THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

FACE GENERATOR USING DEEP GENERATIVE ADVERSARIAL NETWORKS (DCGAN)

Machine Learning II Academic

- Aira Domingo
- Renzo Castagnino

Outline

1. Introduction
    - Motivation
    - Some Theory
2. Main Question
3. Data Collection & Preprocessing
4. Model
5. Results
6. Conclusion

Motivation

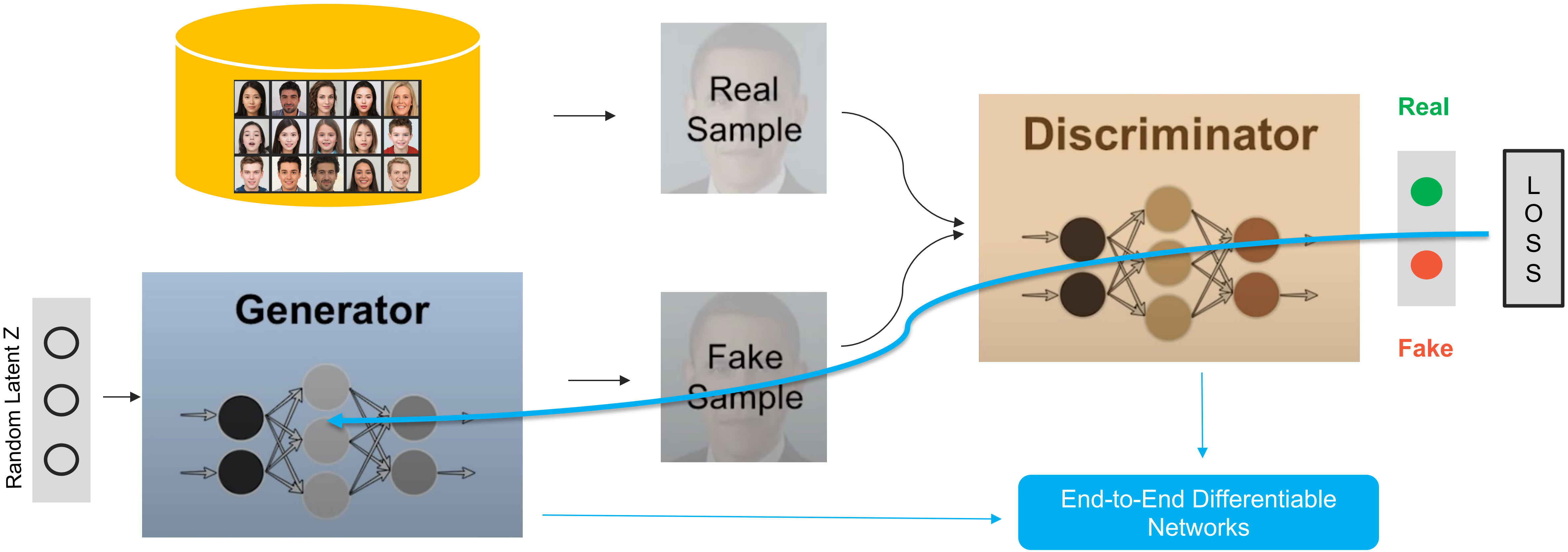
- Advancements in technology can alter our perception of reality.
- Fake images or videos can be generated:
  - Fake news
  - Propaganda
  - Defamation of individuals
- Led for a need to distinguish if an image or video has been manipulated.

Main Question

How can we create faces using machine learning that are unrecognizable as fake to the human eye?

Generative Adversarial Network (GAN)

Some Theory: Generative Adversarial Network (GAN)



Some Theory: GAN Objective Function

$$\min_G \max_D V(D, G) = \underbrace{\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{\text{Training Loss}}$$

