Computer Vision
Final Project - DCGAN

Name:	**NetIDs**:
Yash Sadhwani	ys2913
Abhinav Gupta	ag5799

Overview:
In this project, we implement a Deep Convolutional Generative Adversarial Network (DCGAN) for unsupervised learning. A DCGAN consists of two components: the generator and the discriminator. The generator is responsible for creating images sampled from random noise. The discriminator in turn tries to classify between the real and the fake images (i.e. images generated from the generator). We trained on different datasets like CelebA and LSUN databases including church, classroom, etc. Using a pretrained classification model, the top-5 labels of both the real and generated images are compared.

Approach:
The DCGANs were trained on two datasets namely Large Scale Scene Understanding and CelebA datasets.
The process involves the following steps:

1. Extract the image
2. Preprocess the image
   (a) Scaling image to the given dimension with aspect ratio maintained
   (b) Cropping the scaled image to the desired output dimension
   (c) Rotating half of the images about vertical axis
   (d) Scaling the images to the range of tanh activation function [-1,1]
3. Training the Discriminator:
   (a) Discriminator is trained with the real images (label = 1))
   (b) A random noise sampled from uniform/normal distribution (100x1x1) is passed through the generator to form an image of dimension 64x64 (or 128x128)
   (c) Discriminator is trained with the image generated above (label = 0)

4. Training the Generator:
   (a) Image is generated with a random noise sampled from uniform/normal distribution(100x1x1)
   (b) Image generated is passed as an input to the discriminator
   (c) Loss is calculated with respect to (label=1) and the corresponding gradient is calculated through the discriminator
   (d) Using the above calculated gradient, the generator is trained
5. Generated Image Validation
   (a) A Resnet-101 model pretrained on the ImageNet database for classification task with 1000 classes is used for predicting the top-5 labels for the real images.
   (b) Similarly we predict the top-5 labels for the generated images.
   (c) We compare both set of labels and find the set of common labels to calculate the accuracy.

Models(64x64 Image Size):

1. Generator:
   FullConvolution(input=100, feature maps=64*8, filter =4x4)
   BatchNormalization
   ReLU activation
   output= (64*8)x4x4
   FullConvolution(input=64*8, feature maps=64*4, filter =4x4,stride=2,padding=1)
   BatchNormalization
   ReLU activation
   output= (64*4)x8x8
   FullConvolution(input=64*4, feature maps=64*2, filter =4x4,stride=2,padding=1)
   BatchNormalization
   ReLU activation
   output= (64*2)x16x16
   FullConvolution(input=64*2, feature maps=64, filter =4x4,stride=2,padding=1)
   BatchNormalization
   ReLU activation
   output= (64*4)x32x32
   FullConvolution(input=64, feature maps=3, filter =4x4,stride=2,padding=1)
   Tanh activation
   output= (3)x64x64

2. Discriminator:
   input=3x64x64
   Convolution(input=3, feature maps=64, filter=4x4, stride=2,padding=1)
   LeakyReLU(.2) activation
   output=(64)x32x32
   Convolution(input=64, feature maps=64*2, filter=4x4, stride=2,padding=1)
   BatchNormalization
   LeakyReLU(.2) activation
   output=(64*2)x16x16
   Convolution(input=64*2, feature maps=64*4, filter=4x4, stride=2,padding=1)
   BatchNormalization
   LeakyReLU(.2) activation
   output=(64*4)x8x8
   Convolution(input=64*4, feature maps=64*8, filter=4x4, stride=2,padding=1)
   BatchNormalization
   LeakyReLU(.2) activation
   output=(64*8)x4x4
   Convolution(input=64*8, feature maps=1, filter=4x4, stride=2,padding=1)
   Sigmoid activation
   output=(1)

Running the code:

1. Downloading Data:

   (a) LSUN
   Download data from the given link:
   `https://github.com/fyu/lsun`

   (b) CelebA
   Download "img_align_celeba.zip" from
   `http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`

