An Exploration into Generative Models

Dipankar Ghosh | Simran Jumani | Kushagradhi Bhowmik | Sneha Shet | Nigel Flower

Traditional Autoencoder

- Form of unsupervised learning.
- Used to learn a latent representation of a given dataset.
- Traditionally used in denoising data or image compression.

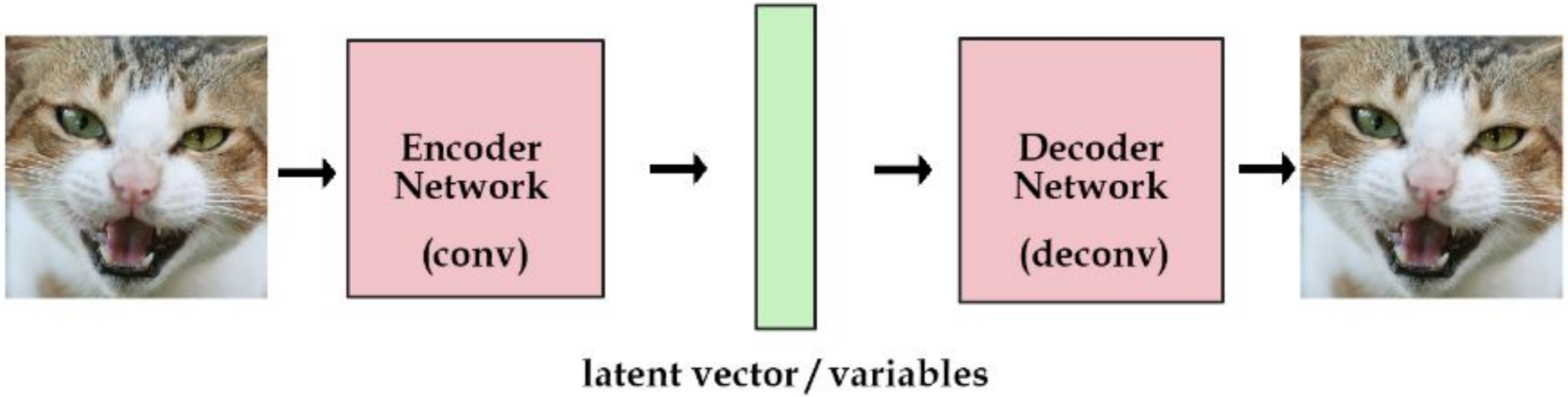


Image from: http://kvfrans.com/variational-autoencoders-explained/

VAE (Variational Autoencoder)[1]

- Uses a similar architecture to the traditional autoencoder.
- Change the latent vector layer (Kingma and Welling, 2013).
- Encoder takes images and converts them into unit gaussian encodings.
- Decoder network takes latent variables and deconvoles them into an image.
- Now we can generate images (or other data) along a latent space without getting garbage results.

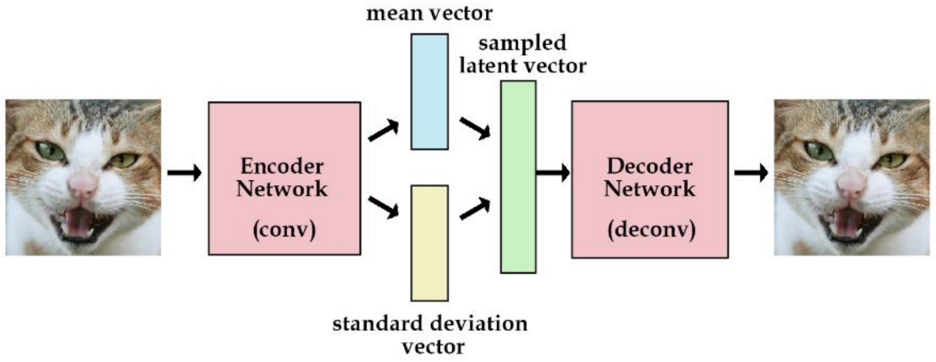
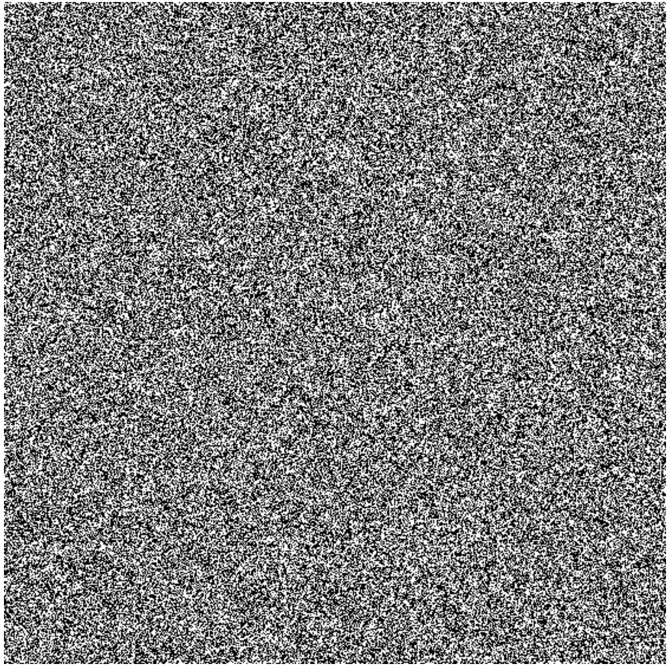
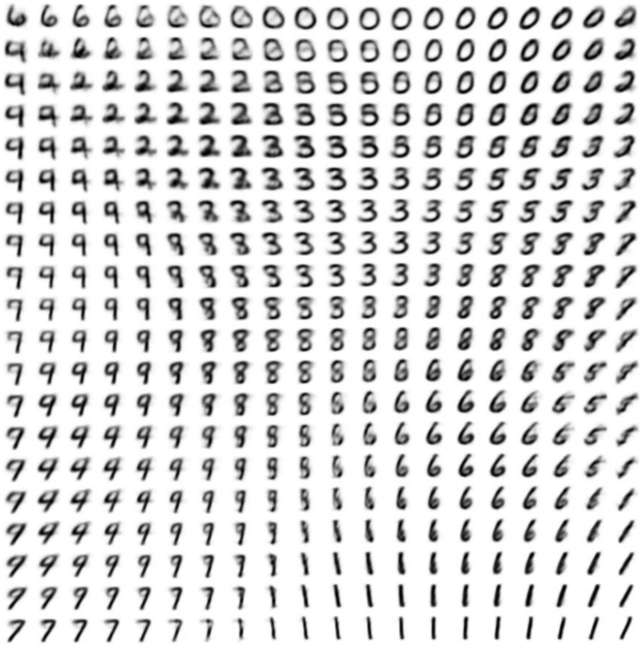


