A Guided Learning Approach for Generative Adversarial Networks

Sidhant Nagpal, Siddharth Verma, Shikhar Gupta and Swati Aggarwal*
Department of Computer Engineering
Netaji Subhas Institute of Technology, Delhi, India
{sidhantn.co, siddharthv.co, shikharg.co}@nsit.net.in, swati1178@gmail.com
*Corresponding author

***Abstract*—In this paper, we propose a novel technique for training Generative Adversarial Networks (GANs) using autoencoders. GANs, in recent years, have emerged as one of the most popular generative models. Despite their success, there are several challenges in maintaining the trade-off between diversity and quality of the generated distribution. Our idea stems from the fact that deeper layers of an autoencoder contain high-level feature representation of the input data distribution. Reusing these layers provides GAN with information about the representative characteristics of real data and hence can guide its adversarial training. We call our model Guided GAN since the autoencoder (guiding network) provides a direction to train the GAN (generative network). Guided GAN also minimizes both the forward and reverse Kullback-Leibler (KL) divergence in a single model, exploiting the complementary statistical properties of the two. We conduct extensive experiments and use various metrics for assessing the quality, diversity of generated images and convergence of the model. Our model is evaluated on two standard datasets: CIFAR-10 and CelebA demonstrating either superior or competitive performance compared to baseline GANs, especially in the earlier training stages. Our guided training procedure has been tested on different baseline GANs without any changes to their hyper-parameter configuration or architecture.**

I. Introduction

Reusing feature representations from trained layers of deep neural networks is a popular technique to utilize low-level or high-level information of a data distribution. For instance, in transfer learning [1], the low-level feature layers from a pre-trained neural network are reused as frozen layers (parameters are fixed) in another network, to act as feature extractors. In Coupled GAN [2], weight-sharing is used in the initial layers of generator and final layers of discriminator of two GANs to capture high-level semantics and to effectively learn a joint distribution. Our idea is based on the fact that encoder and decoder learn hierarchical feature representations, making deeper layers of the autoencoder capture high-level semantics of the data distribution. We find that reusing this semantic information about real data aids the training of GAN, thereby improving the generated samples.

Autoencoders have been part of the history of neural networks for long and have been used for the task of representation learning [3], [4]. Autoencoders learn the implicit structure of the training data, i.e., the correlation among the input variables. They capture the average diversity of the distribution by minimizing the forward KL divergence. The structure of the model consists of an encoder network which passes the input data into a bottleneck (a layer consisting of fewer neurons), thereby compressing the data into its essential features, and a decoder network which tries to reconstruct the original data from the compressed features. The networks are trained together by minimizing the reconstruction loss between the generated and real data.

In generative modelling, a model is trained to learn the underlying distribution of input data. This helps the model to produce new samples from the learnt data distribution. Some of the popular generative models are Restricted Boltzmann Machines (RBMs) [5] and Variational Autoencoders (VAEs) [6] which maximize the likelihood of an explicitly defined density function by following the gradient uphill. Other models, such as GANs, use a game-theoretic approach for modelling instead of working with any explicit density function.

GAN was first proposed by Goodfellow *et al.* [7] in 2014, which accelerated the research in generative models as the images produced by it were significantly sharper in quality compared to RBMs and VAEs. A typical GAN setup consists of two neural networks - generator and discriminator, opposing each other in a minimax game. The generator produces samples from random input noise and the discriminator judges the authenticity of the images by trying to classify generated images as fake and dataset images as real. The two networks are differentiable functions that are trained by optimizing the minimax objective function, thereby minimizing the reverse KL divergence.

It is well known that GANs are notoriously difficult to train whereas autoencoders have a fairly straightforward training procedure. In this work, we leverage the architectural similarity of the constituent networks of GAN and autoencoder by reusing feature representations. Since encoder and decoder (in autoencoder) are structurally similar to discriminator and generator (in GAN) respectively, we use the initial trained layers of decoder in generator and the latter trained layers of encoder in discriminator to provide GAN with a training direction.

Briefly, the major highlights of this work are:

1) A novel training procedure for GANs using autoencoders that advocates quicker generation of images with competitive or superior quality and diversity.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

2) An intuitive insight manifesting why the guided training technique works.
3) A comprehensive evaluation demonstrating the effectiveness of our approach, both quantitatively and qualitatively, on different baseline GANs for multiple datasets.

II. Background

This section provides a background of GANs and autoencoders. As we describe both the models, it is worth noting the architectural similarities between them to develop an intuitive understanding of the proposed model.