   (c) Pretrained Imagenet model:
   `https://d2j0dndfm35trm.cloudfront.net/resnet-101.t7`
   Place the downloaded file into 'test/labels/'

2. Unzip the data

3. Update the datatype in opts.lua (Possible values = 'lsun'/'celeb')
   If using lsun datatype, update the dataset field in opts.lua with the LSUN dataset name i.e. bridge, classroom

4. Update the path field with the data location (folder containing the unzipped folder) in opts.lua

5. For using gpu, update gpu field in opts.lua as true

6. Creating Generator and Discriminator models
   run the code: th main.lua
   The generator and discriminator models will be stored in the 'model' folder

7. In order to find the top-5 labels for the images produced by the trained generator model,
   The script is written to run on gpu with the associated libraries loaded on HPC with the corresponding path.
   run the code: ./generateimages.q <path_of_savedmodel>
   There are different options available inside the generate.lua file to produce different types of images such as noisemode, outputType, batchSize, etc.

8. In order to find the top-5 labels for the real images,
   run the code: th results.lua
   Different options can be set in the opts.lua file like which dataset to use, path of the dataset, etc.

9. Finally, for comparing the top-5 labels of both real and generated images, do: python compare.py

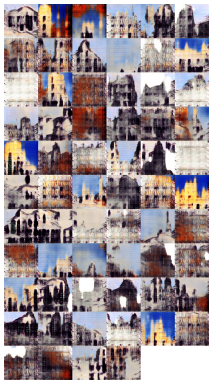
Experiments
Experiments were done by tweaking the hyper parameters namely Output Image Size,learning rate, Number of filters in the convolution layer

Dataset= Church

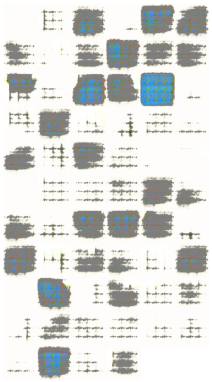
Top-5 labels generated from the trained imagenet model for the real images are:
'space shuttle', 'monastery', 'castle', 'bell cote, bell cot', 'church, church building'

1. Learning rate= .01, Filters=32, Image Size=128x128, Noise Distribution = normal, Noise Input=100x1x1

Generated images top-5 labels:
'theater curtain, theatre curtain', 'fire screen, fireguard', 'fountain', 'window screen', 'lampshade, lamp shade'

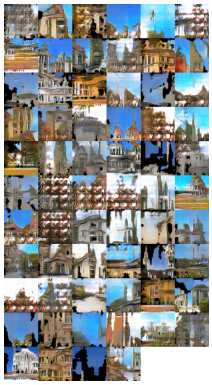


(a) 10 epochs

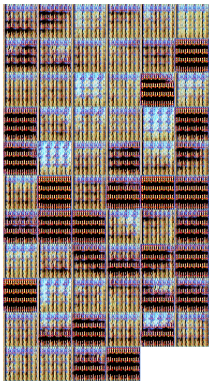


(b) 15 epochs

2. Learning rate= .01, Filters=64, Image Size=64x64, Noise Distribution = normal, Noise Input=100x1x1
   Generated images top-5 labels:
   'theater curtain, theatre curtain', 'safety pin', 'prayer rug, prayer mat', 'panpipe, pandean pipe, syrinx', 'book jacket, dust cover, dust jacket, dust wrapper'
   Following are the images:



(a) 15 epochs



(b) 20 epochs

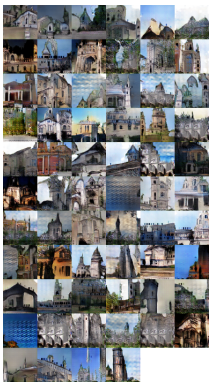
We see that the training failed maybe due to a higher learning rate.

