ADVANCED IMAGE ANALYSIS

GENERATIVE ADVERSARIAL NETWORK

UNIVERSITÉ DE BOURGOGNE

Professor: Abdul Qayyum

Presented by: Muhamad Izzul Azri
Martin Emile
Pranavan Ramakrishnan

Date: 16 December 2020

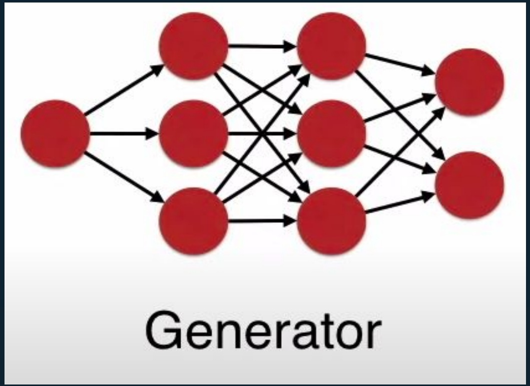
TABLE OF CONTENTS

INTRODUCTION

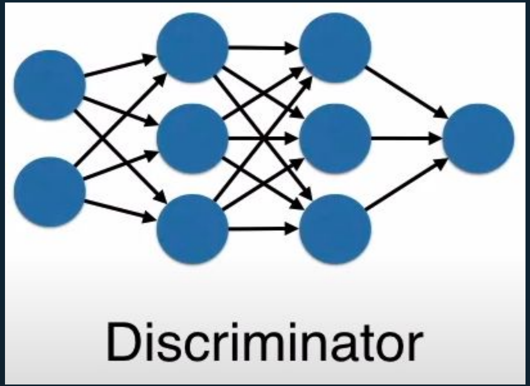
- Motivation for generative models
- GANs Architecture
- GANs Objective
- DCGANs

What is GAN?

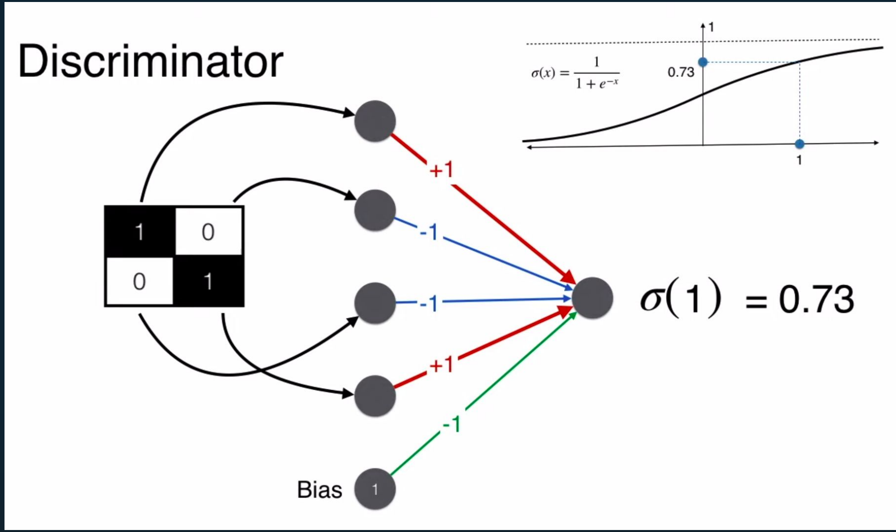
“Training a network to correctly classify adversarial examples(Discriminator) by training the network on adversarial examples(Generator).” - Ian Goodfellow