A. Generative Adversarial Network

As stated in Section I, GANs consist of two neural networks: a generator network and a discriminator network. The generator G takes as input a noise vector z from a prior distribution $p_z(z)$ and tries to produce a sample $G(z)$ such that it is part of the real data distribution $p_{data}(x)$. These generated samples along with real data samples are given as input x to the discriminator D. The discriminator outputs $D(x)$ which is the probability that the samples are from the true distribution $p_{data}(x)$. The two networks optimise the following minimax objective function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{ data }}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \quad (1)$$

Hence, generator tries to fool discriminator to minimize the objective, while discriminator tries to correctly classify the samples as real or fake to maximize the objective.

B. Autoencoder

An autoencoder is a type of neural network which is trained for feature learning through dimensionality reduction. It consists of two neural networks: encoder and decoder. During the training process, the encoder network f converts the input data x into a compressed representation $h = f(x)$. The decoder network g then decompresses this representation h into a reconstruction $r = g(h)$ resembling the input data [8]. These networks are optimized by minimizing the mean squared error between r and x, also called reconstruction loss:

$$||g(f(x)) - x||^2 \quad (2)$$

Although in recent years, autoencoders have been modified to be used as generative models [9], [10], these advancements are of less interest to us as our idea is an improvement upon GANs through the use of a pre-trained autoencoder.

C. Related work

In recent years, various solutions have been proposed to overcome the challenges associated with training GANs. Several architectural improvements such as [11]–[13] were made to solve mode collapse [14] and training instability of GANs. There have also been attempts at improving the performance of GANs by changing the loss function, creating variations in associated divergence. One prominent example is WGAN [15] in which the authors changed the cost function to minimize the Earth-Mover distance between real distribution and model distribution, instead of minimizing the Jensen-Shannon divergence.

In the past, some ideas have been proposed where autoencoders were made to act as generative models using GANs. In VAE-GAN [16], VAE and GAN were placed in series and the generator and the decoder were collapsed into one. This work was an improvement on VAEs rather than GANs, wherein the discriminator's output was used for calculating the loss of VAE. In adversarial autoencoder [17], a discriminator was used to distinguish the output of encoder and the samples taken from an input noise distribution. This would drive the encoder's output distribution towards the input noise distribution and hence the decoder is trained to map the input noise to the real data. Our approach is different as we leverage representative features of autoencoder to train GAN rather than using GAN to modify autoencoder.

Several approaches have been tried where autoencoder is used to improve GAN training. EBGAN [18] and BEGAN [19] models use an energy-based function to calculate the loss, resulting in stable convergence of the GAN. Here, the discriminator of GAN was replaced by a trained autoencoder to obtain the mean squared error between the generated images and the output of the autoencoder. This error was used to drive the generated distribution towards the real distribution. In contrast, our approach keeps the discriminator and generator architecture intact, providing both the networks with high-level feature information from autoencoder.

The approach which is partly related to ours is RFGAN [20] which uses the representative features learnt by an encoder in the discriminator. Here, the last convolution layer of encoder from autoencoder is concatenated with that of discriminator. This results in a slight improvement in the quality of generated samples. On the other hand, our approach involves reusing feature layers of both encoder and decoder from autoencoder in discriminator and generator respectively. We also study the effect of varying the number of autoencoder layers reused in GAN. Comparison between the performance of Guided GAN and RFGAN has been provided in Section IV.

III. Guided GAN

In this section, the proposed approach is described in detail. Referring to Fig. 1, our model consists of two independent networks sharing similar architectures. For this work, we refer to the first network as guide network (autoencoder) and the second as generative network (GAN).

A. Our Approach

Usually, the layers of discriminator and generator are randomly initialised from a normal distribution. In the proposed approach, we instead replace the initial layers of generator and latter layers of discriminator with the initial layers of decoder and latter layers of encoder from the pre-trained autoencoder respectively. The remaining layers are initialised randomly as