3. Learning rate= .001, Filters=64, Image Size=64x64, Noise Distribution = normal, Noise Input=100x1x1

Generated images top-5 labels:
'space shuttle', 'monastery, 'castle', 'milk can', 'moving van'

(a) 15 epochs

(b) 20 epochs

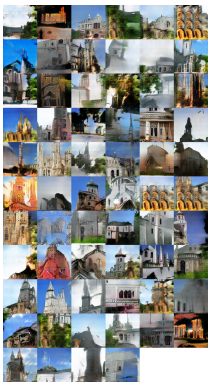


(c) 25 epochs

We see that the training progresses in the right direction but some images get blurred out after 25 epochs.

4. Learning rate= .0001, Filters=64, Image Size=128x128, Noise Distribution = normal, Noise Input=100x1x1

(a) 15 epochs



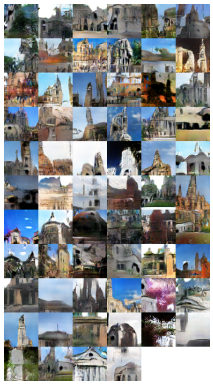
(b) 20 epochs

(c) 25 epochs

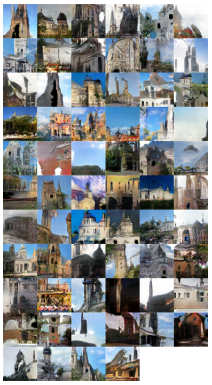
Generated images top-5 labels:
'monastery', 'castle', 'fountain', 'palace', 'church, church building'
We see that increasing the image size gives more fine images and the labels are also now close to the features of a church.

5. Learning rate= .0003, Filters=64, Image Size=64x64, Noise Distribution = uniform, Noise Input=100x1x1
   Generated images top-5 labels:
   'space shuttle', 'monastery', 'castle', 'thresher, thrasher, threshing machine', 'pedestal, plinth, footstall'

6. Learning rate= .0001, Filters=64, Image Size=64x64, Noise Distribution = normal, Noise Input=100x1x1
   Generated images top-5 labels:
   'space shuttle', 'monastery', 'castle', 'thresher, thrasher, threshing machine', 'pedestal, plinth, footstall'



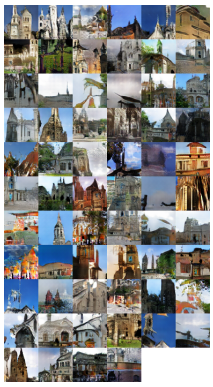
(a) 5 epochs



(b) 10 epochs

(c) 15 epochs

(d) 20 epochs

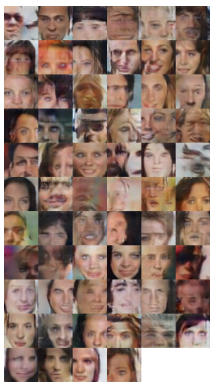


(e) 25 epochs

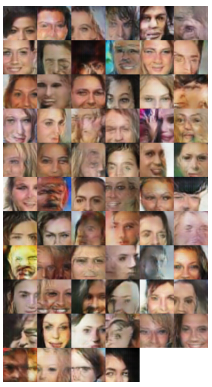
We can see that in the best model(Experiment 4) 4 out of 5 top labels get matched. The labels generated (like church, monastery, castle) also depict features that may be present in a church. The images also look very similar to the real images in the church dataset.

Dataset= Celeb
Experiment 4 in the Church dataset gave the best output images, so same hyper parameters were used for training Celeb dataset
Top-5 labels generated from the trained imagenet model for the real images are:
'brassiere, bra, bandeau', 'Windsor tie', 'suit, suit of clothes', 'neck brace', 'oboe, hautboy, hautbois'
Generated images top-5 labels are:
'bassoon', 'panpipe, pandean pipe, syrinx', 'oboe, hautboy, hautbois', 'neck brace', 'Windsor tie'