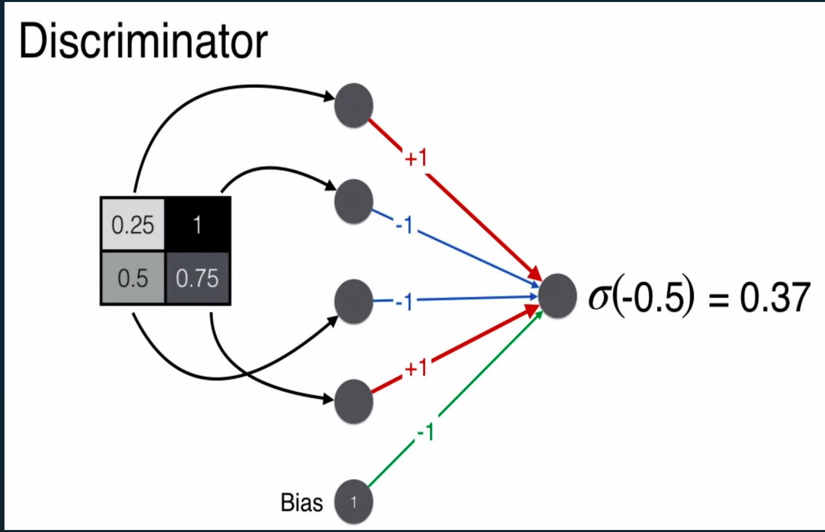


vs

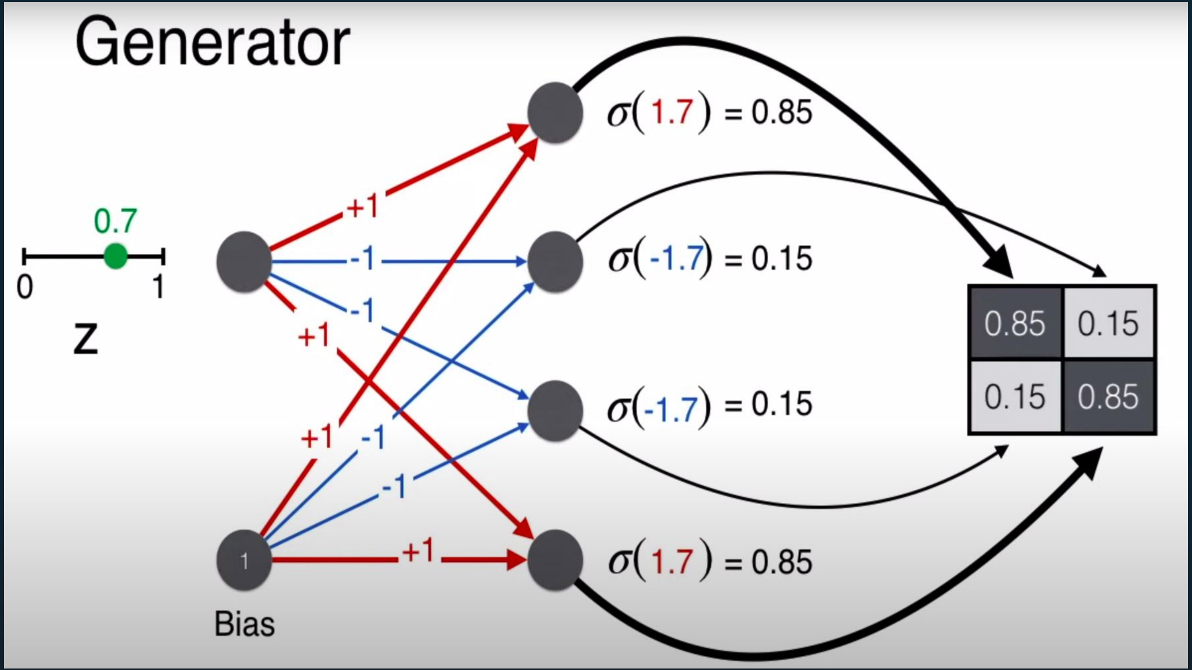


Discriminator:



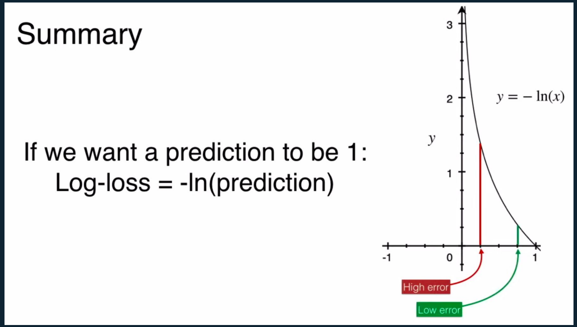


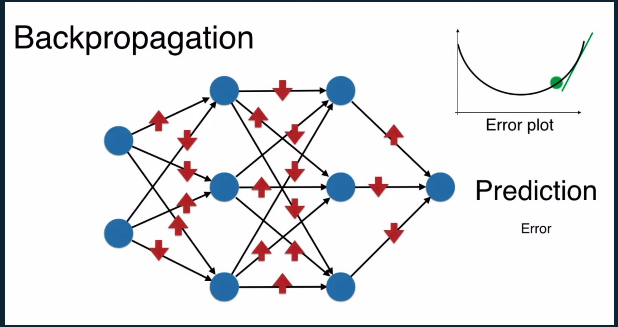
Generator:

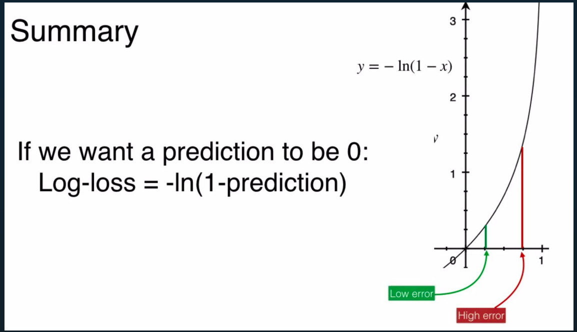


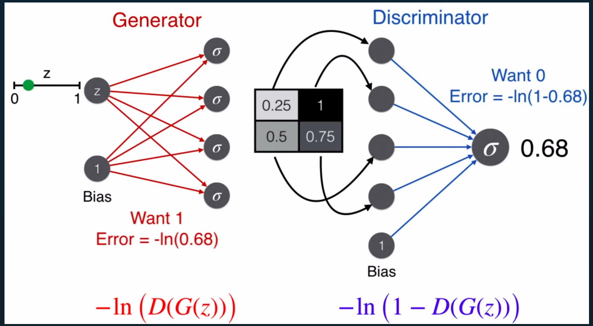
Generator
0.7
0
1
Z
Bias
1
+1
-1
-1
+1
+1
-1
-1
+1
$\sigma(1.7) = 0.85$
$\sigma(-1.7) = 0.15$
$\sigma(-1.7) = 0.15$
$\sigma(1.7) = 0.85$
0.85
0.15
0.15
0.85

The training process:



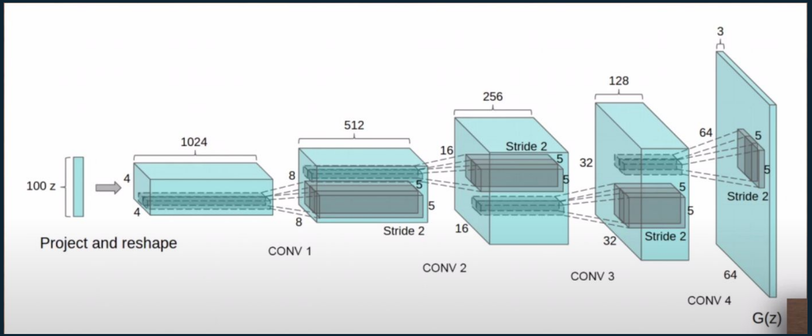


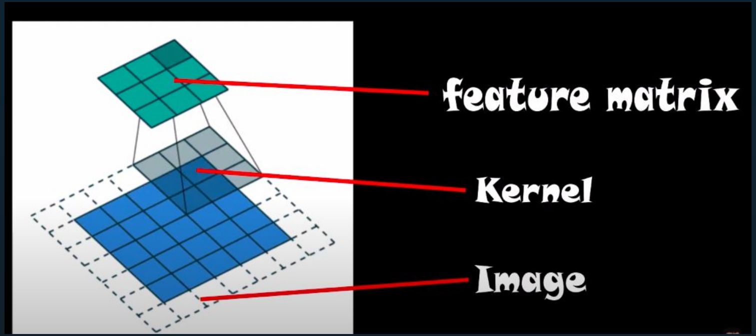




Deep Convolutional GAN:

- Replace any pooling layers with strided convolutions(discriminator) and fractional-strided convolutions(generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.





