Journal of Science and Technology
ISSN: 2456-5660 Volume 06, Issue 03, May-June 2021

www.jst.org.in *DOI: https://doi.org/10.46243/jst.2021.v6.i3.pp169-177*

Deep Convolutional Generative Adversial Network on MNIST Dataset

S. Vijaya Lakshmi[1], Vallik Sai Ganesh Raju Ganaraju[2]
[1]Asst Prof, [2]Student
Dept of CSE, MGIT, Gandipet, Hyderabad
[1]Corresponding Author: svijayalakshmi_ cse@gmail.com

To Cite this Article
S. Vijaya Lakshmi, Vallik Sai Ganesh Raju Ganaraju, "Deep Convolutional Generative Adversial Network on MNIST Dataset", Journal of Science and Technology, Vol. 06, Issue 03, May-June 2021, pp169-177

Article Info
Received: 07-02-2021 Revised: 15-05-2021 Accepted: 28-05-2021 Published: 02-06-2021

Abstract: *In recent years, supervised learning with convolutional networks (CNNs) has seen huge adoption in computer vision applications. Comparatively, unsupervised learning with CNNs has received less attention. In this work we hope to help bridge the gap between the success of CNNs for supervised learning and unsupervised learning. We introduce a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning. Training on various image datasets, we show convincing evidence that our deep convolutional adversarial pair learns a hierarchy of representations from object parts to scenes in both the generator and discriminator. The generator uses tf.keras.layers.conv2Dtranspose (up sampling) layers to produce an image from a seed (random noise). Start with a dense layer that takes this seed as input, then up sample several times until you reach the desired image size of 28x28x1. The discriminator is a CNN-based image classifier. The model will be trained to output positive values for real images, and negative values for fake images. We define the Generator loss and the discriminator loss and we finally we get new images that look similar to our input(MNIST) images.*

Key Word*: Supervised learning, unsupervised learning, Generator, discriminator, Generator loss, Discriminator loss*

I. Introduction

Learning reusable feature representations from large unlabeled datasets has been an area of active research. In the context of computer vision, one can leverage the practically unlimited amount of unlabeled images and videos to learn good intermediate representations, which can then be used on a variety of supervised learning tasks such as image classification. We propose that one way to build good image representations is by training Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), and later reusing parts of the generator and discriminator networks as feature extractors for supervised tasks. GANs provide an attractive alternative to maximum likelihood techniques. One can additionally argue that their learning process and the lack of a heuristic cost function (such as pixel-wise independent mean-square error) are attractive to representation learning. GANs have been known to be unstable to train, often resulting in generators that produce nonsensical outputs. There has been very limited published research in trying to understand and visualize what GANs learn, and the intermediate representations of multi-layer GANs.

Generative Adversarial Networks(GAN) belong to the set of generative models(Goodfellow al.,2014). The GAN model consists of two networks

- A generative network G(.) that takes in random input z and returns x_g=G(z) that should follow the targeted probability distribution.
- A discriminator network D(.) that takes image vector x_image and classifies whether the generated image is real or generated.

The generator needs to learn how to create data in a way that discriminator isn't able to distinguish as fake. The discriminator network has the task to determine if the image is real or fake. An intuitive way to understand

GAN is to imagine a forger trying to create a fake Picasso painting (Chollet, n.d.). At first, the forger(generator) is pretty bad at this task. As times goes on, the forger becomes increasingly competent at imitating the style of Picasso, and the art dealer becomes increasingly expert at spotting fakes. In the end, they have on their hands some excellent fake Picassos. That's what a GAN is: a forger network and an expert network, each being trained to best the other.

Problem Definition: To develop a neural network system that can take the handwritten digit images and generate new images that are similar to the input images.

It is also focused on diminishing the loss of both the discriminator and the generator. So that generator can understand better about the discriminator and produce the outputs that are of mere resemblance to the input images.

Existing System: Deep Convolutional Generative Adversarial Networks or DCGANs are the 'image version' of the most fundamental implementation of GANs. This architecture essentially leverages Deep Convolutional Neural Networks to generate images belonging to a given distribution from noisy data using the Generator-Discriminator framework.

Generative Adversarial Networks use a generator network to generate new samples of data and a discriminator network to evaluate the generator's performance. So, fundamentally, GANs' novelty lies in the evaluator more than that in the generator.