Image from: http://kvfrans.com/variational-autoencoders-explained/

Latent Space Learned by VAE

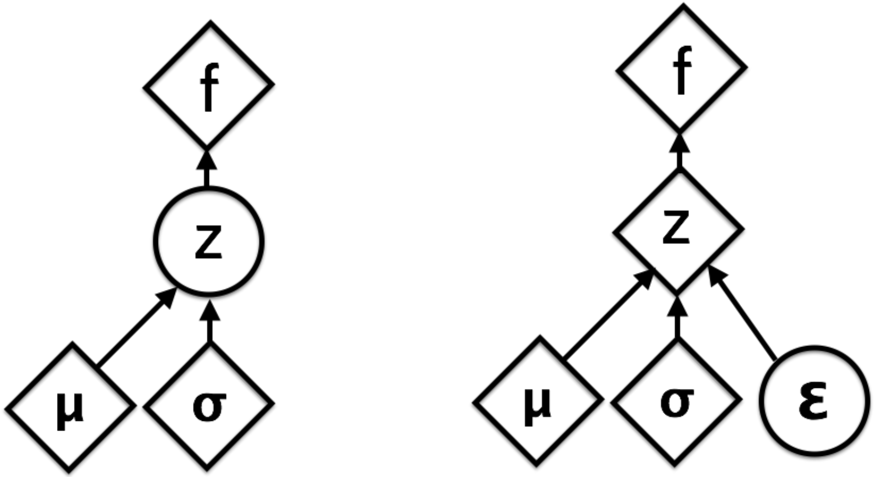




- GIF from https://jaan.io/what-is-variational-autoencoder-vae-tutorial/
- Image from https://ermongroup.github.io/cs228-notes/extras/vae/

VAE Learning

- Two loss functions: Generation Loss and Latent Loss.
- Generation Loss is MSE of generated image and training image.
- Latent Loss is KL divergence of latent vector and unit Gaussian.
- “Reparameterization trick” is used in training.