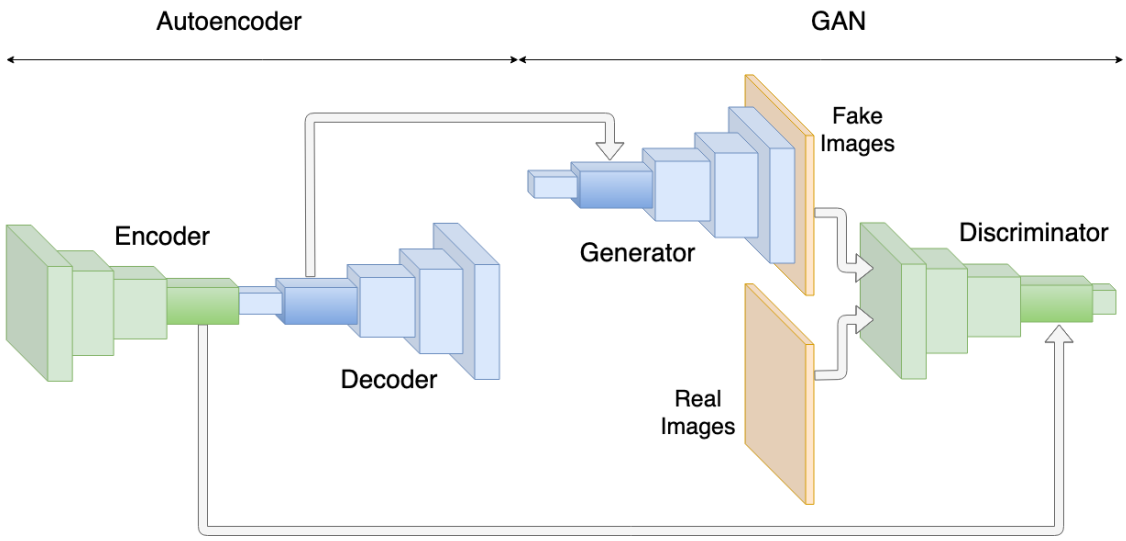

Fig. 1. An illustration of our proposed Guided GAN.

before. The deeper layers of autoencoder contain high-level abstractions of real data which are useful for generation of samples by the generator and classification of samples by the discriminator. The low-level features are learnt during adversarial training. By reusing these autoencoder layers, GAN has information about the input data in the form of representative features and hence can learn better.

The proposed Guided GAN is a training technique rather than a different GAN architecture. There is an implicit pressure to improve image quality and diversity without explicitly changing the cost function of baseline GAN. Autoencoder layers provide GAN with supplementary information in the form of representative features of the given data distribution.

Autoencoder minimizes the forward KL divergence [20] between the data distribution $P(X)$ and the model distribution $Q(X)$ over the feature space X.

$$D_{KL}[P(X)\|Q(X)] = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{3}$$

In Equation 3, D_{KL} is the forward KL divergence and is weighted by the real data $P(x)$. Therefore, for $P(x) > 0$, the term $\log \left(\frac{P(x)}{Q(x)} \right)$ contributes to the overall divergence and thus is minimized during the optimization process. This results in a distribution where $Q(X)$ averages out all the modes of the real data distribution $P(X)$ and minimizes the distance between both distributions. It further leads to the generation of images that cover the average of multiple modes of real distribution but may be blurry and unclear.

In contrast, training a GAN minimizes the reverse KL divergence:

$$D_{KL}[Q(X)\|P(X)] = \sum_{x \in X} Q(x) \log \left(\frac{Q(x)}{P(x)} \right) \tag{4}$$

In the equation above, the divergence is weighted by $Q(x)$. Therefore, for $Q(x) > 0$, the difference between the generated and real data distribution should be as low as possible. In the optimization done through this approach, it is desirable for $Q(X)$ to seek a single mode of $P(X)$. Thus, the images in this distribution are sharper but may not be as diverse as in the real data.

Our model uses the strengths of both autoencoder and GANs. Training of autoencoder is carried independently and in isolation to GAN training. The reused autoencoder layers which contain representative features essentially help in the initial training phase by providing a general idea of the input data distribution. In the latter training phase, the discriminator's feedback drives the generator to produce more realistic samples with sharper quality.

We also hypothesize that the above stated high-level feature utilization followed by the adversarial training prevents GAN from developing a bias towards particular types of samples from the distribution (mode collapse [14]). Since the autoencoder is trained by minimizing the forward KL divergence rather than the reverse KL divergence, it is less likely to prefer a particular mode.

B. Implementation

For this work, we use ResNet architecture for WGAN-GP [21] and DCGAN [11] architecture for all other baselines. Also, we use similar layers in autoencoder to ensure that the decoder and encoder resemble the generator and discriminator of the backbone architecture. Table I shows the details of the layers used and the hyper-parameters. We implement our approach using PyTorch and our code is available on GitHub at: https://github.com/sidhantnagpal/guided-gan

We attempt two techniques for using the pre-trained autoencoder layers in GAN. In the first technique, we freeze the reused layers in GAN during its training. Here, freezing means that gradients are not calculated for the parameters during

TABLE I
NETWORK ARCHITECTURE AND HYPER-PARAMETERS OF GUIDED DCGAN FOR CIFAR-10 DATASET.
THE GENERATOR AND DISCRIMINATOR LAYERS MARKED WITH † DENOTE THE AUTOENCODER FEATURE LAYERS REUSED IN GAN.