GANs Methodology

- Advantages of GANs
- Autoencoders
- Context Encoders for Image Generation
  - Encoder
  - Channel-wise Fully Connected Layer
  - Decoder
  - Loss Function

Advantages of GANs

- Plenty of existing work on Deep Generative Models
  - Boltzmann Machine
  - Deep BeliefNets
  - Variational AutoEncoders (VAE)
- Why GANs?
  - Sampling (or generation) is straightforward.
  - Training doesn't involve Maximum Likelihood estimation
  - Robust to Overfitting since Generator never sees the training data
  - Empirically, GANs are good at capturing the modes of the distribution

Autoencoders

- CNN structure that is used for reconstruction tasks
- The structure of the model is:
  - Output size is same as input size
  - Have two parts:
    - Encoder: for feature encoding, aiming for a compact latent feature representation of input
    - Decoder: For decoding the latent feature representation
  - The middle layer is usually call low-dimensional “bottleneck” layer.

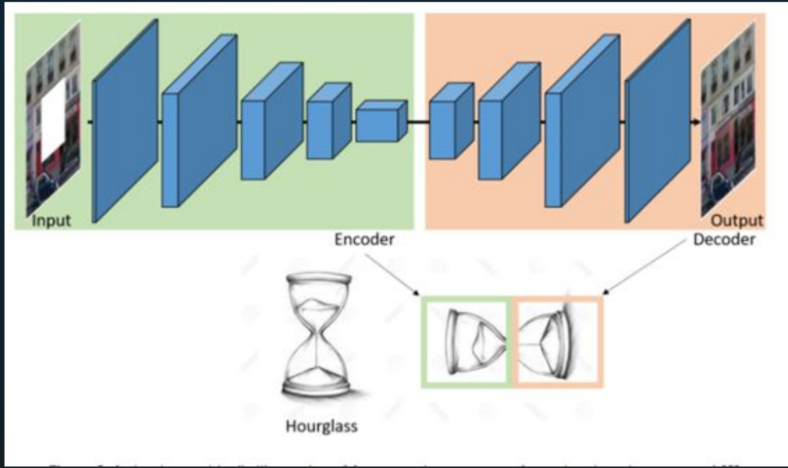


Figure 1. Autoencoders structure (encoder-decoder structure)

Context Encoders for Image Generation

- The figure below is the proposed context encoder
  - First, The input is masked image (The missing in image)
    - The input is fed into encoder for obtaining encoded features
  - Then, The Channel-wise Fully Connected Layer is placed between encoded features and decoded features for getting better semantic features (Bottleneck)
  - Finally, A decoder reconstruct the missing parts using bottleneck features