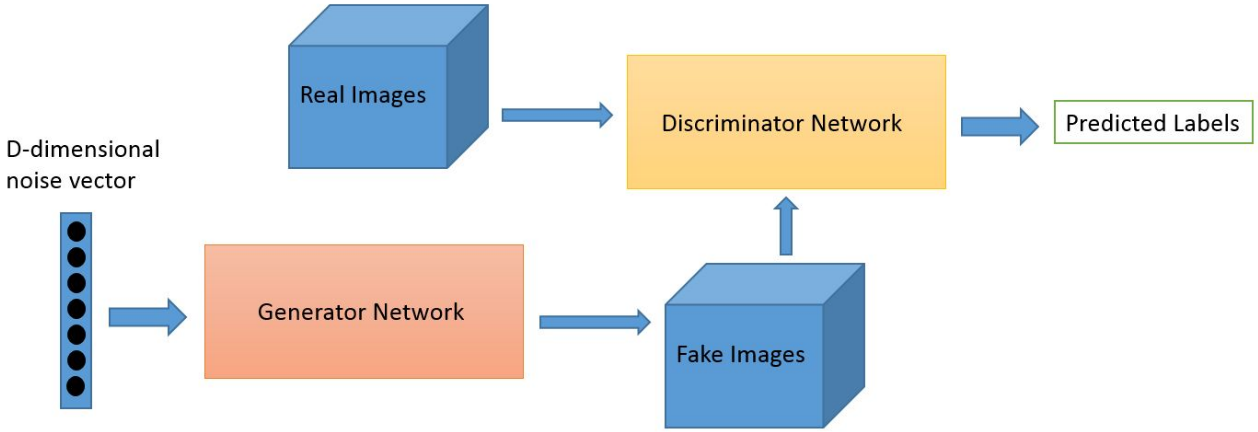


Original

Reparametrized

GAN (Generative Adversarial Network)

- Generative models (devised by Ian Goodfellow in 2014)
- Two differentiable functions (as neural networks) are locked in a game (adversarial process)
- Generative model G captures the data distribution
- Discriminative model D estimates the probability that a sample came from the training data rather than G.
- Training procedure for G is to maximize the probability of D making a mistake.



GAN Learning Mechanism

Generator & Discriminator is held constant while training the other. Training alternates between the two.

Training the Discriminator

- Simple binary classification task - given an image, predicts whether the image is real / artificial.
- Real images from train set, label → 1 Artificial images, label → 0, by passing a random vector to the Generator and getting its output. Binary Cross Entropy is used as the loss function and the weights are updated using Back-propagation.

Training the Generator

- Random vector passed to the Generator G as input
- Output (artificial image) of G passed via the Discriminator to make a prediction
- The label is taken as 1, since we want the Generator to produce such images for which the Discriminator would predict 1.
- If it doesn't predict 1, then the error (Binary Cross Entropy) is back-propagated and only the Generator weights are updated.

MNIST Dataset[1]

(1) "Gradient-based learning applied to document recognition." *Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner,* 1998

Sample from the Data set

VAE model architecture on MNIST

Encoder Network:

- Input Layer: 28 x 28 x 1
- Convolutional Layer: 32 filters, kernel size of 3, stride of 2, relu activation function
- Convolutional Layer: 64 filters, kernel size of 3, stride of 2, relu activation function
- Dense Layer: 2 x latent_dim

Generative Network:

- Input Layer: latent_dim number of neurons
- Dense Layer: 7 * 7 * 32
- Reshape into (7, 7, 32) channels
- Transposed Convolutional Layer: 64 filters, kernel of size 3, stride of 2, relu activation function
- Transposed Convolutional Layer: 32 filters, kernel of size 3, stride of 2, relu activation function
- Transposed Convolutional Layer: 1 filter, kernel of size 3, stride of 1, relu activation function

GAN model architecture on MNIST

Generator Model:

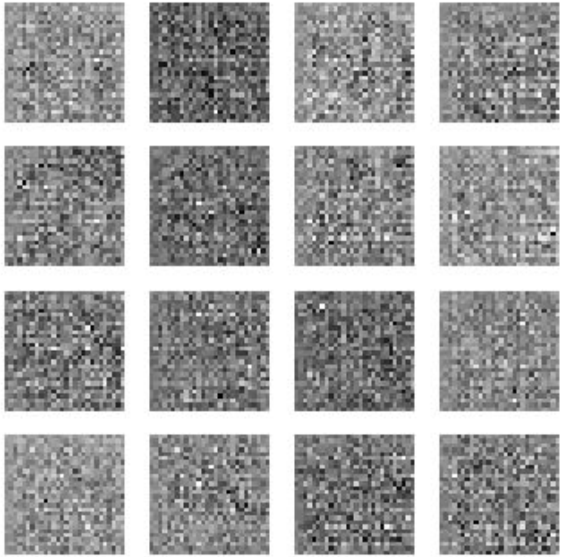
- Input Layer: 100
- Dense Layer: 7 * 7 * 256,, leaky relu activation function
- Transposed Convolutional Layer: 128 filters, kernel of size 5, stride of 1, leaky relu activation function
- Transposed Convolutional Layer: 64 filters, kernel of size 5, stride of 2, leaky relu activation function
- Transposed Convolutional Layer: 1 filter, kernel of size 5, stride of 2, tanh activation function

Discriminator Model:

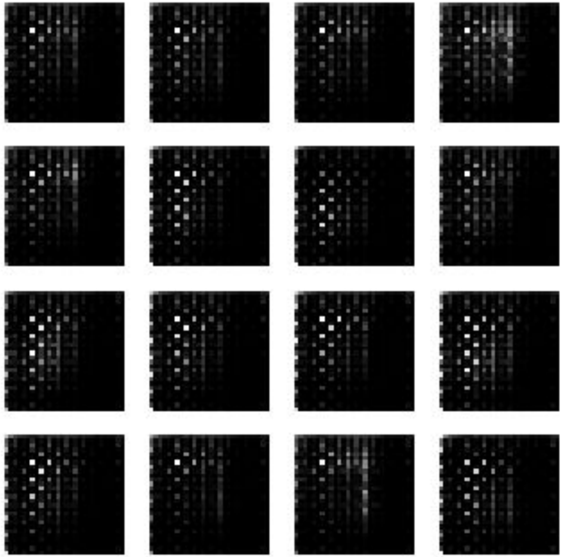
- Input Layer: 28 x 28 x 1
- Convolutional Layer: 64 filters, kernel of size 5, stride of 2, leaky relu activation function
- Dropout Layer: 0.3
- Convolutional Layer: 128 filters, kernel of size 5, stride of 2, leaky relu activation function
- Dropout Layer: 0.3
- Sigmoid output neuron

MNIST Results

VAE



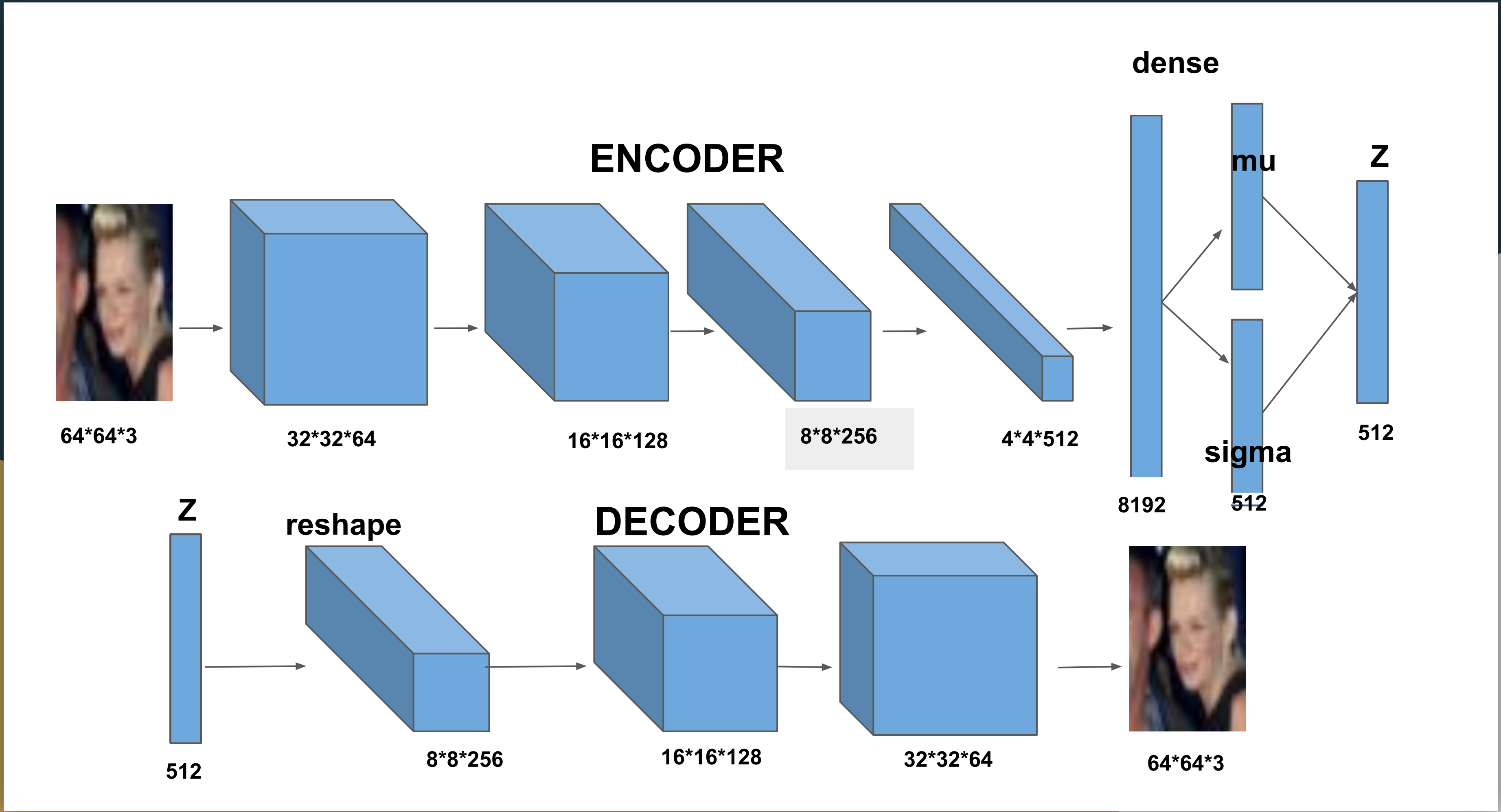
GAN



Labeled Faces in the Wild Dataset[1]