Operation	Kernel	Stride	Feature maps	BN?	Nonlinearity
$G(z) : z \sim \mathcal{N}(0, I)$			100		
Transposed Convolution†	4×4	1×1	512	✓	ReLU
Transposed Convolution†	4×4	2×2	256	✓	ReLU
Transposed Convolution	4×4	2×2	128	✓	ReLU
Transposed Convolution	3×3	1×1	64	✓	ReLU
Transposed Convolution	4×4	2×2	3	×	Tanh
$D(x)$			$32 \times 32 \times 3$		
Convolution	4×4	2×2	64	×	Leaky ReLU
Convolution	3×3	1×1	128	✓	Leaky ReLU
Convolution	4×4	2×2	256	✓	Leaky ReLU
Convolution†	4×4	2×2	512	✓	Leaky ReLU
Convolution	4×4	1×1	1	×	Sigmoid
Batch size	64				
Number of epochs	65				
Leaky ReLU slope	0.2				
Learning rate	0.0002				
Optimizer	Adam($\beta_1 = 0.5, \beta_2 = 0.999$)				
Weight initialization, bias	$\mathcal{N}(\mu = 0, \sigma = 0.02)$, not used				

back-propagation, hence the weights do not change throughout the training.

In the second technique (our main approach) we reuse the autoencoder layers in GAN but do not freeze them, i.e., the layers are kept trainable. From hereon, we use "reusing layers" to refer to the fact that the autoencoder layers are reused and kept trainable in GAN unless mentioned otherwise. We also try varying the number of reused layers in generator and discriminator. Since discriminator is more prone to overfitting as compared to generator, it is expected that the training will benefit by reusing more layers in generator. This turns out to be true from our analysis. Hence, for the experiments in Section IV, we reuse the last two layers of decoder in generator and the penultimate layer of encoder in discriminator.

To study the applicability of our approach, we employ this guided procedure for training three baseline GANs - DCGAN [11], LSGAN [22] and WGAN-GP [21] and report the results in Section IV-B3. Throughout our evaluations, we use the same hyper-parameters and training configurations as proposed in the original baseline GAN.

IV. EXPERIMENTS

In this section, we demonstrate the effectiveness of our model by conducting experiments on synthetic datasets and large-scale real-world datasets. Synthetic data is chosen for analyzing the data diversity achieved by our model and to ensure that the initial boost in training is not at the cost of lesser diversity. Through evaluations on large-scale datasets, we illustrate the quality of images produced and the applicability of our model on real-world images. Our experiments use a variety of evaluation metrics as well as visual inspection of images for quantitative and qualitative analysis respectively.

A. Synthetic Data

The experiments conducted on real-world data reasonably capture the image quality but the diversity of generated samples is better understood on synthetic data. Therefore, we use synthetic 2D Gaussian datasets to check the diversity of samples produced by Guided GAN. In the first experiment, we demonstrate how our model is able to capture multiple modes in the data by following the setting described in [23]. The training data is created from a 2-dimensional mixture of 8 Gaussian distributions arranged in a circle of radius 2.0 centered at zero with a co-variance matrix of $0.04\boldsymbol{I}$. This low value of variance creates low-probability regions around each mode so that they are separated by some distance from each other in 2D space. We also perform the experiments on another 2D Gaussian dataset containing 25 distributions in a grid [24]. The center and co-variance matrix of 25 Gaussian mixture are chosen such that it makes it harder for the GAN to identify all the distributions which are separated by low-density regions.

Fig. 2 shows the Kernel Density Estimation (KDE) plots of target data and generated data on 8 Gaussian and 25 Gaussian mixtures. We train Guided GAN on 8 Gaussian and 25 Gaussian for 100k steps and 200k steps respectively and compare the plots obtained to that of GAN. The results show that for 8 Gaussian mixture, Guided GAN matches the target distribution closely, which is similar to the behaviour shown by GAN. For 25 Gaussian mixture, Guided GAN is competitive to GAN, showing that it generalises well in capturing diversity for the excessive mode case. The difference between the comparative plots of GAN and Guided GAN is better noted in the initial training stage where Guided GAN captures the modes more effectively.