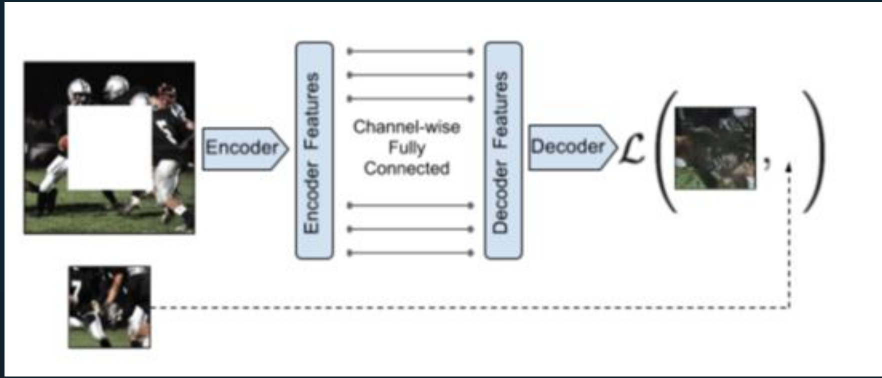


Figure 2. Overview of the proposed Context Encoder

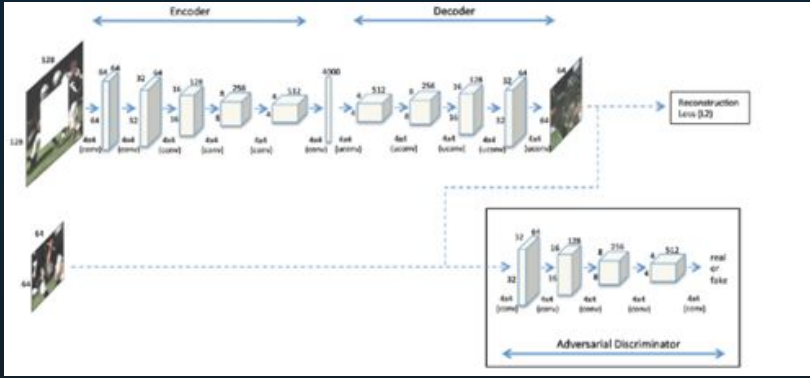


Figure 3. Detailed architecture of the proposed network

Encoder

- Follows the AlexNet architecture. Trained their network from scratch with randomly initialized weights
- Compared to original AlexNet architecture and Autoencoders as shown in Figure 1 ,
  - The main difference is the middle **Channel-wise Fully Connected Layer**
  - If only convolutional layers in the network, it's no way to make use of the features at distant spatial locations in feature maps. To solve this,
    - Use fully-connected layers such the value of each neuron at current layer is depended on all the values of the neurons at previous layer.
    - However, fully-connected layer induces many parameters.
    - As example, 4x4x512= 8192 results in 8192x8192 = 67.1M parameters.
    - The proposed method was channel-wise fully connected layer

Channel-wise Fully Connected Layer

- Fully connect each channel independently instead of all the channels
- As example,
  - We have m feature maps with size of n x n. If standard fully-connected layer is used, we will have $m^2 n^4$ parameters excluding bias term
  - For channel-wise fully-connected layer, we have mn^4 parameters.
  - Therefore, we can capture the features from distant spatial locations without adding so many extra parameters

Decoder

- It is simply reverse of the encoding process
- Use a series of transposed convolutions to obtain reconstructed image with desired size

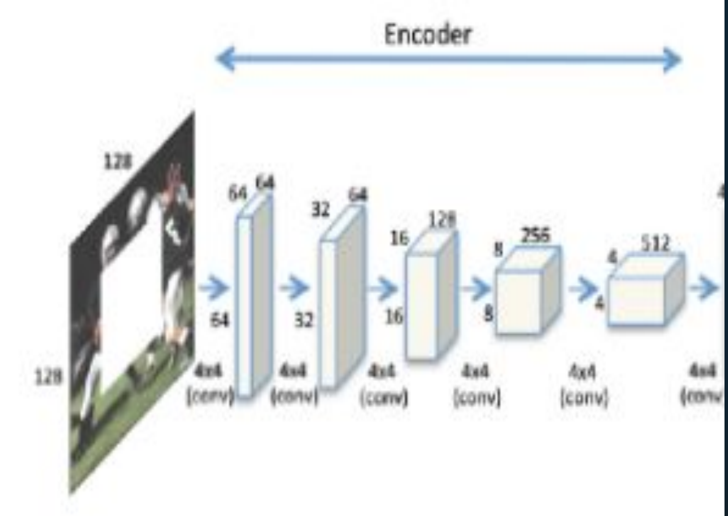


Figure 4. Encoder

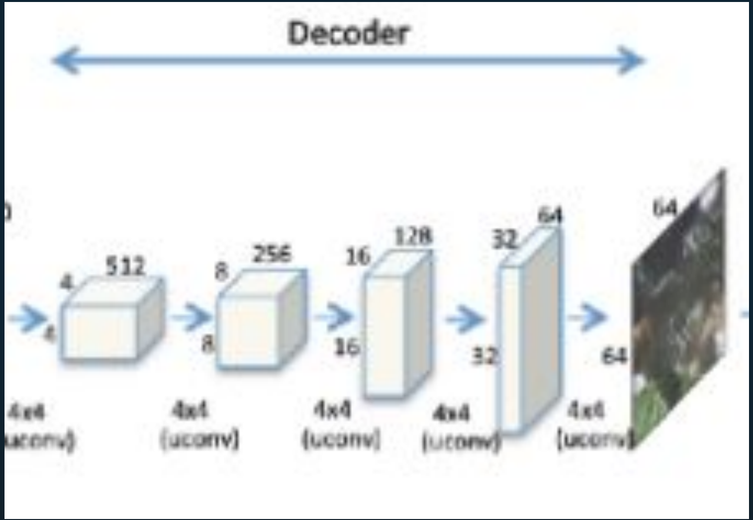


Figure 5. Decoder

Loss Function

- It is consist of two terms
  - First term, reconstruction loss (L2 loss) which focuses on pixel-wise reconstruction accuracy (PSNER-Oriented loss) and always results in blurry images
  - Second term, an adversarial loss which is commonly used in GANs. It encourages closer data distributions between real images and filled images.