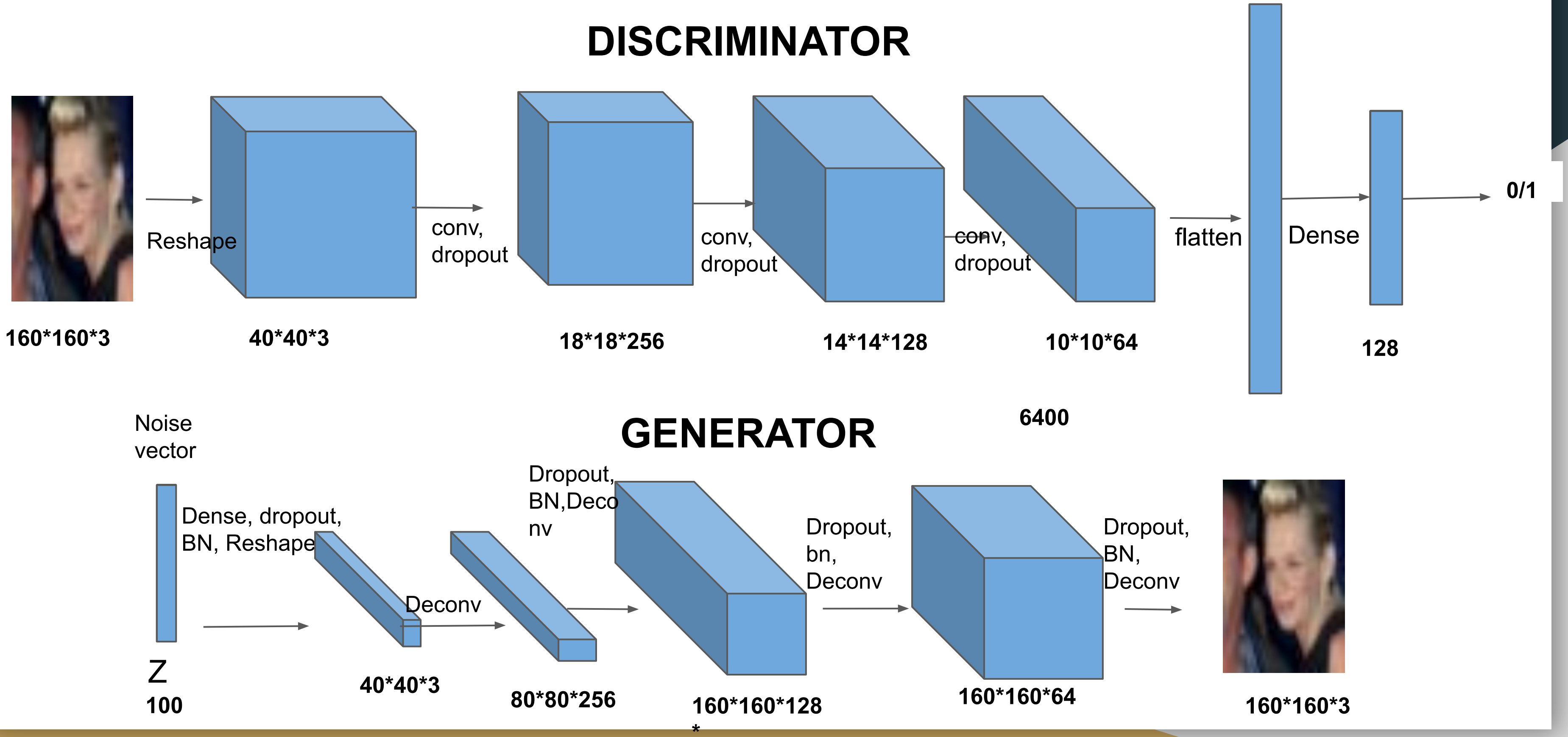
(1) "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments" *Huang G., Ramesh M., Berg T. Learned-Miller E.*, 2007