Proposed System: The goal of this project is to develop a neural network system that can take the handwritten digit images and generate new images that are similar to the input images. It is also focused on diminishing the loss of both the discriminator and the generator. So that generator can understand better about the discriminator and produce the outputs that are of mere resemblance to the input images. The proposed system consists of the following goals and advantages.

II. Literature Survey

"Implementing Deep Convolutional Generative Adversial networks" is developed by Rhan Jagtap. Here he generated new images from random data using DCGAN. He implemented DCGAN using TensorFlow and observed the results for different databases. He successfully implemented working with GANs. There are other flavors of GANs that produce conditional outputs and hence can prove to be very useful.

"DCGANs- Generating dog images with Keras and TensorFlow" by Konstantin Georgiev is based on Kaggles generative dogs' competition. Here he genereted images of dogs over epochs. This work is a tutorial on the basic ideas behind the DCGANs, as well as methods to improve their performance. As he observed the MIFID steadily improved for 280 epochs. They are also very computationally intensive and it is incredibly hard to build a high-scoring model for a runtime of ~ 9 hours.

"How to develop a GAN to generate CIFAR-10 small color photographs" by Jason Brownlee develop a generative adversarial network with deep convolutional networks for generating small photographs of objects. He defined and trained the standalone discriminator model for learning the difference between real and fake images. He also defined how to evaluate the performance of the GAN and use the final standalone generator model to generate new images.

"DCGAN- Image Generation" by Ashutosh chapagain explored the potential of Deep Learning to generate real like images. He used Deep Convolutional Generative Adversarial Network (DCGAN) which has proven to be a great success in generating images. He discussed the theoretical aspect of GAN and also discussed about methodology to create a DCGAN Model for CelebA Datasets.

"Unsupervised Representation Learning with Deep Convolutional Generative Adversial Networks" by SoumithChintala. In this work we hope to help bridge the gap between the success of CNNs for supervised learning and unsupervised learning. We introduce a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning.

"Generative Adversial Nets" by Ian J Goodfellow e proposed a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G and Discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake.

III. Related Work

Representation Learning from Unlabeled Data: Unsupervised representation learning is a fairly well studied problem in general computer vision research, as well as in the context of images. A classic approach to unsupervised representation learning is to do clustering on the data (for example using K-means), and leverage the clusters for improved classification scores. In the context of images, one can do hierarchical clustering of image patches to learn powerful image representations. Another popular method is to train auto-encoders (convolutionally, stacked, separating the what and where components of the code, ladder structures (Rasmus et al., 2015)) that encode an image into a compact code, and decode the code to reconstruct the image as accurately as possible. These methods have also been shown to learn good feature representations from image pixels. Deep belief networks have also been shown to work well in learning hierarchical representations.

Generating Natural Images: Generative image models are well studied and fall into two categories: parametric and nonparametric. The non-parametric models often do matching from a database of existing images, often matching patches of images, and have been used in texture synthesis, super-resolution and in-painting. Parametric models for generating images has been explored extensively (for example on MNIST digits or for texture synthesis). However, generating natural images of the real world have had not much success until recently. A variational sampling approach to generating images has had some success, but the samples often suffer from being blurry. Another approach generates images using an iterative forward diffusion process . Generative Adversarial Network generated images suffering from being noisy and incomprehensible. A laplacian pyramid extension to this approach showed higher quality images, but they still suffered from the objects looking wobbly because of noise introduced in chaining multiple models. A recurrent network approach and a deconvolution network approach have also recently had some success with generating natural images. However, they have not leveraged the generators for supervised tasks.

Visualizingthe Internals of CNNs: One constant criticism of using neural networks has been that they are black-box methods, with little understanding of what the networks do in the form of a simple human-consumable algorithm. In the context of CNNs, Zeiler et. al. showed that by using deconvolutions and filtering the maximal activations, one can find the approximate purpose of each convolution filter in the network.

IV. Approach and Model Architechure

Historical attempts to scale up GANs using CNNs to model images have been unsuccessful. This motivated the authors of LAPGAN (Denton et al., 2015) to develop an alternative approach to iteratively upscale low resolution generated images which can be modeled more reliably. We also encountered difficulties attempting to scale GANs using CNN architectures commonly used in the supervised literature. However, after extensive model exploration we identified a family of architectures that resulted in stable training across a range of datasets and allowed for training higher resolution and deeper generative models.

Core to our approach is adopting and modifying three recently demonstrated changes to CNN architectures.

