DCGANs: Unsupervised Representational Learning

Rahul Singh
Khoury College of Computer Science
Northeastern University
Boston, MA
singh.rahu@northeastern.edu

Ankit Kshatriya
Khoury College of Computer Science
Northeastern University
Boston, MA
kshatriya.a@northeastern.edu

1 Paper

For this project we will be replicating: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (2015) paper by Alec Radford, Luke Metz and Soumith Chintala.[1]

2 Context

The authors of the paper introduce a new class of CNNs for unsupervised learning with four architectural constraints. These constraints were based upon the changes introduced to CNN architecture at the time of inception of the paper. Following were the ideas/constraints included for implementation of DCGANs:

- All convolutional net (Springenberg et al., 2014)[2] which replace deterministic spatial pooling functions (such as maxpooling) with strided convolutions, allowing the network to learn their own spatial downsampling. This approach was used in the generator, allowing it to learn its own spatial upsampling, and discriminator.
- Eliminating fully connected layers on top of convolutional features. The authors used global average pooling, based on then state of the art image classification models (Mordvintsev et al.)[3] ,which increased model stability but hurt convergence speed. As a middle ground of directly connecting the highest convolutional features to the input and output respectively of the generator and discriminator worked well for their implementation. 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.
- Batch Normalization (Ioffe & Szegedy, 2015)[4] which stabilizes learning by normalizing the input to each unit to have zero mean and unit variance. This helped to deal with training problems that arise due to poor initialization and helped 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)[5] was used in the generator with the exception of the output layer which uses the Tanh function. The authors 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 the authors' found the leaky rectified activation (Maas et al., 2013)[6] (Xu et al., 2015)[7] 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)[8].

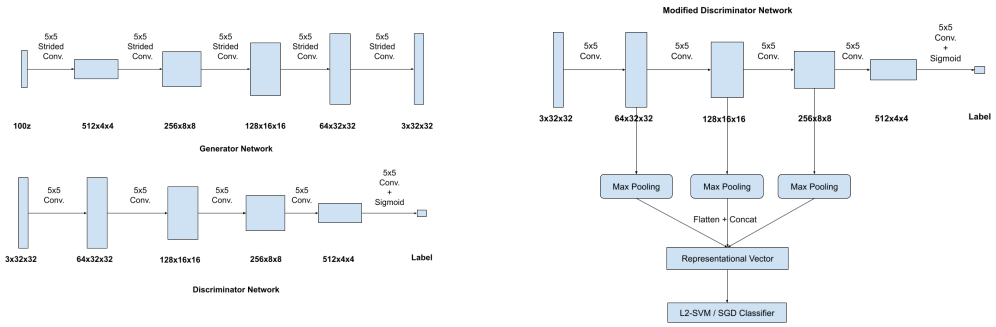
3 Results to be Replicated

The authors implemented a classifier on CIFAR-10 that used a DCGAN trained on ImageNet for feature extraction. Below are the results of that experiment, along with results from the standard methods for unsupervised classification. We try to replicate the results achieved by the authors with a DCGAN based classifier, but using different datasets and classifiers.

Model Accuracy	Accuracy	(400 per class)	max # of features 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 (author) + L2-SVM	82.8%	73.8% (±0.4%)	512

Table 1: Benchmark results

4 Implementation Details



(a) DCGAN (b) DCGAN + Classifier

Figure 1: DCGAN Architecture

Hyperparameter	**Value**
Learning Rate	0.0002
Optimizer	Adam ($\beta_1 = 0.5, \beta_2 = 0.999$)
Input Batch Size	128
Weights(initialize)	Normal Distribution (μ=0, σ=0.02)
LeakyReLU (slope)	0.2
Feature map size (generator)	64
Feature map size (discriminator)	64
Epochs	10
Loss-function	BCELoss

Table 2: Model Hyperparameters

Using the original ImageNet-1k dataset was prohibitively large for us, hence we used the following datasets instead to train our networks:
1. CelebA (178X218) (200K Images)
2. ImageNet64 (Downsampled 64x64) (1.2M Images)
3. TinyImageNet (Stanford) (Downsampled 64x64) (100K Images) (used for prototyping)

Our implementation focused on using GANs as feature extractors for classifying CIFAR-10 dataset. For this we used the same hyper-parameters as suggested by the authors in the paper, shown in table 2. We've also used the same image pre-processing steps used originally in the paper. We take the 32x32 center-crop of each input image while training the DCGAN. Additionally, the pixels are scaled in range of $[-1, 1]$. We trained the GANs separately for each dataset.

To evaluate the quality of the representations learned by DCGANs for unsupervised tasks, we train the DCGAN on the given dataset, and then use the convolutional features from the trained discriminator. For all but the last Conv2D layer, we max-pool each layer's output. For max pooling, we used a kernel of size 4 with a stride of 2. The 3-d output of the pooling process is flattened to a 1-d vector. We then concatenate the flattened outputs from each of the layers to form a 28672 dimensional vector.

Lastly, we trained a regularized linear L2-SVM and an SGDClassifier separately on the CIFAR-10 dataset. As part of the training process, each input image (from CIFAR-10) is passed through the discriminator, which returns the 1-d vector representation of the image. This input, followed by appropriate scaling is then fed into the linear classifiers.