Sample from LFW Dataset



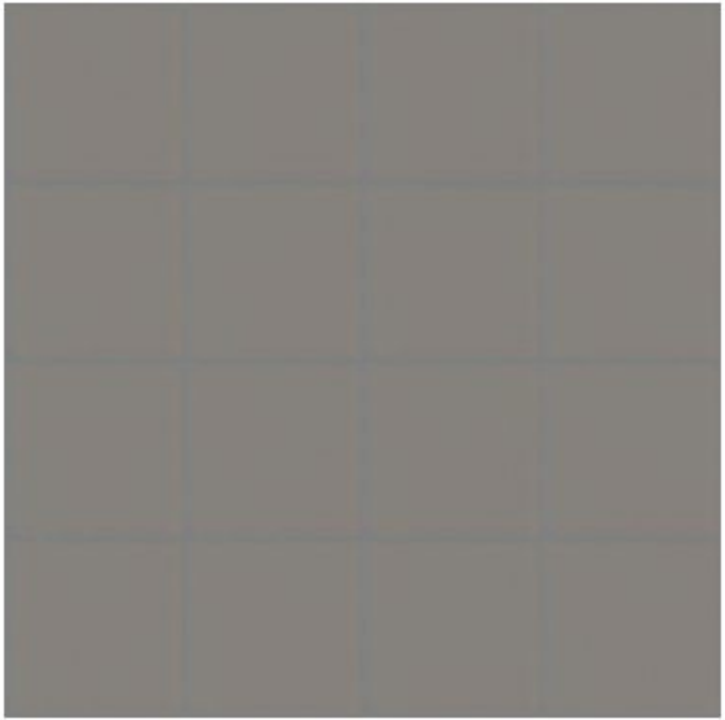
ENCODER
dense
mu
Z
64*64*3
32*32*64
16*16*128
8*8*256
4*4*512
sigma
512
8192
512
Z
reshape
DECODER
512
8*8*256
16*16*128
32*32*64
64*64*3