The first is the all-convolutional net (Springenberg et al., 2014) which replaces deterministic spatial pooling functions (such as maxpooling) with strided convolutions, allowing the network to learn its own spatial down sampling. We use this approach in our generator, allowing it to learn its own spatial up sampling, and discriminator.

Second is the trend towards eliminating fully connected layers on top of convolutional features. The strongest example of this is global average pooling which has been utilized in state of the art image classification models (Mordvintsev et al.). We found global average pooling increased model stability but hurt convergence speed. A middle ground of directly connecting the highest convolutional features to the input and output respectively of the generator and discriminator worked well. The first layer of the GAN, which takes a uniform noise distribution Z as input, could be called fully connected as it is just a matrix multiplication, but the result is reshaped into a 4-dimensional tensor and used as the start of the convolution stack. For the discriminator, the last convolution layer is flattened and then fed into a single sigmoid output. See Fig. 1 for a visualization of an example model architecture.

Third is Batch Normalization (Ioffe&Szegedy, 2015) which stabilizes learning by normalizing the input to each unit to have zero mean and unit variance. This helps deal with training problems that arise due to poor initialization and helps gradient flow in deeper models. This proved critical to get deep generators to begin learning, preventing the generator from collapsing all samples to a single point which is a common failure mode observed in GANs. Directly applying batchnorm to all layers however, resulted in sample oscillation and model instability. This was avoided by not applying batchnorm to the generator output layer and the discriminator input layer.

The ReLU activation (Nair & Hinton, 2010) is used in the generator with the exception of the output layer which uses the Tanh function. We observed that using a bounded activation allowed the model to learn more quickly to saturate and cover the color space of the training distribution. Within the discriminator we found the leaky rectified activation (Maas et al., 2013) (Xu et al., 2015) to work well, especially for higher resolution modeling. This is in contrast to the original GAN paper, which used the maxout activation (Goodfellow et al., 2013).

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

V. Details of Adversarial Training

We trained DCGANs on three datasets, Large-scale Scene Understanding (LSUN) (Yu et al., 2015), Imagenet-1k and a newly assembled Faces dataset. Details on the usage of each of these datasets are given below. No pre-processing was applied to training images besides scaling to the range of the tanh activation function [-1, 1]. All models were trained with mini-batch stochastic gradient descent (SGD) with a mini-batch size of 128. All weights were initialized from a zero-centered Normal distribution with standard deviation 0.02. In the LeakyReLU, the slope of the leak was set to 0.2 in all models. While previous GAN work has used momentum to accelerate training, we used the Adam optimizer (Kingma& Ba, 2014) with tuned hyperparameters. We found the suggested learning rate of 0.001, to be too high, using 0.0002 instead. Additionally, we found leaving the momentum term β1 at the suggested value of 0.9 resulted in training oscillation and instability while reducing it to 0.5 helped stabilize training

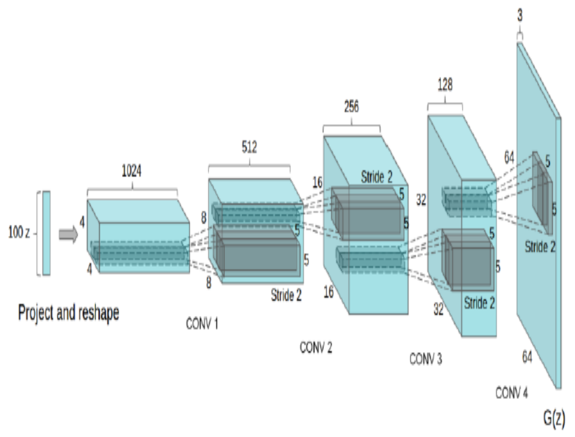

Figure no 1: DCGAN generator used for LSUN scene modeling.

FACES:

We scraped images containing human faces from random web image queries of people's names. The people names were acquired from dbpedia, with a criterion that they were born in the modern era. This dataset has 3M images from 10K people. We run an OpenCV face detector on these images, keeping the detections that are sufficiently high resolution, which gives us approximately 350,000 face boxes. We use these face boxes for training. No data augmentation was applied to the images.

Figure no 2: Generated bedrooms after one training pass through the dataset.

Theoretically, the model could learn to memorize training examples, but this is experimentally unlikely as we train with a small learning rate and minibatch SGD. We are aware of no prior empirical evidence demonstrating memorization with SGD and a small learning rate.

Figure no 3: Generated images after five epochs of training.