$$\mathcal{L}_{rec}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2, \qquad (1)$$

Figure 6. Reconstruction Loss (L2 Loss), M(hat) indicates the missing regions (1 for missing parts, 0 for valid pixels), F is the generator

- L2 Loss: Compute the L2 distance (Euclidean distance) between generated pixels and ground truth pixels from corresponding real image

Loss Function

$$\mathcal{L}_{adv} = \max_{D} \quad \mathbb{E}_{x \in \mathcal{X}}[\log(D(x)) + \log(1 - D(F((1 - \hat{M}) \odot x)))], \qquad (2)$$

Figure 7. Adversarial Loss, D is the discriminator. We want to train a discriminator that can distinguish filled images from real images

- Adversarial Loss: The structure of adversarial discriminator is shown in previous figure. The output of the discriminator is single binary value either 0 or 1. If the input is real image while 0 if the input is a filled image

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}.$$

Figure 8. Joint Loss. Lambda_rec is set to 0.999 while Lambda_adv is set to 0.001

- Both generator and discriminator are trained alternately using Stochastic Gradient Descent (SGD), Adam optimizer

Our Chosen Application of GAN

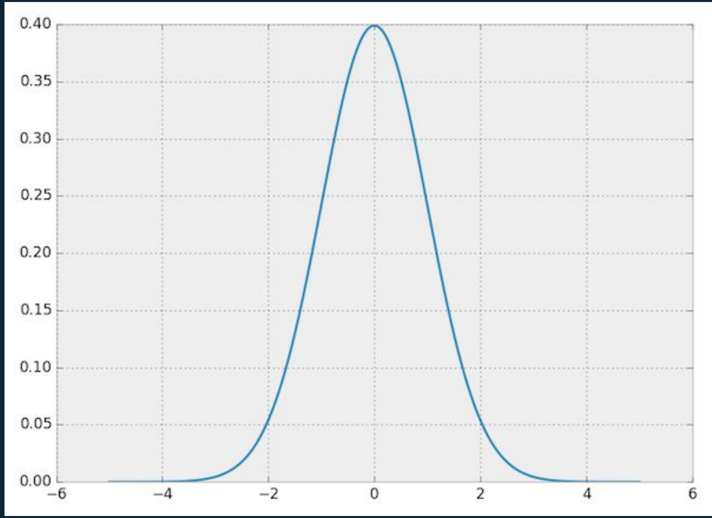
- Image Inpainting

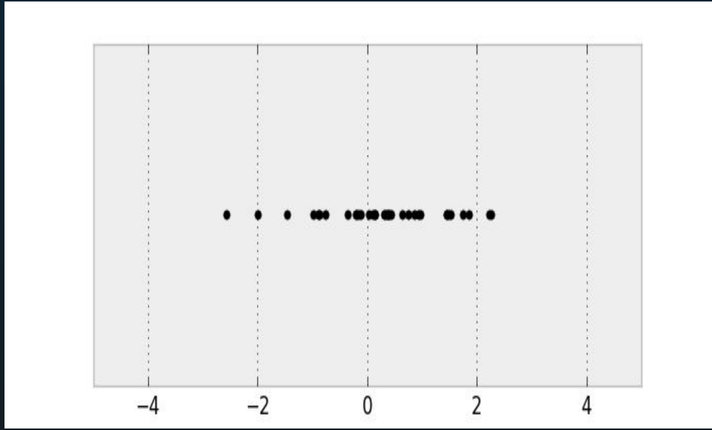
Applications on GAN

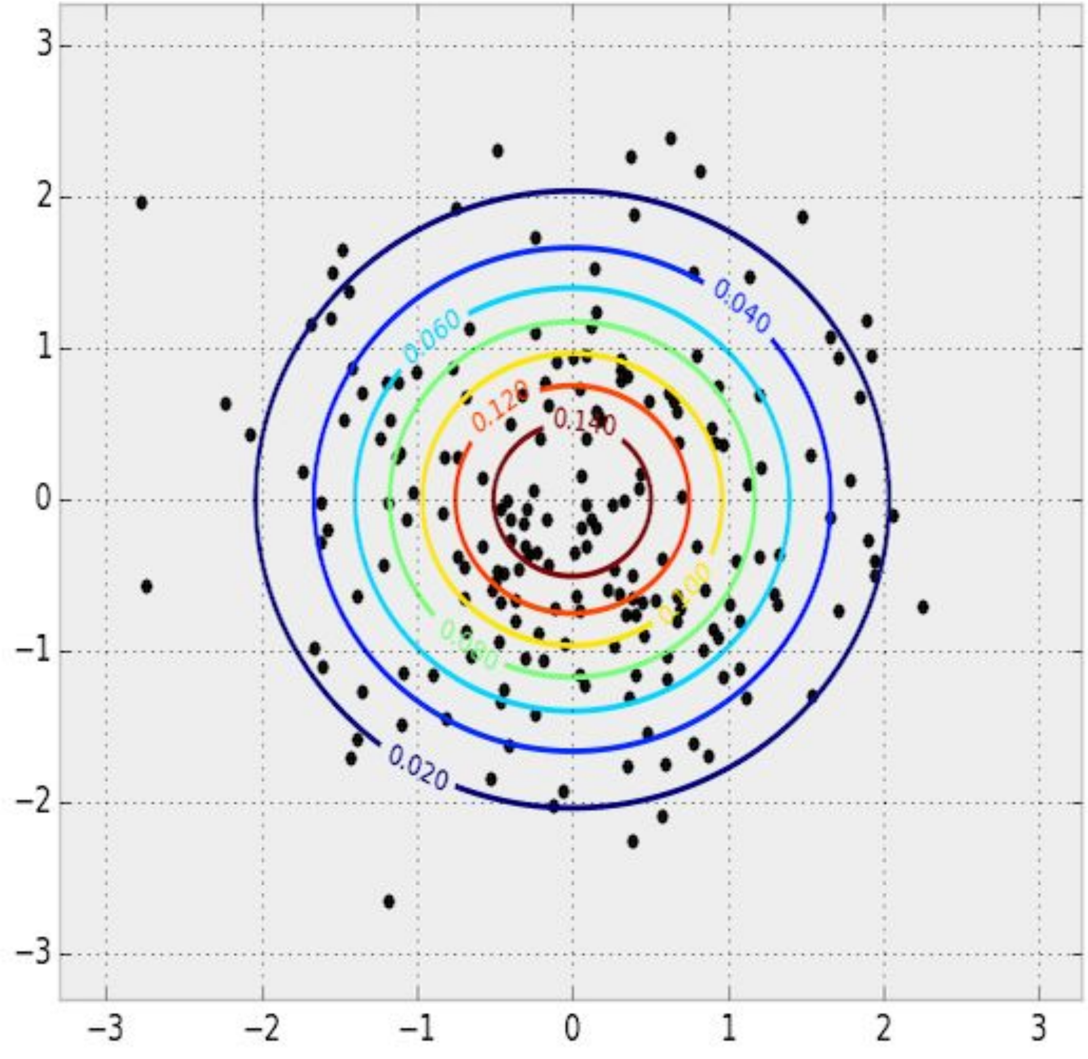
Type of losses

- Contextual loss “What to fill !!”
- Perceptual Loss “ Is it real !!”