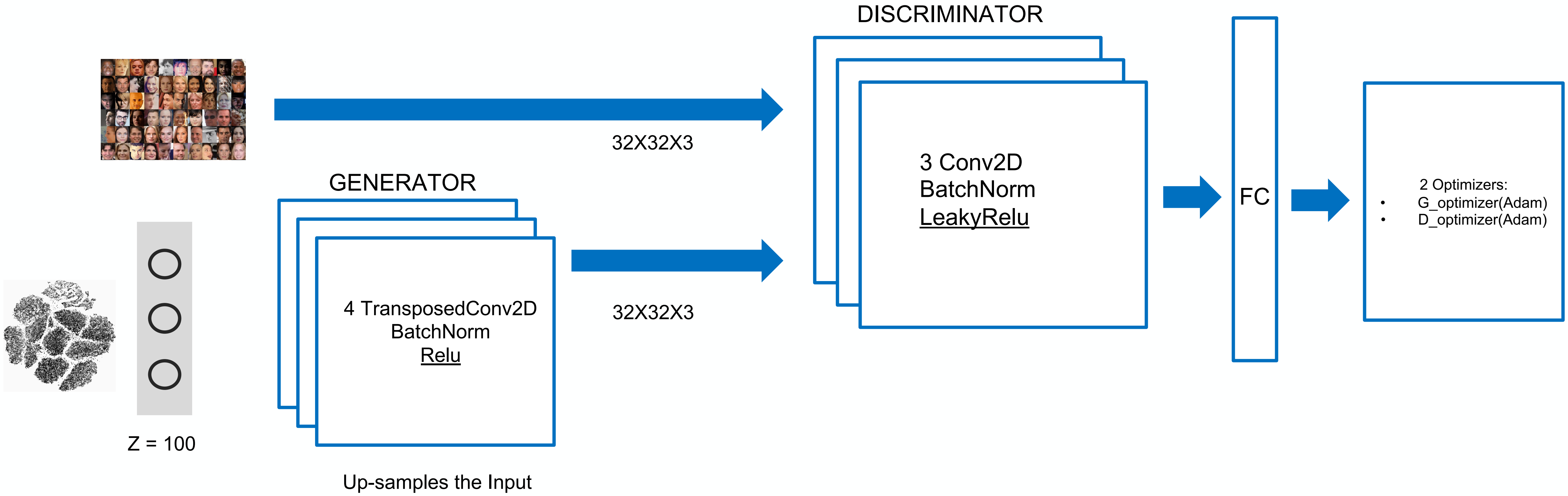
Min-**Max** game:

- **The generator is trying to make good images. Minimize this LOSS**
- **The discriminator is trying to distinguish real images from the generated ones. MAXIMIZE THE LOSS**

Data Collection & Preprocessing

- CelebA dataset
  - Large-scale face attributes dataset.
  - **200K** celebrity images.
  - Images cropped to remove areas that doesn’t include the face
  - Resized to 64x64x3
- Scaled to [-1,1] to match output of tanh activated generator

Model

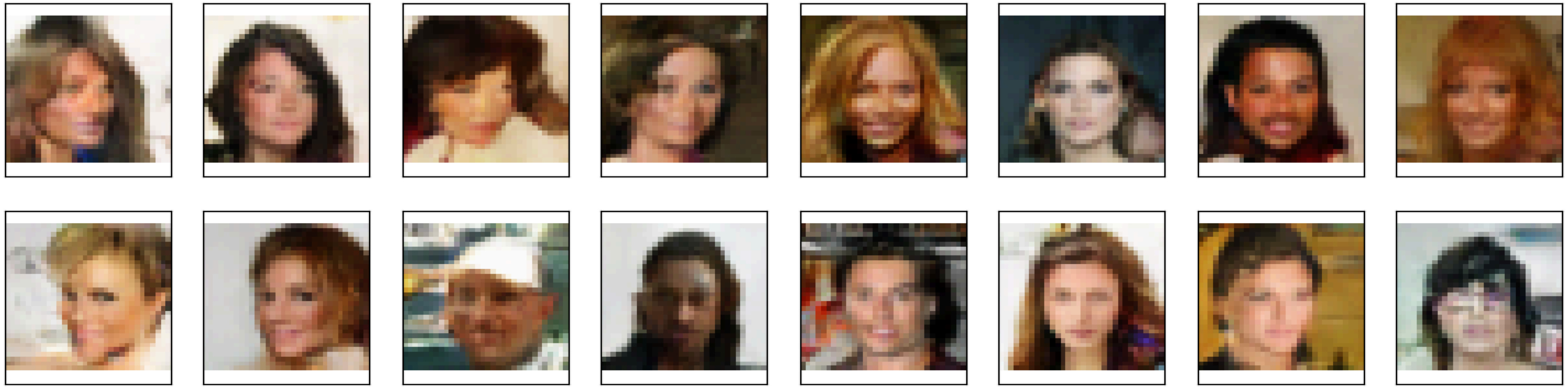


UNSUPERVISED!