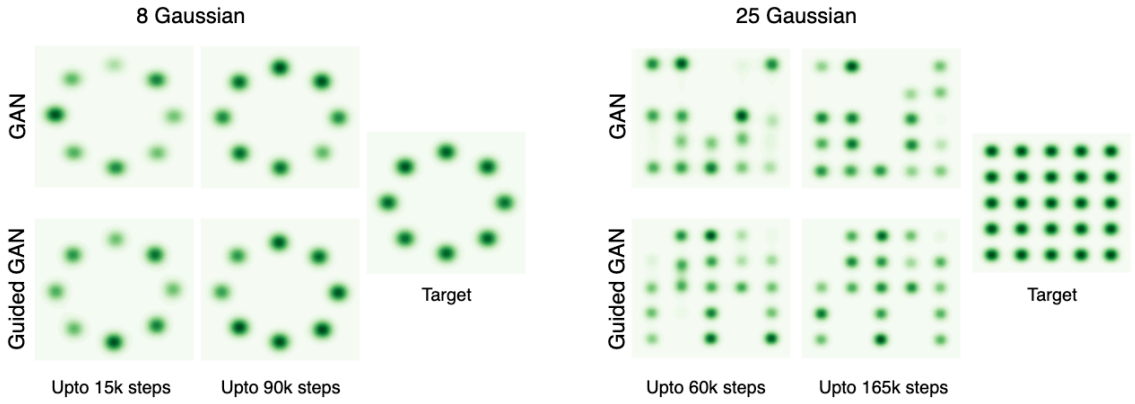

Fig. 2. KDE plots of GAN and Guided GAN on 8 Gaussian and 25 Gaussian mixtures. Note that generated distribution of Guided GAN is competitive (marginally better) to GAN showing that the initial boost during training is not at the cost of lesser diversity.

B. Real-world data

To illustrate the performance of our model in a more practical setup, we use datasets of natural images which are more diverse and bigger in size compared to synthetic data.

1) Datasets: For our analysis we use two common datasets in literature: CIFAR-10 [25] and CelebA [26]. CIFAR-10 is an image classification dataset containing 60k coloured images (50k training and 10k testing) of size 32×32, divided into 10 classes. CelebA (CelebFaces Attributes) dataset contains more than 200k images of celebrity faces having 40 attribute annotations. The distributions of both these datasets are challenging for GANs to learn, given their diverse collection and several intra-class variations.

2) Evaluation metrics: Performance evaluation of generative models is difficult due to the various probability criteria involved [27] and the lack of a definitive similarity metric to check how close the generated samples are to the real distribution. For our experiments, we adopt the two commonly used metrics in literature: Inception score proposed by [28] and Fréchet Inception Distance (FID) [29].

Inception score: This metric is widely used for evaluating generative models as it correlates well with the human way of assessing how realistic the images are. It computes the expression: $\exp\left(\mathbb{E}_x\left[D_{KL}(\,p(y|x)\,||\,p(y)\,)\right]\right)$ where D_{KL} is the KL-divergence between $p(y|x)$, the conditional class distribution, and $p(y)$, the marginal class distribution. Here $p(y|x)$ should have low entropy for sharp and clear images and $p(y)$ should have high entropy to assure diversity of images. To ensure both the criteria, divergence should be large and hence a higher score signifies better quality and diversity of generated images. In our implementation, we use the Inception [30] IV2 network for computing the class probabilities [31].

Fréchet Inception Distance: One disadvantage of using Inception score for the evaluation of generative models is that it does not account for how close the generated distribution is to the real distribution. For calculating FID, we use the activations from the last hidden layer of the Inception network [30]. The aim is to measure the Fréchet Distance [32] between the two multivariate Gaussian distributions (real and generated), also known as the 2-Wasserstein distance. Lower FID is desirable since it quantifies the deviation of the generated distribution from the real distribution.

Therefore, computing the Inception score and FID helps in drawing a comparison based on quality, diversity of generated images and closeness of generated distribution to the real distribution.

