Regularizer setup also achieves similar convergence in FID value to the GAN without the regularizer. We show the FID curves for the CIFAR-10 (Imbalance Ratio = 10) experiments in Figure 2.

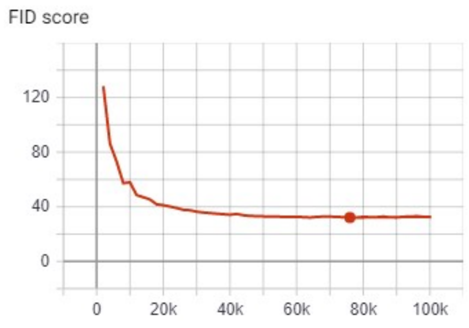
Ablation on β: We find that for the CIFAR-10 dataset ($\rho = 10$) the choice of $\beta = 1$ obtains similar FID (30.48) to $\beta = \alpha$ which obtains FID of 30.46. The KL Divergence is also approximately the same for both cases i.e. 0.01.

Parameter	Values(CIFAR-10)	Values(LSUN)
Iterations	100k	100k
β	1	1
Generator lr	0.002	0.002
Discriminator lr	0.002	0.002
Adam (β_1)	0.5	0.0
Adam (β_2)	0.999	0.999
Batch Size	256	256
EMA(Start After)	20k	20k
EMA(Decay Rate)	0.9999	0.9999

Table 6: Hyperparameter Setting for Image Generation Experiments.



(a) GAN



(b) GAN + Proposed Regularizer

Figure 2: Plots of FID (y axis) vs Number of iteration steps. We observe a similar curve in both the cases.

1.4.3 Hyperparameters for the Semi Supervised GAN Architecture

We use a ImageNet and ImageNet-21k pre-trained model with ResNet 50 architecture as the base model. The fine tuning of the model on CIFAR-10 and LSUN has been done by using the code of notebook present here [2]. The accuracy of the classifiers fine-tuned on validation data, trained with 0.1% of labelled data is 84.96 % for CIFAR-10 and 82.40 % for LSUN respectively. The lambda (regularizer coefficient) values are present in the table below:

Imbalance Ratio (ρ)	100	10
CIFAR-10	10	7.5
LSUN	10	7.5

Table 7: Values of λ for different imbalance cases. For LSUN the λ gets divide by 5 and for λ it gets divided by 10 before multiplication to regularizer term.

The training hyper parameters are same as the ones present in the Table 6. Only in case of LSUN semi supervised experiments we use a batch size of 128 to fit into GPU memory for semi supervised experiments.

1.5 EXPERIMENTAL DETAILS ON CIFAR-100 AND INATURALIST-2019

1.5.1 CIFAR-100

In this section we show results on CIFAR-100 dataset which has 100 classes having 500 training images for each class. We use SNResGAN architecture from Miyato et al. [2018] for generating 32×32 size images, which is similar to SNResGAN architecture used for LSUN experiments. We use the same hyperparameters used for LSUN experiments listed in Table 6. We use a λ value of 0.5 for CIFAR-100 experiments, the effective value that will get multiplied with L_{reg} is $\frac{0.5}{100}$. The results in Table 4 show that our method on long-tailed CIFAR100 of using GAN + Regularizer achieves the best FID and also have class balance similar to cGAN (conditional GAN). The labels for the samples generated by GAN are obtained by a classifier trained on balanced CIFAR-100 dataset for KL Divergence calculation. The KL Divergence between the GAN label distribution and uniform distribution is present in Table 4. The classifier for obtaining class labels for KL Divergence evaluation is trained on balanced CIFAR-100 with setup described in 4 which serves as annotator for all methods. The balanced classifier achieves an accuracy of 70.99% and the pre-trained classifier trained on long-tailed data achieves an accuracy of 57.63%. The pre-trained classifier is used in the process of GAN training.

