**SOUTHEAST MISSOURI STATE UNIVERSITY-1873**



**PROJECT TITLE**

UNSUPERVISED GENERATIVE ADVERSARIAL NETWORKS USING DEEP CONVOLUTIONAL

**SEMESTER:** FALL 2023

**COURSE:** CS505-01DATA MINING

**INSTRUCTOR: MADHU PULLELA**

**GitHub Link:** [**https://github.com/abedin-ashraf/DCGAN-CS505-**](https://github.com/abedin-ashraf/DCGAN-CS505-)

**Group Project Report by,**

Mohammed Nurul Abedin Ashraf

Sheshanth Reddy Ananthula

Krishna Sai Bezavada

Saikrishna Karnati

Sindhu Mandalapu

**ABSTRACT**

Convolutional networks (CNNs) have recently been widely used in computer vision applications for supervised learning. Unsupervised learning using CNNs has comparatively gotten less attention. In this project, we want to close the gap between CNN success in supervised learning and unsupervised learning. Deep convolutional generative adversarial networks (DCGANs), a subclass of CNNs with specific architectural restrictions, are introduced, showing that they are a promise for unsupervised learning. Our deep convolutional adversarial pair learns a hierarchy of representations from object pieces to scenes in both the generator and discriminator, as demonstrated by training on various assets. We showed their usefulness as general image representations by applying them to new challenges.

**1. INTRODUCTION**

It has been a focus of active research to learn reusable feature representations from big unlabeled datasets. In computer vision, one can take advantage of the infinite number of unlabeled photos and videos to develop effective intermediate models, which can be applied to various supervised learning tasks like image classification. One method for creating effective image representations is to train generative adversarial networks and then reuse components of the discriminator and generator networks as feature extractors for supervised tasks. GANs offer a compelling alternative to maximum likelihood methods. One may also contend that representation learning is appealing due to its learning method and the absence of a heuristic cost function. GANs have a reputation for being difficult to train, frequently resulting in generators that create absurd outputs. Little published research has attempted to comprehend and display what multi-layer GANs learn and their intermediate representations.

We provide the following contributions to this work.

* Convolutional GANs are stable to train in most situations thanks to a set of restrictions that we suggest and analyze for their architectural topology. We refer to this group of structures as Deep Convolutional GANs (DCGAN)
* We use the taught discriminators to picture classification tasks, outperforming previous unsupervised algorithms in performance.
* We demonstrate empirically how certain filters have learned to draw objects by visualizing the filters that GANs have learned.
* We demonstrate how the generators' intriguing vector arithmetic properties enable simple manipulation of various generated samples' semantic properties.

**2. RELATED WORKS**

**2.1 REPRESENTATION LEARNING FROM UNLABELED DATA**

Unsupervised representation learning has received much attention in general computer vision and image-related research. Using clustering on the data (for instance, using K-means) and utilizing the clusters for better classification results is a traditional method of unsupervised representation learning. Hierarchical clustering of picture patches in the context of images can be used to learn potent image representations. Another common technique is to train auto-encoders that compress an image into a small amount of data and decode the data to rebuild the image as correctly as possible. It has also been demonstrated that these techniques can extract useful feature representations from image pixels. It has also been shown that deep belief networks perform well while learning hierarchical words.

**2.2 GENERATING NATURAL IMAGES**

Two types of generative image models have been thoroughly studied: parametric and nonparametric. The non-parametric models have been employed in texture generation, super-resolution, and in-painting. They frequently match from a database of existing images, matching patches of images.

Much research has been done on parametric models for creating images, for instance, MNIST digits or texture generation. However, until recently, there was little success in making realistic representations of the outside world. Although a variational sampling method for creating images has had some success, the samples frequently suffer from blurriness. An alternative approach uses an iterative forward diffusion technique to create images. Images produced using Generative Adversarial Networks suffered from noise and illegibility. Although a laplacian pyramid adaptation to this method produced higher-quality images, the objects still appeared shaky due to noise caused by chaining numerous models. Other methods for creating natural images have recently shown some promise, including recurrent networks and deconvolution networks.

**2.3 VISUALIZING THE INTERNALS OF CNNS**

The use of neural networks has frequently come under fire for being "black-box" techniques with no explanation of what the networks accomplish in the form of an easy-to-understand algorithm. In the past, it has been demonstrated that one might determine the general function of each convolution filter in a CNN by utilizing deconvolutions and filtering the maximal activations. Similarly, we can examine the ideal image that activates specific groups of filters by applying gradient descent to the inputs.