VI. Methodology of DCGAN on MNIST Dataset

Datasets:

1. MNIST dataset is a curated list of all handwritten digits. We have used dataset for quick validation of our model. All images are scaled to 28X28.
2. The CelebA dataset consists of over 10k identities and 200k total images. All images are originally of size 160X160 pixels. They are rescaled to 28X28 pixels.

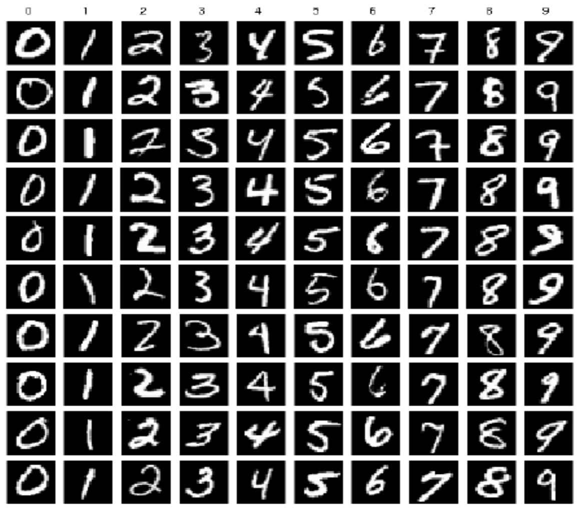

Figure no 4: MNIST Dataset

VII. Empirical validation of DCGANS Capabilities

Classifying MNIST using GANS as a Feature Extractor: One common technique for evaluating the quality of unsupervised representation learning algorithms is to apply them as a feature extractor on supervised datasets and evaluate the performance of linear models fitted on top of these features. On the MNIST dataset, a very strong baseline performance has been demonstrated from a well tuned single layer feature extraction pipeline utilizing K-means as a feature learning algorithm. When using a very large amount of feature maps (4800) this technique achieves 80.6% accuracy. An unsupervised multi-layered extension of the base algorithm reaches 82.0% accuracy (Coates & Ng, 2011). To evaluate the quality of the representations learned by DCGANs for supervised tasks, we train on Imagenet-1k and then use the discriminator's convolutional features from all layers, maxpooling each layers representation to produce a 4 × 4 spatial grid. These features are then flattened and concatenated to form a 28672 dimensional vector and a regularized linear L2-SVM classifier is trained on top of them. This achieves 82.8% accuracy, outperforming all K-means based approaches. Notably, the discriminator has many less feature maps (512 in the highest layer) compared to K-means based techniques, but does result in a larger total feature vector size due to the many layers of 4 × 4 spatial locations. The performance of DCGANs is still less than that of Exemplar CNNs (Dosovitskiy et al., 2015), a technique which trains normal discriminative CNNs in an unsupervised fashion to differentiate between specifically chosen, aggressively augmented, exemplar samples from the source dataset. Further improvements could be made by finetuning the discriminator's representations, but we leave this for future work. Additionally, since our DCGAN was never trained on CIFAR-10 this experiment also demonstrates the domain robustness of the learned features.

Table no 1: MNIST classification results using our pre-trained model. Our DCGAN is not pretrained on MNIST, but on Imagenet-1k, and the features are used to classify MNIST images

Model	Accuracy	Accuracy(400 per class)	Max # of feature units
1 Layer k-means	80.6%	63.7% (±0.7%)	4800
3 Layer k-means Learned RF	82.0%	70.7% (±0.7%)	3200
View Invariant k-means	81.9%	72.6% (±0.7%)	6400
Exemplar CNN	84.3%	77.4% (±0.2%)	1024
DCGAN(ours)+L2SVM	82.8%	73.8% (±0.4%)	512

Classifying Svhn Digits using Gans as a Feature Extractor:

On the Street View House Numbers dataset (SVHN)(Netzer et al., 2011), we use the features of the discriminator of a DCGAN for supervised purposes when labeled data is scarce. Following similar dataset preparation rules as in the CIFAR-10 experiments, we split off a validation set of 10,000 examples from the non-extra set and use it for all hyperparameter and model selection. 1000 uniformly class distributed training examples are randomly selected and used to train a regularized linear L2-SVM classifier on top of the same feature extraction pipeline used for CIFAR-10.