1.5.2 iNaturalist-2019

We use SNResGAN architecture described in Table 5 and Table 2 for generating 64×64 images for the iNaturalist 2019 dataset. In case of iNaturalist all batch-norms are conditional batch norms (cBN) in Generator, in case of our method and

[2]https://github.com/google-research/big_transfer/blob/master
/colabs/big_transfer_pytorch.ipynb

the unconditional baseline (SNResGAN) we use random labels as conditioning labels. As in our method have access to class distribution N_k^t we sample random labels with weight distribution proportional to $1/N_k^t$. With this change we see an FID decrease from 11.58 to 9.01. The λ value of 0.5 and $\alpha = 0.005$ is used for the experiments. The effective value of λ that will be multiplied with L_{reg} is $\frac{0.5}{1010}$. The statistics N_k is updated for each iteration in this case. We follow SAGAN [Zhang et al., 2019] authors recommendation and use spectral normalization in generator in addition to the discriminator for stability on large datasets. Other hyperparameters are present in Table 8.

Parameter	Values(CIFAR-100)	Values(iNat19)
Iterations	100k	200k
β	1	α
Generator lr	0.002	0.002
Discriminator lr	0.002	0.002
Adam (β_1)	0.5	0.0
Adam (β_2)	0.999	0.999
Batch Size	256	256
EMA(Start After)	20k	20k
EMA(Decay Rate)	0.9999	0.9999

Table 8: Hyperparameter Setting for Image Generation Experiments on CIFAR-100 and iNatuaralist-2019

The pre-trained classifier for iNaturalist-2019 is ResNet-32 trained with usual cross entropy loss with learning schedule as described in Appendix 4. The classifier is trained on resolution of 224×224 and achieves an accuracy of 46.90 % on the validation set. As in iNaturalist 2019 case we don't have balanced training set available we use this classifier to get the KL Divergence from uniform distribution.