GAN model architecture on LFW



VAE- LFW Results

GAN- LFW Results

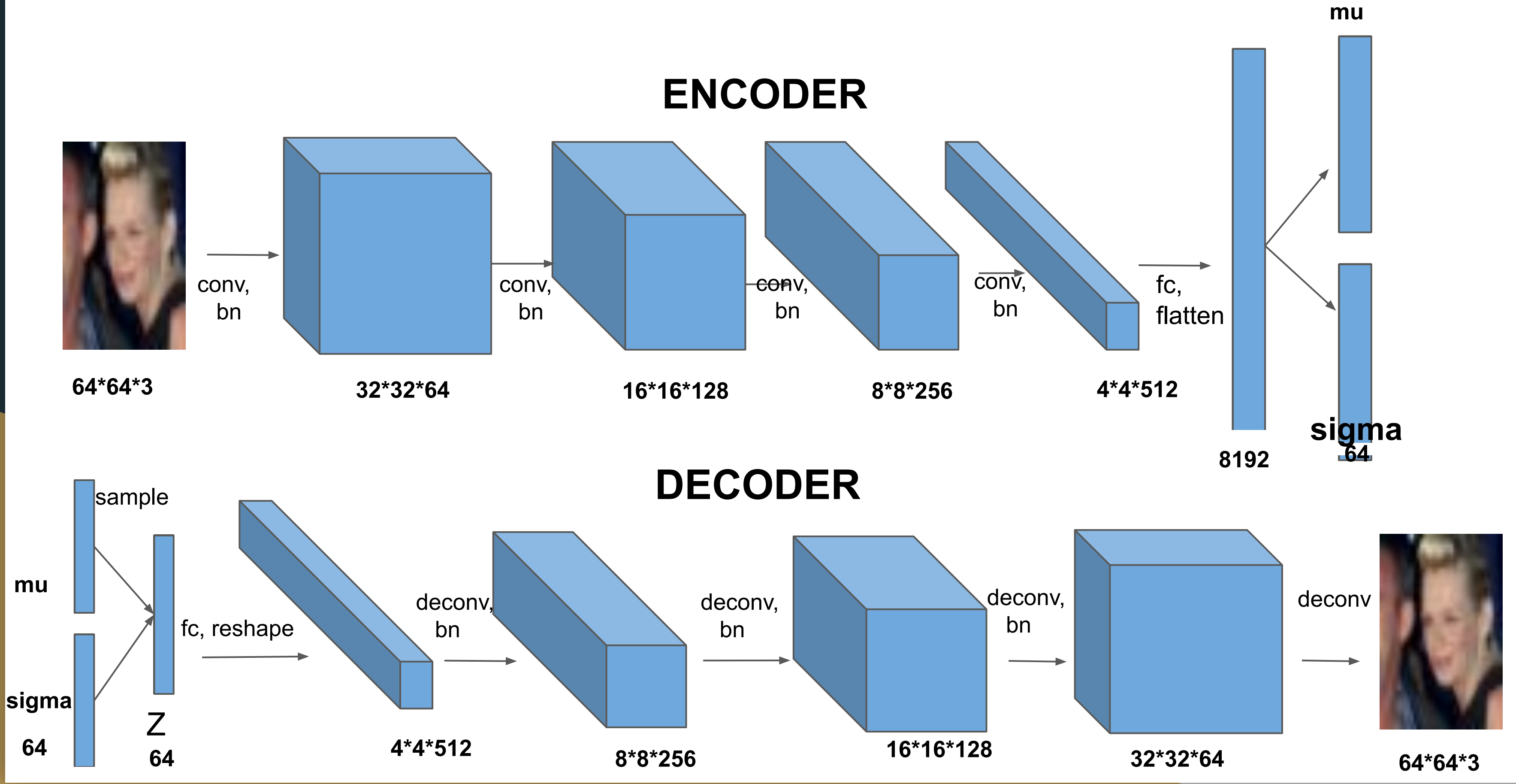


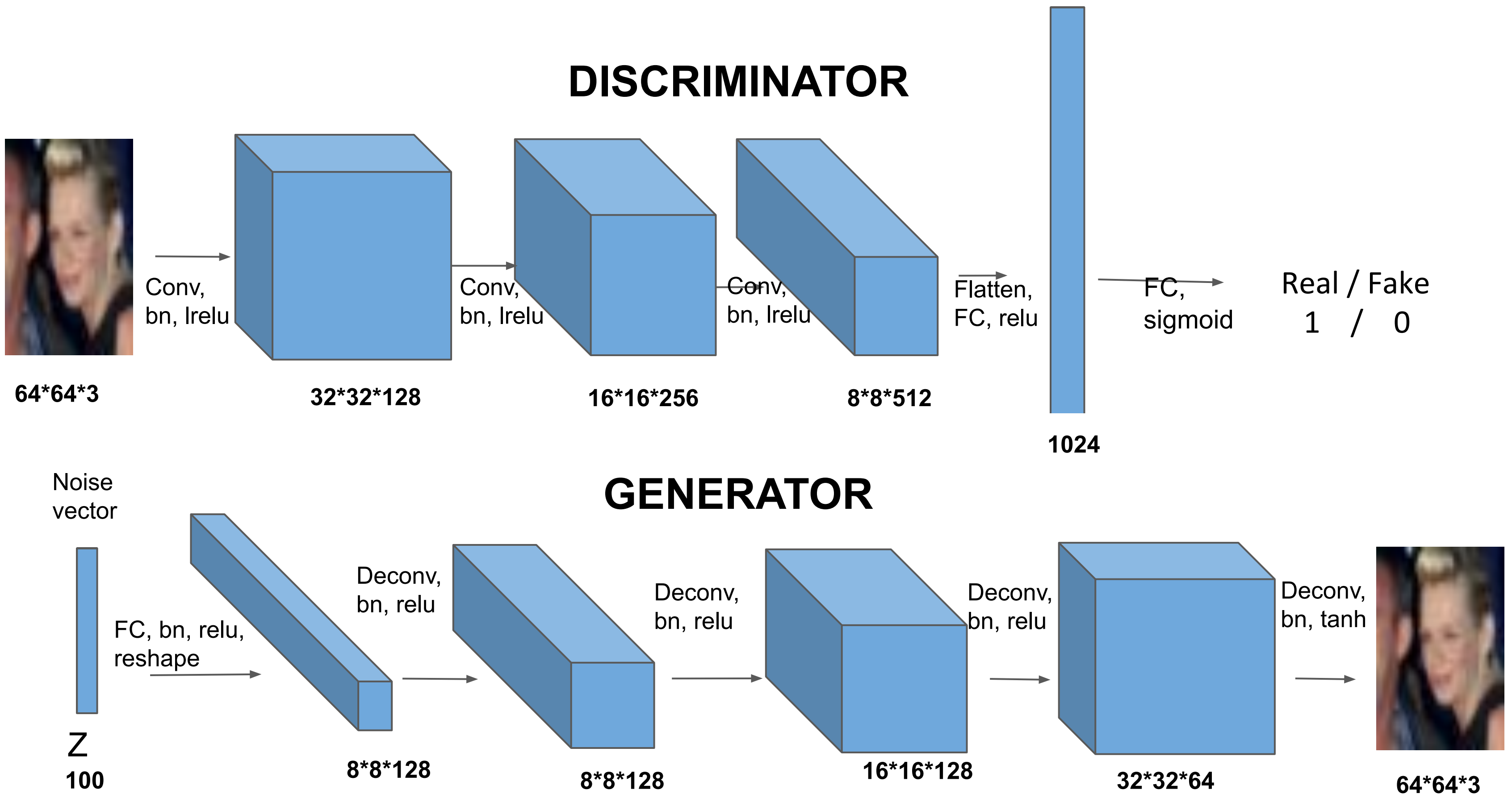
CelebA Dataset[1]

(1) “Deep Learning Face Attributes in the Wild” *Liu Z., Luo P., Wang X., Tang* X., ICCV 2015