**3. APPROACH AND MODEL ARCHITECTURE**

There have been previous unsuccessful attempts to scale up GANs using CNNs to model images. To train deeper and better resolution, generative models, we discovered a family of architectures following significant model exploration that produced robust training across various datasets. Three recently shown modifications to CNN architectures are at the heart of our strategy, which we have adopted and modified.

First, there is the all-convolutional net, which uses stridden convolutions in place of deterministic spatial pooling functions (like max-pooling) to enable the network to learn its spatial downsampling. This strategy is used in our generator to let it develop its spatial upsampling and discriminator.

The second tendency is the removal of fully connected layers over convolutional features. The most effective illustration of this is global average pooling, used in innovative image classification algorithms. We discovered that pooling global averages improved model stability but slowed convergence. Directly connecting the highest convolutional features to the generator's and discriminator's input and output worked well as a compromise. As it is essentially a matrix multiplication, the first layer of the GAN, which accepts a uniform noise distribution Z as input, may be referred to as fully connected. However, the result is reshaped into a 3-dimensional tensor and employed as the foundation of the convolution stack. The final convolution layer is flattened and fed into a single sigmoid output for the discriminator. An illustration of a sample model architecture can be found in.

The third method, batch normalization, stabilizes learning by adjusting the input to each unit's mean and variance to zero. As a result, gradient flow is improved in deeper models, which aids in solving training issues brought on by subpar initialization. Preventing the generator from collapsing all samples into a single point, a typical failure scenario seen in GANs, proved crucial in enabling deep generators to start learning. However, sample oscillation and model instability occurred when the batch norm was applied directly to all layers. Not using batch norm on the discriminator input layer and the generator output layer prevented this.

The generator uses ReLU activation, except for the output layer, which employs the Tanh function. We found that bounded activation helped the model learn to saturate and cover the training distribution's color space more quickly. We discovered that the leaky corrected activation in the discriminator performs well, particularly for higher-resolution modeling, unlike the original GAN paper, which employed the maxout activation.

Stride (discriminator) and fractional strived (discriminator) convolutions should be used in place of any pooling layers (generator).

* Apply batch norm to the discriminator and generator.
* For deeper structures, remove connected entirely hidden layers.
* Utilize Tanh activation in the output layer alone; all other layers should use ReLU activation.
* For all layers, use LeakyReLU activation in the discriminator.

**4. DETAILS OF ADVERSARIAL TRAINING**

Diagram

Description automatically generatedWe used the Large-scale Scene Understanding Network (LSUN), Imagenet-1k, and a recently created Faces dataset to train DCGANs. The use of each of these datasets is described in more detail below. The only pre-processing done to the training images was scaling them to fit the tanh activation function's [-1, 1] range. All models were trained with a mini-batch size of 32 for stochastic gradient descent (SGD). All weights were calculated using a normal distribution with a zero center and a standard deviation of 0.2. The slope of the leak in the LeakyReLU was fixed to 0.2 in all models. The Adam optimizer with customized hyperparameters was employed in place of earlier GAN work's usage of momentum to quicken training. The suggested learning rate of 0.001 was too high; we settled on 0.0002. In addition, we discovered that dropping the momentum term one at the

In the figure, the DCGAN generator is used to represent the LSUN scenario. A modest spatial extent convolutional representation with several feature maps is projected from a 100-dimensional uniform distribution Z. Following that, this high-level representation is transformed into a 64x64 pixel image using a sequence of four fractionally-strided convolutions (in some recent works, these are incorrectly referred to as deconvolutions). It is noteworthy that neither pooling nor fully connected layers are employed. While lowering it to 0.5 helped stabilize training, the advised value of 0.9 caused oscillation and instability in training.

For forward convolution, we are going from right to left at the same time, we must know the pixel value at this location, which is a vector, and each pixel is a vector, which will have a dimension of 1024. If we take a window that is five by five, or it could be three by three, you have, in this case, nine matrices, and then you take your x and where the stride is going to come in here because then you are jumping from one portion of your image to another part as it jumping by a step sizes of two. So, this is a forward convolution. It is a three-by-three convolution. Now we want to take its derivative if we want to know the result of loss with respect to x and how it relates to the product of loss with respect to y.