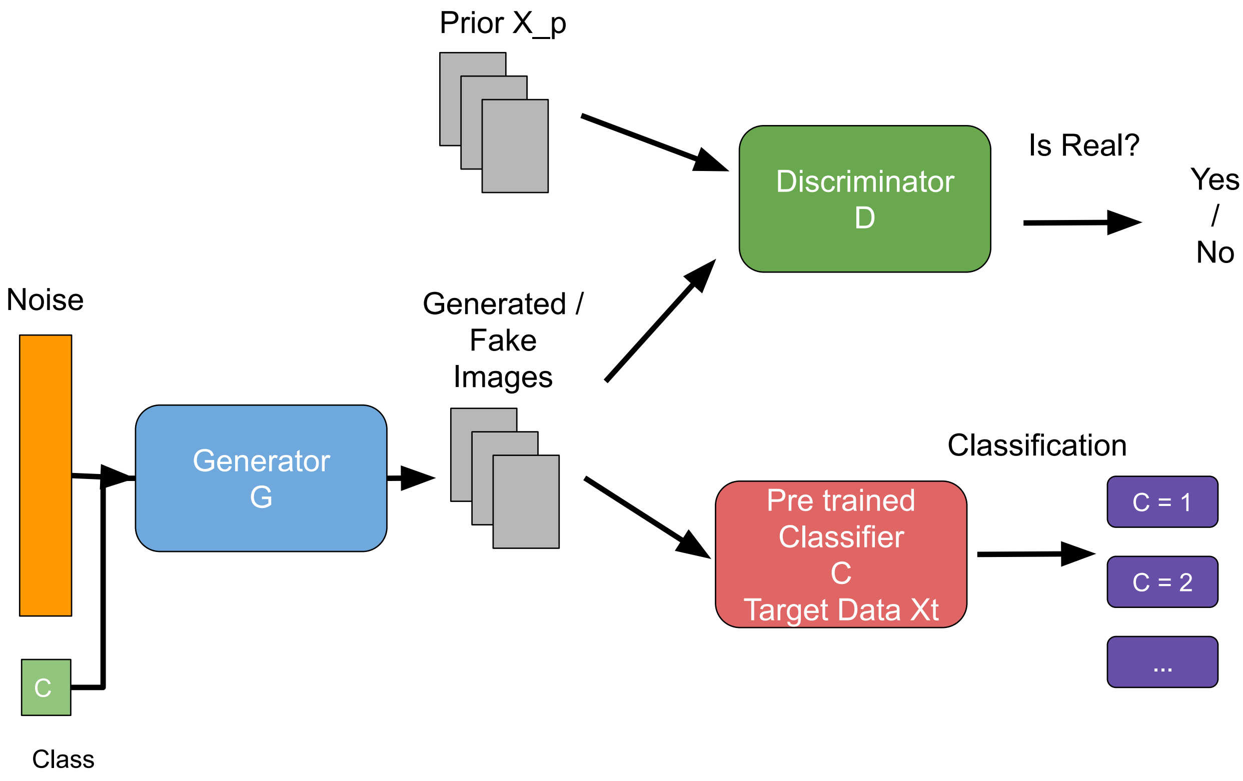
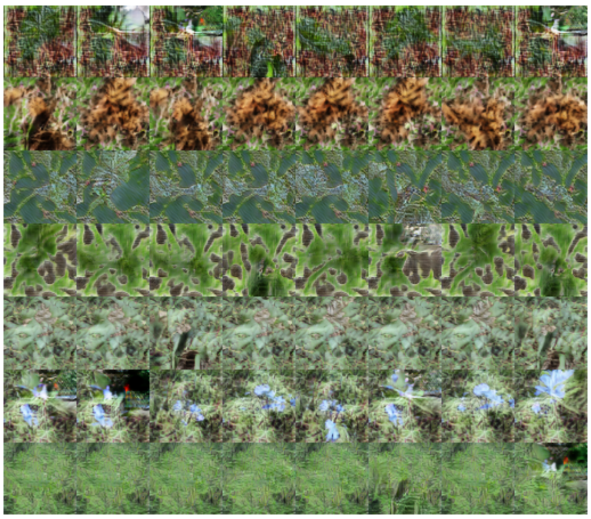


Figure 3: Diagram showing the overview of the other baseline.

1.6 PRE-TRAINED CLASSIFIER BASELINE

In these experiments we introduce a pre-trained classifier in place of the classifier being jointly learned by ACGAN on iNaturalist-2019 dataset. This makes the comparison fair as both approaches use a pre-trained classifier and unlabelled images. The hyperparameters used are same as present in Table 8. We use a classification loss (i.e. cross entropy loss) in addition to the GAN loss similar to ACGAN. The λ value for the classification loss is set to 0.5. The illustration of the approach is present in the Figure 2. We find that this approach leads to mode collapse and is not able to produce diverse samples within each class. The comparison of the generated images is present in the Figure 4.