3) Quantitative analysis: For this analysis, we judge our model by computing Inception score and FID for CIFAR-10 dataset and also FID for CelebA. We record the Inception score of our two approaches explained in Section III-B and compare it with DCGAN [11] during the course of their training on CIFAR-10 which is shown in Fig. 3. The plot in green corresponds to the first strategy mentioned in Section III-B in which we freeze the pre-trained autoencoder layers during the training of GAN. The plot in red shows the trend for our main approach where we reuse the pre-trained autoencoder layers for the training of GAN and the plot in blue corresponds to DCGAN. The plot also shows that Guided DCGAN performs notably better in the initial epochs as compared to baseline DCGAN and achieves a higher Inception score after convergence. Notice how reusing the layers performs significantly better than freezing them. One possible reason for this behaviour is that the reduction in the number of trainable parameters by freezing the layers inhibits the model's performance. Also, reusing the layers gives an initial training direction to GAN without restricting its learning capability. At the same time, this approach benefits from the adversarial training setup and is, therefore able to yield a better score.

Table II shows the scores obtained for CIFAR-10 dataset. Note that Guided GAN has a higher Inception score and a lower FID in comparison to the corresponding baseline. The table also lists the score of DCGAN-RF discussed in Section II-C. It can be seen from the scores that reusing layers in both generator and discriminator works more effectively than concatenating encoder features in the discriminator. These scores help us deduce that the initial direction provided by

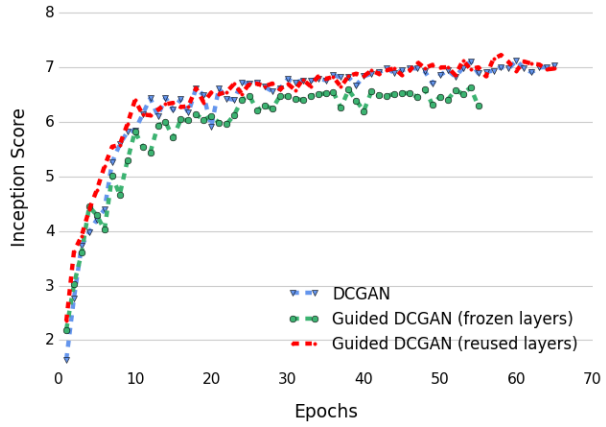

Fig. 3. Variation of Inception score on CIFAR-10 over the training period for our two techniques and DCGAN.

TABLE II
INCEPTION SCORE (HIGHER IS BETTER) AND FID (LOWER IS BETTER) FOR THE CIFAR-10 DATASET

Method	Inception Score	FID
Real Images	11.24 ± 0.11	7.8
LSGAN [22]	6.47	36.5
Guided LSGAN	**6.82**	**30.6**
WGAN-GP [21]	6.20	40.5
Guided WGAN-GP	**6.24**	**38.7**
DCGAN [11]	6.40 ± 0.50	37.7
DCGAN-RF [20]	6.63	-
Guided DCGAN	$\mathbf{7.22 \pm 0.10}$	**26.3**

the autoencoder improves the performance of the adversarial model regardless of the baseline model under consideration.

The convergence plot for initial training of Guided DCGAN is shown in Fig. 4. The improvement in convergence during initial epochs is also obtained when the guided training approach is used for LSGAN and WGAN-GP. This is evident from Fig. 5 and Fig. 6 for Guided LSGAN and Guided WGAN-GP respectively.

Table III shows the FID obtained on CelebA dataset using our approach with WGAN-GP and DCGAN. Our model achieves a lower FID in comparison to the baseline it is built on top of.

4) Qualitative analysis: We also performed a qualitative analysis of our model through visual inspection. Fig. 7 shows samples from the generator of DCGAN and Guided DCGAN. The figure shows that Guided DCGAN is able to produce recognisable faces quite early in the training as compared to

TABLE III
FID (LOWER IS BETTER) OF DIFFERENT MODELS FOR CELEBA DATASET.

Method	FID
WGAN-GP [21]	24.23
Guided WGAN-GP	**23.61**
DCGAN [11]	21.80
Guided DCGAN	**21.02**

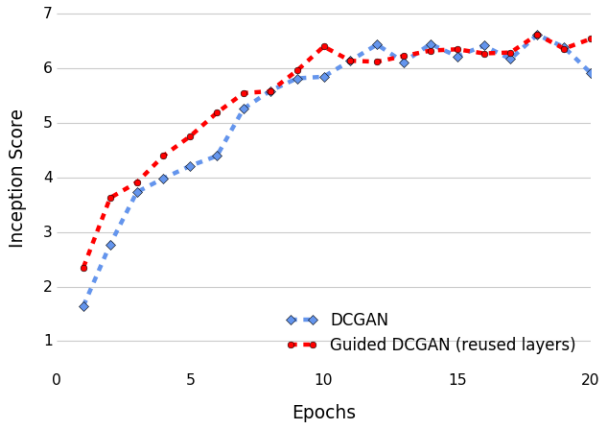

Fig. 4. Variation of Inception score for baseline DCGAN and Guided DCGAN trained on CIFAR-10 dataset in initial 20 epochs.