Results

- Image Size: 32
- Batch Size: 256
- Epochs: 40
- LR = 0.0001

- Image Size: 32
- Batch Size: 128
- Epochs: 30
- LR = 0.0005

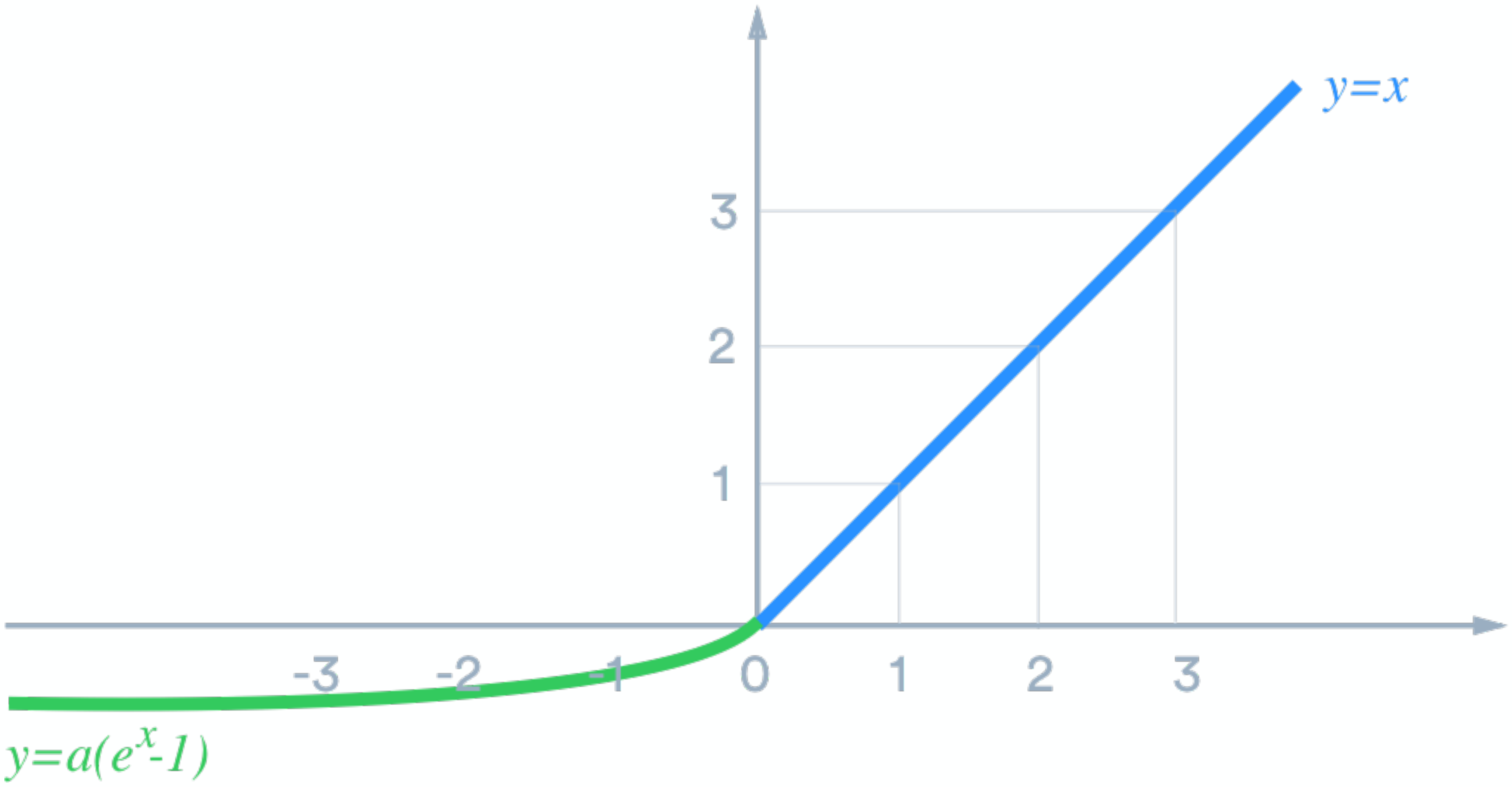


Conclusions

- In this project we have shown that with simple Machine Learning techniques, such as Generative Adversarial Networks, it is possible to create faces.
- To improve the model, further research can be done in state of the art methods such as progressive growing.
- Better quality of image as an input leads to better fake image results.
- Training a GAN can be very time-consuming. (A realistic image of 1024X1024 could take up to 14 days).

QUESTIONS

Leaky ReLU vs ReLU



Go Back

GAN Algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**
  for k steps **do**
- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end for
- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

GAN Type	Key Take-Away
GAN	The original (JSD divergence)
WGAN	EM distance objective
Improved WGAN	No weight clipping on WGAN
LSGAN	L2 loss objective
RWGAN	Relaxed WGAN framework
McGAN	Mean/covariance minimization objective
GMMN	Maximum mean discrepancy objective
MMD GAN	Adversarial kernel to GMMN
Cramer GAN	Cramer distance
Fisher GAN	Chi-square objective
EBGAN	Autoencoder instead of discriminator
BEGAN	WGAN and EBGAN merged objectives
MAGAN	Dynamic margin on hinge loss from EBGAN

Go Back

GAN Algorithm

- Discriminator is too strong, such that the gradient for the generator vanishes and the generator can't keep up
- Can be fixed as follows:

Instead of gradient descent with

$$\nabla_{\mathbf{w}_G} \frac{1}{n} \sum_{i=1}^{n} \log \left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

Do gradient ascent with

$$\nabla_{\mathbf{w}_G} \frac{1}{n} \sum_{i=1}^{n} \log \left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$