(a) Classifier Baseline

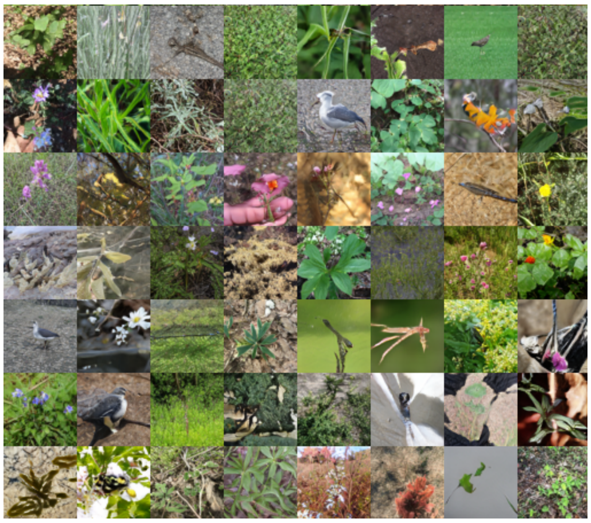
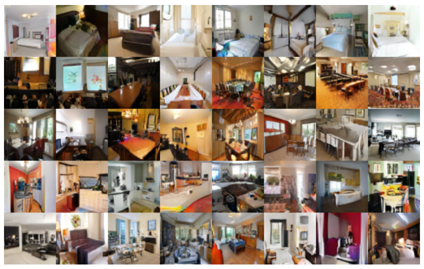
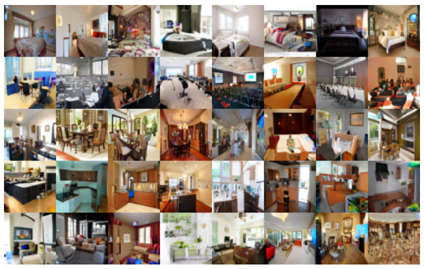
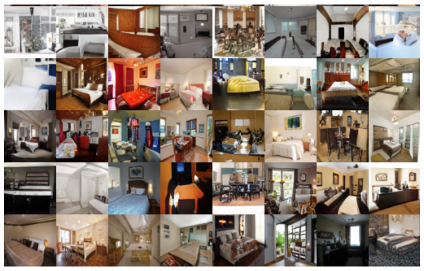

(b) Our method

Figure 4: Qualitative comparison of images generated by ACGAN like baseline and our method.


(a) ACGAN (Conditional)


(b) cGAN (Conditional)


(c) SNResGAN (Unconditional)

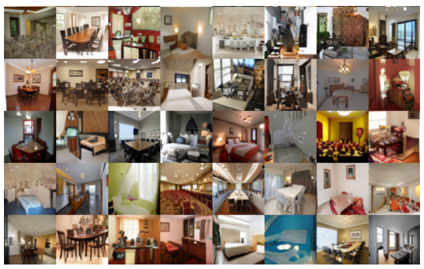
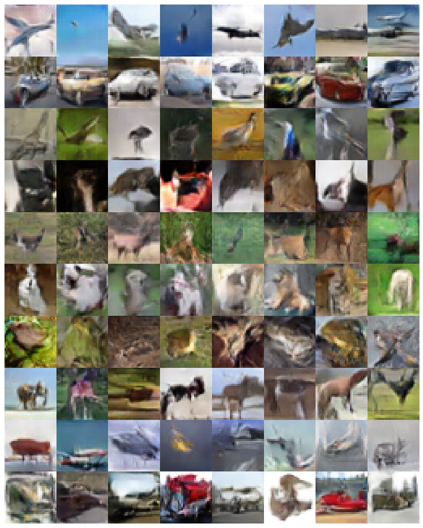
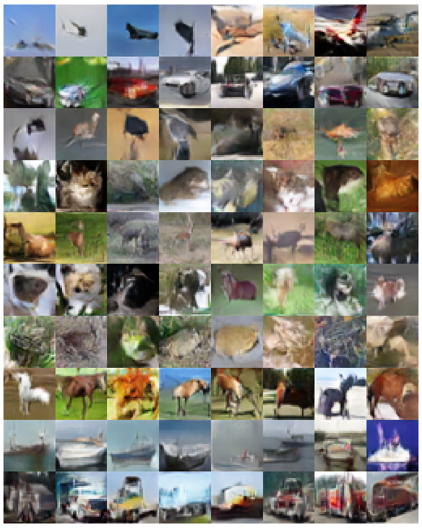
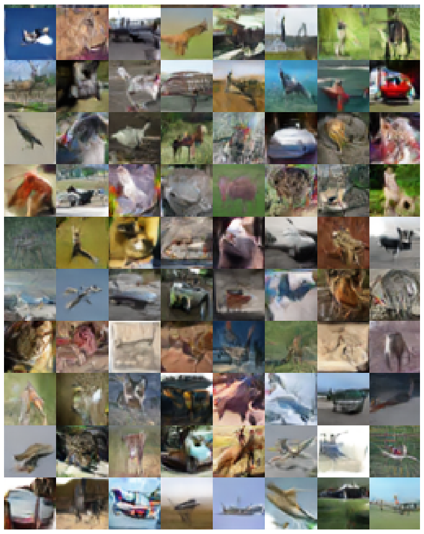