This achieves state of the art (for classification using 1000 labels) at 22.48% test error, improving upon another modification of CNNs designed to leverage unlabeled data (Zhao et al., 2015). Additionally, we validate that the CNN architecture used in DCGAN is not the key contributing factor of the model's performance by training a purely supervised CNN with the same architecture on the same data and optimizing this model via random search over 64 hyperparameter trials (Bergstra&Bengio, 2012). It achieves a significantly higher 28.87% validation error.

VIII. Investigating and Visualizing the Internals of the Networks

We investigate the trained generators and discriminators in a variety of ways. We do not do any kind of nearest neighbor search on the training set. Nearest neighbors in pixel or feature space are trivially fooled by small image transforms. We also do not use log-likelihood metrics to quantitatively assess the model, as it is a poor metric.

Table no 2: SVHN classification with 1000 labels

Model	Error Rate
KNN	77.93%
TSVM	66.55%
M1+KNN	65.63%
M1+TSVM	54.33%
M1+M2	36.02%
SWWAE without dropout	27.83%
SWWAE with dropout	23.56%
DCGAN(ours)+L2-SVM	22.48%
Supervised CNN with same Architechure	28.87%

Walking in the Latent Space: The first experiment we did was to understand the landscape of the latent space. Walking on the manifold that is learnt can usually tell us about signs of memorization (if there are sharp transitions) and about the way in which the space is hierarchically collapsed. If walking in this latent space results in semantic changes to the image generations (such as objects being added and removed), we can reason that the model has learned relevant and interesting representations. The results are shown in Fig.

Visualizing the Discriminator Features: Previous work has demonstrated that supervised training of CNNs on large image datasets results in very powerful learned features (Zeiler& Fergus, 2014). Additionally, supervised CNNs trained on scene classification learn object detectors (Oquab et al., 2014). We demonstrate that an unsupervised DCGAN trained on a large image dataset can also learn a hierarchy of features that are interesting. Using guided backpropagation as proposed by (Springenberg et al., 2014), we show in Fig.5 that the features learnt by the discriminator activate on typical parts of a bedroom, like beds and windows. For comparison, in the same figure, we give a baseline for randomly initialized features that are not activated on anything that is semantically relevant or interesting.

IX. EVALUATING DCGANS CAPABILITY TO CAPTURE DATA DISTRIBUTIONS

We propose to apply standard classification metrics to a conditional version of our model, evaluating the conditional distributions learned. We trained a DCGAN on MNIST (splitting off a 10K validation set) as well as a permutation invariant GAN baseline and evaluated the models using a nearest neighbor classifier comparing real data to a set of generated conditional samples. We found that removing the scale and bias parameters from batchnorm produced better results for both models. We speculate that the noise introduced by batchnorm helps the generative models to better explore and generate from the underlying data distribution. The results are shown in Table 3 which compares our models with other techniques.

The DCGAN model achieves the same test error as a nearest neighbor classifier fitted on the training dataset - suggesting the DCGAN model has done a superb job at modeling the conditional distributions of this dataset. At one million samples per class, the DCGAN model outperforms InfiMNIST a hand developed data augmentation pipeline which uses translations and elastic deformations of training examples. The DCGAN is competitive with a probabilistic generative data augmentation technique utilizing learned per class transformations (Hauberg et al., 2015) while being more general as it directly models the data instead of transformations of the data.

Table no 3: Nearest neighbor classification results.

Model	Test Error(50k samples)	Test Error(10M samples)
AlignMNIST	-	1.4%
InfiMNIST	-	2.6%
Realdata	3.1%	-
GAN	6.28%	5.65%
DCGAN(ours)	2.98%	1.4%

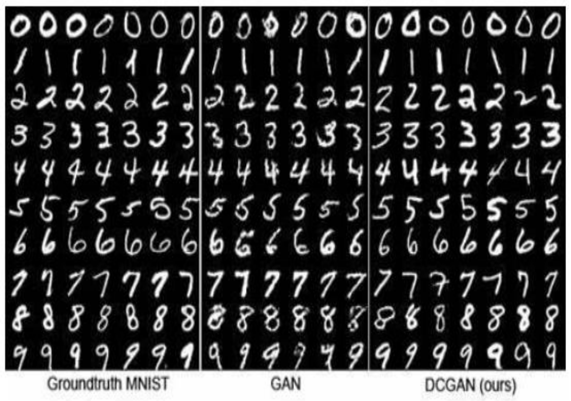

Figure no 5: Illustrations of MNIST dataset

X. Result

The digits produced image for MNIST dataset were:

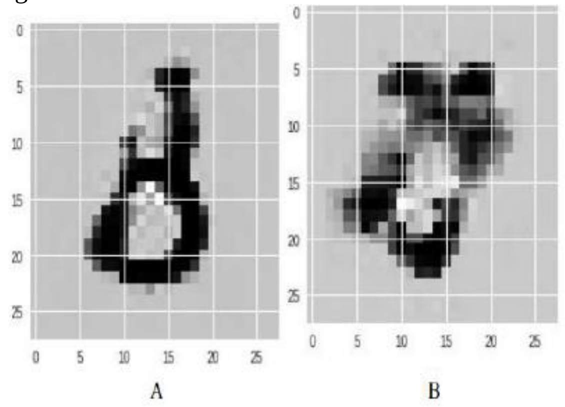

Figure no 6: Images generated

Image generated from our GAN Model.

(A) generated 8.

(B) generated 7(inverted 7).

Loss in Training:

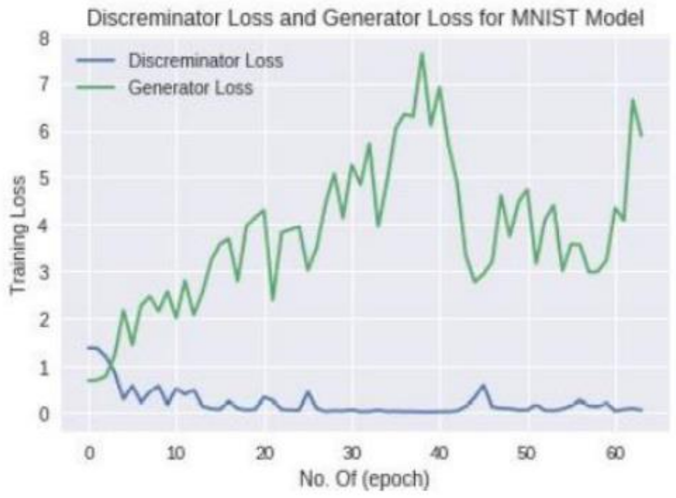

Figure no 7: Discrimonator and Generator loss

XI. Conclusion

Sampling from a latent space of images to create entirely new images or edit existing ones is currently the most popular and successful application of creative AI . In this paper, we demonstrated a way to generate images from training the existing similar images. GAN is a dynamic system where the optimization process is seeking not a minimum, but an equilibrium between two forces. It was difficult to train the generated image and they were not as good as the real image. Hence we had to use a pre-trained model for MNIST Dataset.

This work propose a more stable set of architectures for training generative adversarial networks and will give evidence that adversarial networks learn good representations of images for supervised learning and generative modeling. There are still some forms of model instability remaining noticed as models are trained longer they sometimes collapse a subset of filters to a single oscillating mode. Further work is needed to tackle this from of instability.

We think that extending this framework to other domains such as video (for frame prediction) and audio (pre-trained features for speech synthesis) should be very interesting. Further investigations into the properties of the learnt latent space would be interesting as well.

References

[1]. Understanding Generative Adversarial Networks (GANs). (2019). Retrieved from https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

[2]. Chollet, F. Deep learning with Python.

[3]. Radford, A., Metz, L., &Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ICLR 2016.

[4]. Denton, Emily, Chintala, Soumith, Szlam, Arthur, and Fergus, Rob. Deep generative image models using a laplacian pyramid of adversarial networks. arXiv preprint arXiv:1506.05751, 2015.

[5]. Dosovitskiy, Alexey, Springenberg, Jost Tobias, and Brox, Thomas. Learning to generate chairs with convolutional neural networks. arXiv preprint arXiv:1411.5928, 2014.

[6]. Danilo Jimenez Rezende, Shakir Mohamed, and DaanWierstra, “Stochastic Backpropagation and Approximate Inference in Deep Generative Models,” arXiv (2014).

[7]. Diederik P. Kingma and Max Welling, “Auto-Encoding Variational Bayes, arXiv (2013), https://arxiv.org/ abs/1312.6114.

[8]. Bergstra, James and Bengio, Yoshua. Random search for hyper-parameter optimization. JMLR, 2012.

[9]. Coates, Adam and Ng, Andrew Y. Learning feature representations with k-means. In Neural Networks: Tricks of the Trade, pp. 561–580. Springer, 2012.

[10]. Coates, Adam and Ng, Andrew. Selecting receptive fields in deep networks. NIPS, 2011.

[11]. Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 248–255. IEEE, 2009.