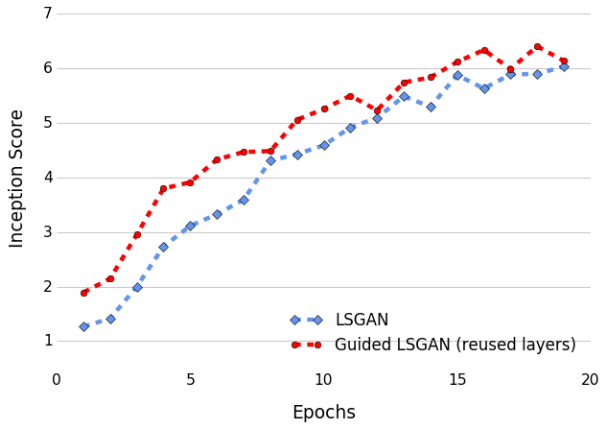

Fig. 5. CIFAR-10 Inception score comparison of LSGAN and Guided LSGAN.

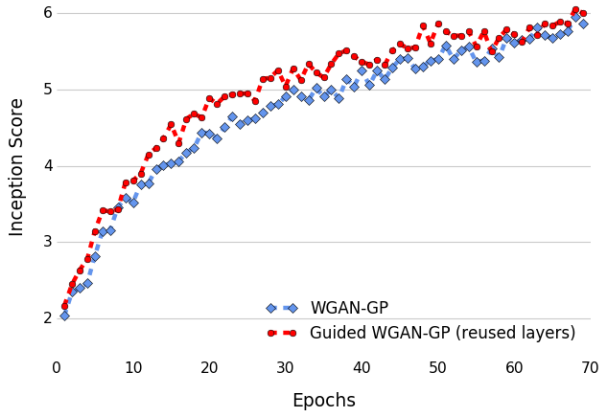

Fig. 6. Variation of Inception score of WGAN-GP and Guided WGAN-GP for CIFAR-10.

Fig. 7. Faces generated at different epochs by DCGAN (top) and Guided DCGAN (bottom). Notice the higher quality images generated by our proposed model compared to the DCGAN samples throughout the training.

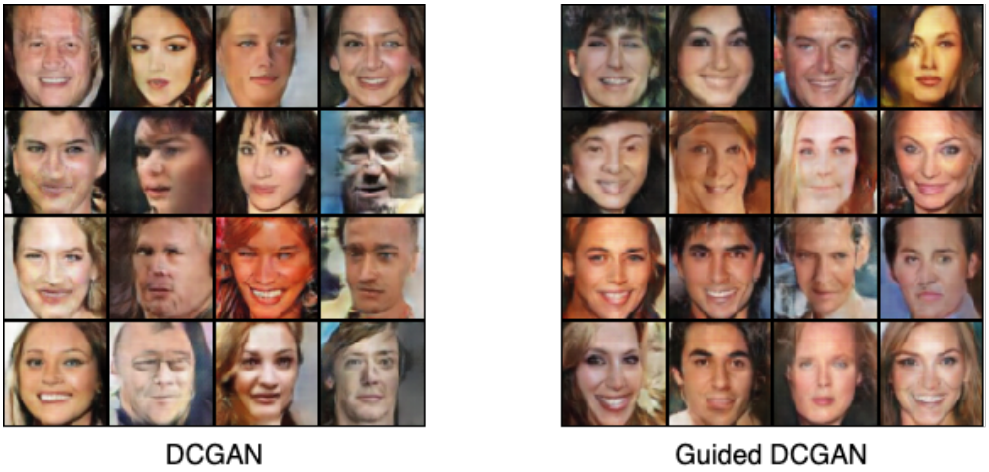

Fig. 8. Randomly drawn samples generated by DCGAN (on the left) and Guided DCGAN (on the right) trained on the CelebA dataset.