(d) Ours (Unconditional)

Figure 5: Images from different GANs with imbalance ratio ($\rho = 10$)


(a) ACGAN (Conditional)


(b) cGAN (Conditional)


(c) SNDCGAN (Unconditional)

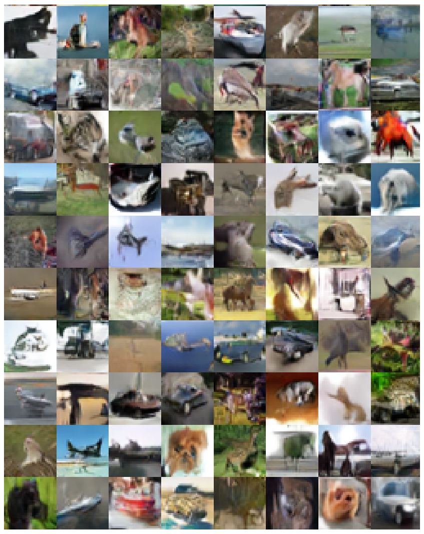

(d) Ours (Unconditional)

Figure 6: Images generated by different GANs for CIFAR-10 with imbalance ratio ($\rho = 10$).

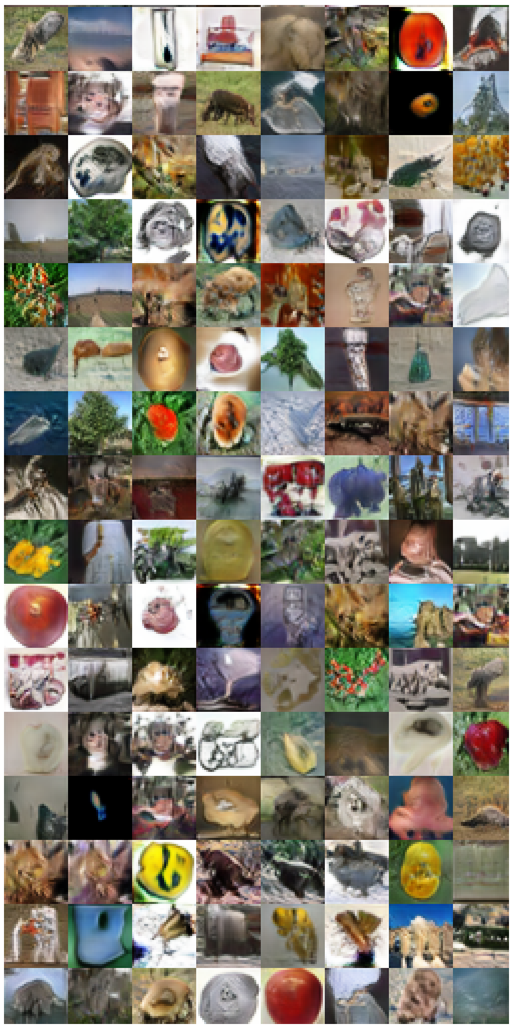


Figure 7: Images generated for CIFAR-100 dataset with our method (GAN + Regularizer).

References

Silviu Guiaşu. Weighted entropy. *Reports on Mathematical Physics*, 2(3):165–179, 1971.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

iNaturalist. The inaturalist 2019 competition dataset. `https://github.com/visipedia/inat_comp/tree/2019`, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

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

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015. URL `http://dblp.uni-trier.de/db/journals/corr/corr1506.html#YuZSSX15`.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363. PMLR, 09–15 Jun 2019. URL `http://proceedings.mlr.press/v97/zhang19d.html`.