Sample from the CelebA dataset

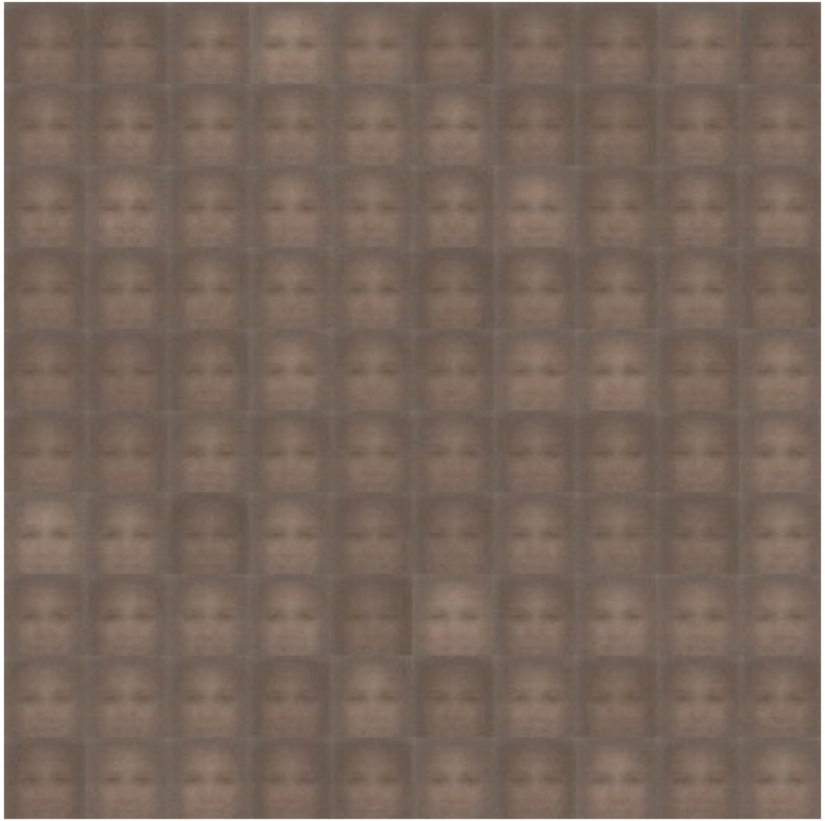
VAE model architecture on CelebA



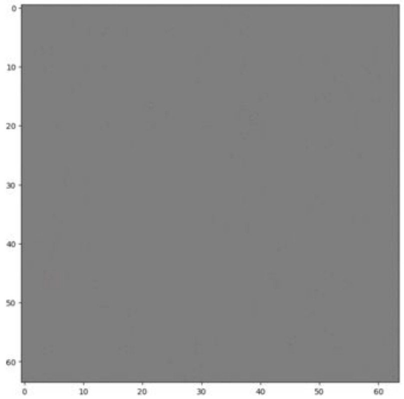

DISCRIMINATOR
64*64*3
Conv, bn, lrelu
32*32*128
Conv, bn, lrelu
16*16*256
Conv, bn, lrelu
8*8*512
Flatten, FC, relu
1024
FC, sigmoid
Real / Fake
1 / 0
GENERATOR
Noise vector
Z
100
FC, bn, relu, reshape
8*8*128
Deconv, bn, relu
8*8*128
Deconv, bn, relu
16*16*128
Deconv, bn, relu
32*32*64
Deconv, bn, tanh
64*64*3

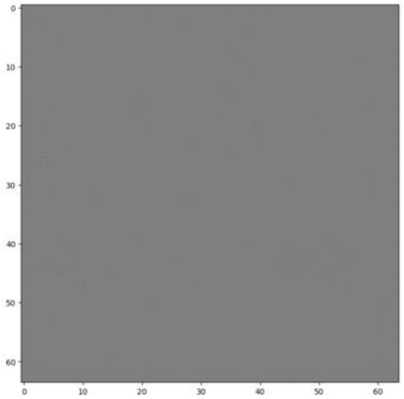
VAE- CelebA Results

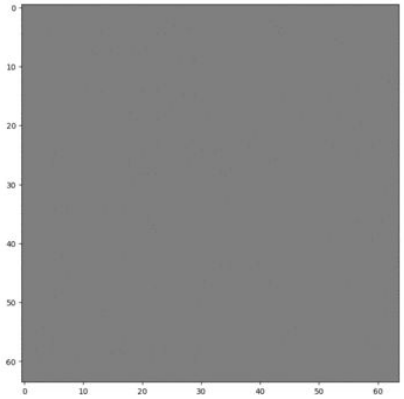
After 20 epochs

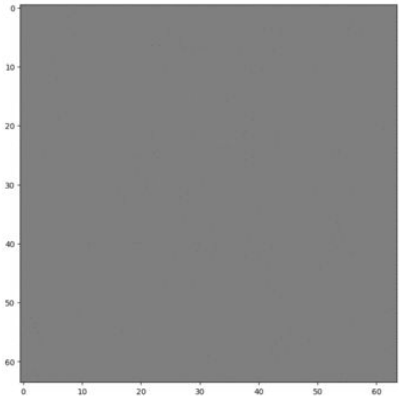


GAN - CelebA Results









Possible Future Work

1. Be more systematic in comparing our models
    - Explore different hyperparameter settings for each model
2. Try more complex architecture for GAN in order to get better output
3. Try higher resolution input images for GAN
4. Try using GANs with latent vectors: http://blog.otoro.net/2016/04/01/generating-large-images-from-latent-vectors/

Thank You!
Questions?