**4.1 ADDING CONVOLUTIONAL LAYERS TO GANS.**

The primary idea of the DCGANS compared to the original generative adversarial network models, such as the multi-layer perception again, is that it adds up sampling convolutional layers between the input vector Z and the output image in t generator in addition to the discriminator it uses convolutional layers like a regular convolutional neural network to classify the generated and natural images as the corresponding label real effect. So the inspiration that the authors had for the DCGANS around the time of this publication is the all convolutional net and eliminating fully connected layers on the top of convolutional layers so the idea was that instead of having these operations like max pooling and average pooling which would like these 2x2 kernels that just kind of like group the statistics of pixels together and decretes decrease the spatial resolution by a factor 2 so like a 32 by 32 image is processed into a 16 by 16 through these pooling functions these are really popular foundation the convolutional networks there and paper’s like the le net and the alex net so another idea is batch normalization and batch normalization takes a vector features or it could be a you know a matrix and it normalizes them so that they have some the same parameter mean and another parameter for the standard deviation so just like a standard multivariate gaussian is how the features in the layers of a neural network are distributed another idea that they use is they build on the leaky ReLu activation with the leaky ReLu which has negative slope.

**4.1.1 LSUN**

As the visual quality of samples from generative picture models has increased, worries about over-fitting and memory of training examples have increased. We train a model using the LSUN human face dataset, which contains over 3 million training examples, to show how our technique scales with more data and higher resolution generation. Recent research has demonstrated a direct correlation between a model's generalization ability and how quickly it learns. To show that our model needs to produce high-quality samples by simply overfitting/memorizing training examples, we present samples from one training epoch, simulating online learning and samples following convergence. The photos did not receive any data augmentation.

**4.2 FACES**

Chart

Description automatically generatedWe extracted pictures with human faces from random web image searches for the names of people. The names of the individuals were taken from Kaggle with the condition that they had modern birth dates. 10K individuals contributed 202k. Photos to this dataset. These photos are subjected to an OpenCV face detector, which yields 10k unique identities. We keep the detections that are sufficiently high resolution. We practice using these face boxes. The photos did not have any data enhancement. The images in this database are 178x213 resolution, but they are scaled down to 64x64 resolution for easiness of training.

**5 CONCLUSIONS**

We suggest a more stable set of topologies for training generative adversarial networks. We prove that these networks can learn accurate representations of pictures for both supervised learning and generative modeling. We found that as models are trained for longer, a subset of filters occasionally collapses into a single oscillating mode.

Unfortunately, we could only run the Program for up to 20 epochs; ideally, running 100 epochs would yield good results, but we couldn't output good results due to hardware and time restraints. Also, since we only ran 20 epochs, we could not determine perfect loss functions for the generator and discriminator since, at a lower epoch, the losses will be high at the generator and higher success at the discriminator. In these 20 epochs, we have generated. 1280 images.

A collage of a person's face

Description automatically generated with medium confidence

Chart, histogram

Description automatically generated We have uploaded our results to the GitHub link, and the images can be found there. As per the losses, we are putting our predicted losses as if the model ran for almost 100 epochs below.

**6 FUTURE WORK**

Progressive Growing GAN, also known as ProGAN, is an extension of the GAN training procedure that enables the generator models to train steadily and produce huge, high-quality images. It was developed by Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen of NVIDIA. It entails training by beginning with a small image and adding blocks of layers incrementally so that the generator model's output size and the discriminator model's input size increase until the required image size is attained. This kind of strategy has shown to be quite effective at producing realistic synthetic images of the highest quality.

* Progressive growing (of model and layers)
* Minibatch std on Discriminator
* Normalization with PixelNorm
* Equalized Learning Rate
* Tracking and visualizing metrics such as loss and accuracy
* Visualizing the model graph (ops and layers)
* Viewing histograms of weights, biases, or other tensors as they change over time
* Projecting embeddings to a lower-dimensional space
* Displaying images, text, and audio data

**ACKNOWLEDGMENTS**

We are fortunate and appreciative of all the suggestions and help we have gotten throughout this project, particularly from Ian Goodfellow, Tobias Springenberg, Arthur Szlam, and Durk Kingma. Additionally, we would like to thank everyone at Indico for their assistance, dialogue, and support, especially Dan Kuster and Nathan Lintz, the other two members of the Indico research team. Finally, we thank Nvidia for providing a Titan-X GPU for this project.

**REFERENCES**

* Bergstra, James and Bengio, Yoshua. Random search for hyper-parameter optimization. JMLR, 2012.
* Coates, Adam and Ng, Andrew. Selecting receptive fields in deep networks. NIPS, 2011.
* Coates, Adam and Ng, Andrew Y. Learning feature representations with k-means. In Neural Networks: Tricks of the Trade, pp. 561–580. Springer, 2012.
* 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.