The original paper only uses a linear L2-SVM for classification. However, SVMs pose difficulty when optimizing over large input sizes. Given the large size of each input representation, SVMs had poor convergence characteristics, along with an unreasonable memory requirement. We therefore trained SGDClassifier, using the same loss function as the L2-SVM. This approach allowed us to train the classifier in batches, and also improved accuracy while reducing training time and memory requirements.

Lastly, we tried to generate the feature activation maps using the Generational Backprop method in order to visualize the final convolution layers of the discriminator, but were unable to complete it in time. We also tried Gradient-weighted Class Activation Heatmap along with a preexisting library of FlashTorch, but all of them failed due to non conformity of our architecture to standard torch-vision network model.

5 Results

For the task of CIFAR-10 classification, we trained 2 variants of DCGAN. Both variants had the same model architecture, but differed in the kernel sizes in the convolutional layers along with associated padding/stride changes.In the results presented below, these models are identified as DCGAN_4x4 and DCGAN_5x5, which use kernel sizes of 4 and 5 respectively. Changing the kernel sizes had corresponding effects on the size of the image representation vector (used to train the classifiers), as well as model accuracy and overall GAN characteristics. Also, for each of the two datasets, we trained both the L2-SVM and the SGDClassifier. Additionally, the classifiers were trained with a limited sample of the training set using only 400 images per label. This naturally resulted in a lower classification accuracy. The various models and their corresponding accuracy is presented below.

Model	Classifier	Dataset	Accuracy	Accuracy (400 per class)
DCGAN_4x4	L2-SVM	CelebA	50.88%	49.32%
DCGAN_4x4	SGDClassifier	CelebA	56.65%	44.56%
DCGAN_4x4	L2-SVM	ImageNet64	58.2%	57.76%
DCGAN_4x4	SGDClassifier	ImageNet64	64.35%	52.53%
DCGAN_5x5	L2-SVM	CelebA	60.45%	59.28%
DCGAN_5x5	SGDClassifier	CelebA	62.53%	47.32%
DCGAN_5x5	L2-SVM	ImageNet64	64.12%	63.08%
DCGAN_5x5	SGDClassifier	ImageNet64	68.04%	53.42%

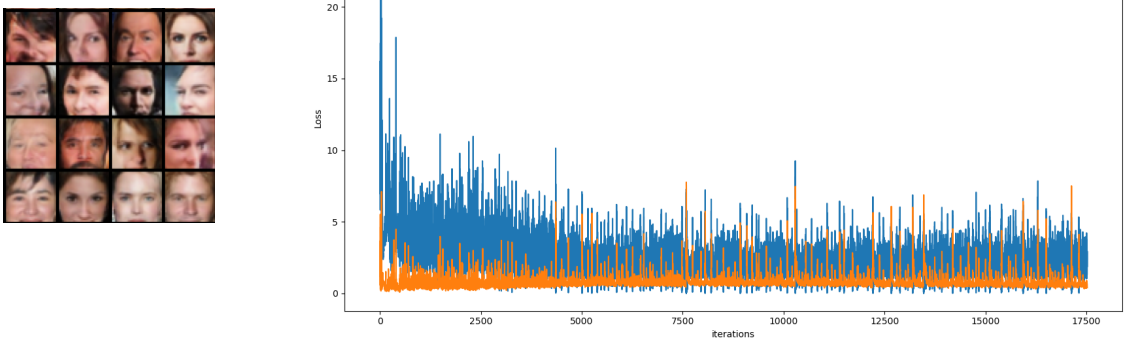


Figure 2: (Left) A sample of images generated by a trained Generator; (Right) The training loss curves for the Generator (orange) and the Discriminator (Blue)

6 Observations and future work

As part of this project, we implemented DCGANs to learn unsupervised representations for image classification task. While training the GANs, we observed some model instability while using 4x4 convolutional kernels in networks. Understanding this source of instability could provide deeper understanding of the inner workings of GANs in general. Additionally, due to constraints on computation power, we were unable to train the models on the original dataset used by the authors, nor were we able to tune hyperparameters extensively. Both of these could be the reason behind the lower classification accuracy achieved by us compared to the authors.

We therefore think that there's further exploration to be done in the areas of model stability, as well as in finding more suitable hyperparameters to improve the classification accuracy and to bring it at par with supervised learning approaches.

References

[1] Radford, Alec & Metz, Luke & Chintala, Soumith. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.

[2] Springenberg, Jost Tobias, Dosovitskiy, Alexey, Brox, Thomas, and Riedmiller, Martin. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806, 2014.

[3] Mordvintsev, Alexander, Olah, Christopher, and Tyka, Mike. Inceptionism : Going deeper into neural networks. http://googleresearch.blogspot.com/2015/06/ inceptionism-going-deeper-into-neural.html

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

[5] Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 807– 814, 2010.

[6] Maas, Andrew L, Hannun, Awni Y, and Ng, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In Proc. ICML, volume 30, 2013.

[7] Xu, Bing, Wang, Naiyan, Chen, Tianqi, and Li, Mu. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853, 2015.

[8] Goodfellow, Ian J, Warde-Farley, David, Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Maxout networks. arXiv preprint arXiv:1302.4389, 2013.