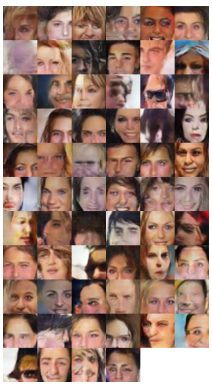
1. Learning rate= .0002, Filters=64, Image Size=64x64, Noise Distribution = normal, Noise Input=100x1x1
   Below are the images generated for selected epochs:



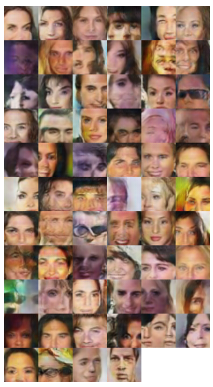
(a) 5 epochs



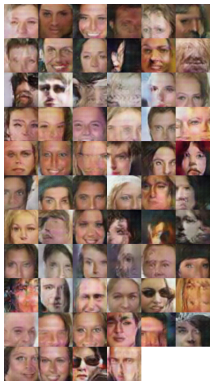
(b) 10 epochs



(c) 15 epochs



(d) 20 epochs



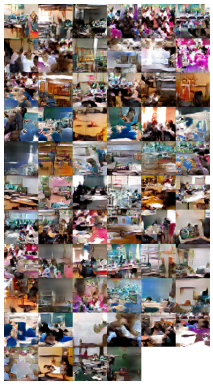
(e) 25 epochs

We can see that in the best model 3 out of 5 top labels get matched. The labels generated (like Windsor tie, neck brace) also depict features

that may be present in the celebA dataset. The images also look very similar to the real images in the celebA dataset.

Dataset= Classroom
Experiment 4 in the Church dataset gave the best output images, so same hyper parameters were used for training classroom dataset
Real images top-5 labels 'theater curtain, theatre curtain', 'screen, CRT screen', 'tobacco shop, tobacconist shop, tobacconist', 'comic book', 'home theater, home theatre'
Generated images top-5 labels:
'television, television system', 'toyshop', 'tobacco shop, tobacconist shop, tobacconist', 'neck brace', 'punching bag, punch bag, punching ball, punchball'

1. Learning rate= .0002, Filters=64, Image Size=64x64, Noise Distribution = normal, Noise Input=100x1x1
   Below are the images generated for selected epochs:



(a) 5 epochs

(b) 10 epochs

(c) 15 epochs

(d) 20 epochs

(e) 25 epochs

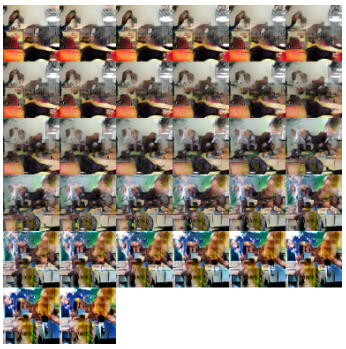
The labels generated (like television, toyshop) depict features that may

be present in a classroom but overall the labels do not depict features present in a classroom. The images generated are also not so clear which indicates that the training was not as successful in this dataset as compared to the above datasets.

Interpolation of noise
We interpolated noise between two points sampled from a uniform distribution for a total number of batchsize. This is done for all 3 datasets. We can see that the image gradually changes to an altogether different image.

(a) CelebA

(b) Church



(c) Classroom

Refrences:

1. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
   Alec Radford, Luke Metz, Soumith Chintala, 2015,ICLR

2. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks,

Emily Denton, Soumith Chintala, Arthur Szlam, Rob Fergus, 2015, NIPS

3. Identity Mappings in Deep Residual Networks
   Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, 2016, CVPR

4. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
   Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, Pieter Abbeel, 2016,arXiv:1606.03657

5. Unsupervised learning using Generative Adversarial Training and Clustering,
   Vittal Premachandran, Alan L. Yuille,2017,ICLR

6. We have used the code
   `https://github.com/facebook/fb.resnet.torch/tree/master/pretrained`
   for running the pretrained resnet model to classify the images.