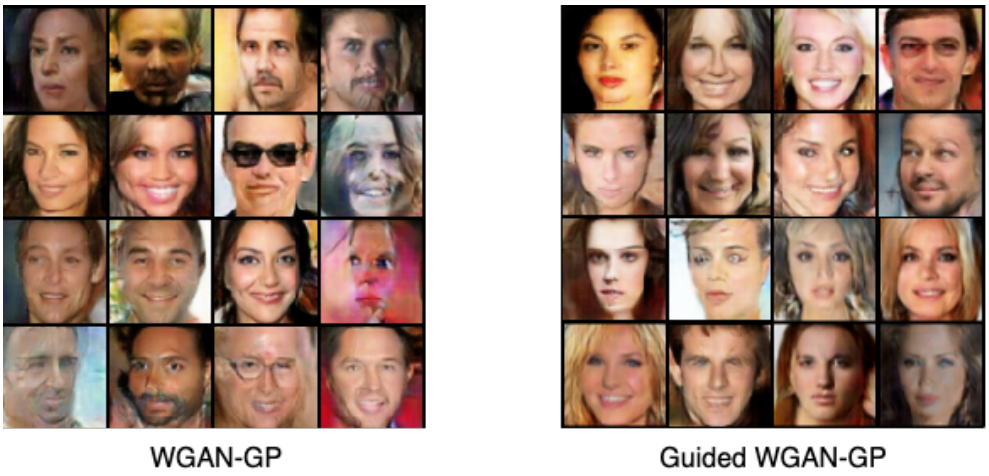

Fig. 9. Randomly drawn samples generated by WGAN-GP (on the left) and Guided WGAN-GP (on the right) trained on the CelebA dataset.

DCGAN. The higher quality of images from Guided GAN is pronounced in the initial epochs. Fig. 8 and Fig. 9 show the comparison of our approach with DCGAN and WGAN-GP respectively on the basis of randomly drawn samples from the best models (the ones with lowest FID). The quality of the images generated by Guided GAN is better, as observed from both the figures.

V. Conclusion

This work proposes a novel procedure for training Generative Adversarial Networks. To the best of our knowledge, this is the first attempt at reusing the deep feature layers of a trained autoencoder in the generator and discriminator to establish a guided adversarial training procedure. Through this, we minimize both forward and reverse KL divergence. This lets the model cover the overall mode range of the input distribution without suffering from a drop in the quality of generated samples. We performed rigorous experiments to evaluate our model from different perspectives like quality, diversity and closeness of generated distribution to real distribution. Our model performed significantly better in the early training stages which is evident from the convergence plots. We evaluated the diversity of generated samples, which showed that our model improved mode coverage both at convergence and also earlier in the training. Our approach also achieved a better Inception score and FID compared to the baseline GAN models. The experiments also show that our training procedure can be applied to any GAN variant irrespective of the architecture or cost function involved. In future, it would be worth trying different guiding networks (like VAE [10]) for possible improvement in the quality, diversity of the generated samples and/or stability of the model. Experimenting with different network initialization techniques and comparing their effect on the overall training would also be an interesting future direction.

References

[1] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI Global, 2010, pp. 242–264.

[2] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in neural information processing systems*, 2016, pp. 469–477.

[3] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Advances in neural information processing systems*, 1994, pp. 3–10.

[4] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4-5, pp. 291–294, 1988.

[5] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[6] D. P. Kingma and M. Welling, "Stochastic gradient vb and the variational auto-encoder," in *Second International Conference on Learning Representations, ICLR*, 2014.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[9] A. B. L. Larsen and S. K. Sønderby, "Generating faces with torch," *http://torch.ch/blog/2015/11/13/gan.html*, 2015.

[10] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[11] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2016.

[12] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems*, 2016, pp. 2172–2180.

[13] T. Nguyen, T. Le, H. Vu, and D. Phung, "Dual discriminator generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2017, pp. 2670–2680.

[14] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

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

[16] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv preprint arXiv:1512.09300*, 2015.

[17] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[18] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.

[19] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[20] D. Bang and H. Shim, "Improved training of generative adversarial networks using representative features," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 433–442. [Online]. Available: http://proceedings.mlr.press/v80/bang18a.html

[21] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.

[22] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2794–2802.

[23] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016.

[24] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, "Adversarially learned inference," *arXiv preprint arXiv:1606.00704*, 2016.

[25] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[26] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3730–3738.

[27] L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," *arXiv preprint arXiv:1511.01844*, 2015.

[28] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems*, 2016, pp. 2234–2242.

[29] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a nash equilibrium," *arXiv preprint arXiv:1706.08500*, vol. 12, no. 1, 2017.

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[31] S. Barratt and R. Sharma, "A note on the inception score," *arXiv preprint arXiv:1801.01973*, 2018.

[32] D. Dowson and B. Landau, "The fréchet distance between multivariate normal distributions," *Journal of multivariate analysis*, vol. 12, no. 3, pp. 450–455, 1982.