Statistical image

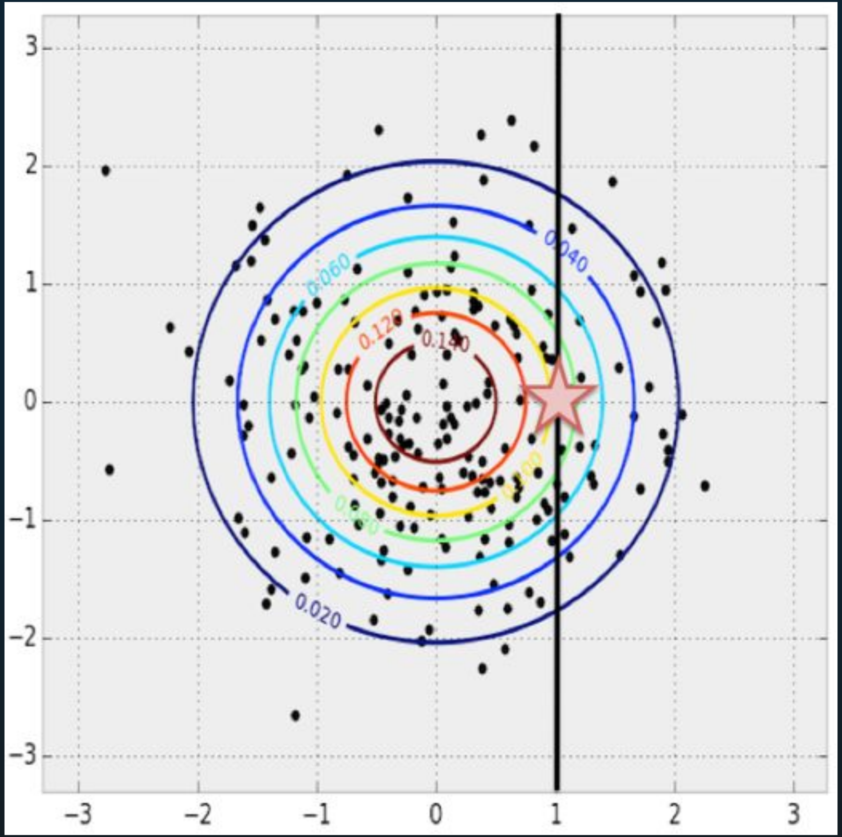




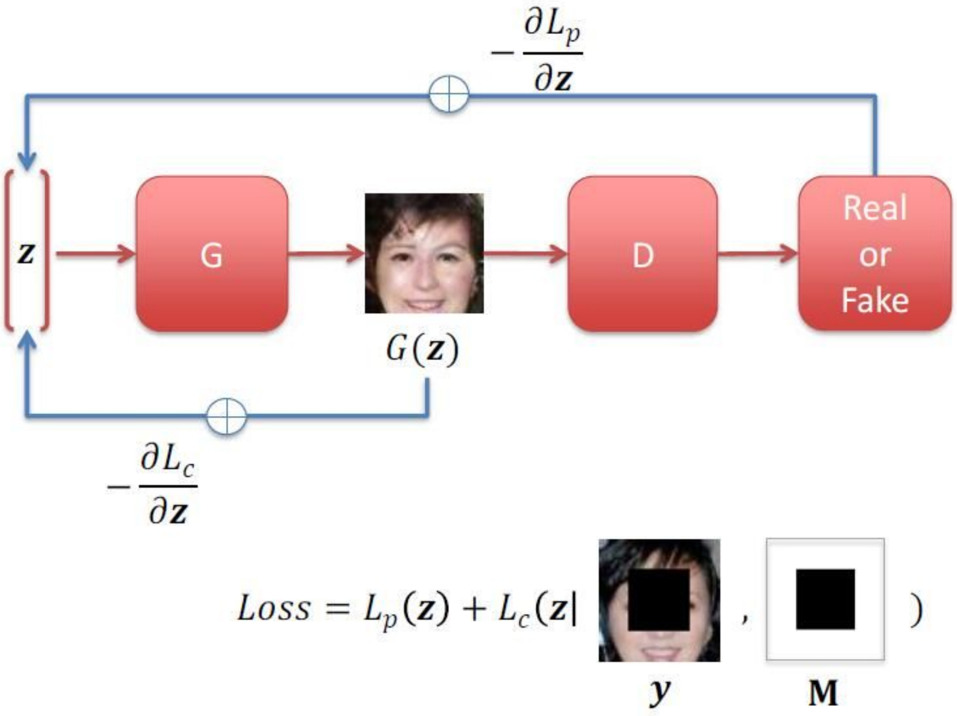


2D Image

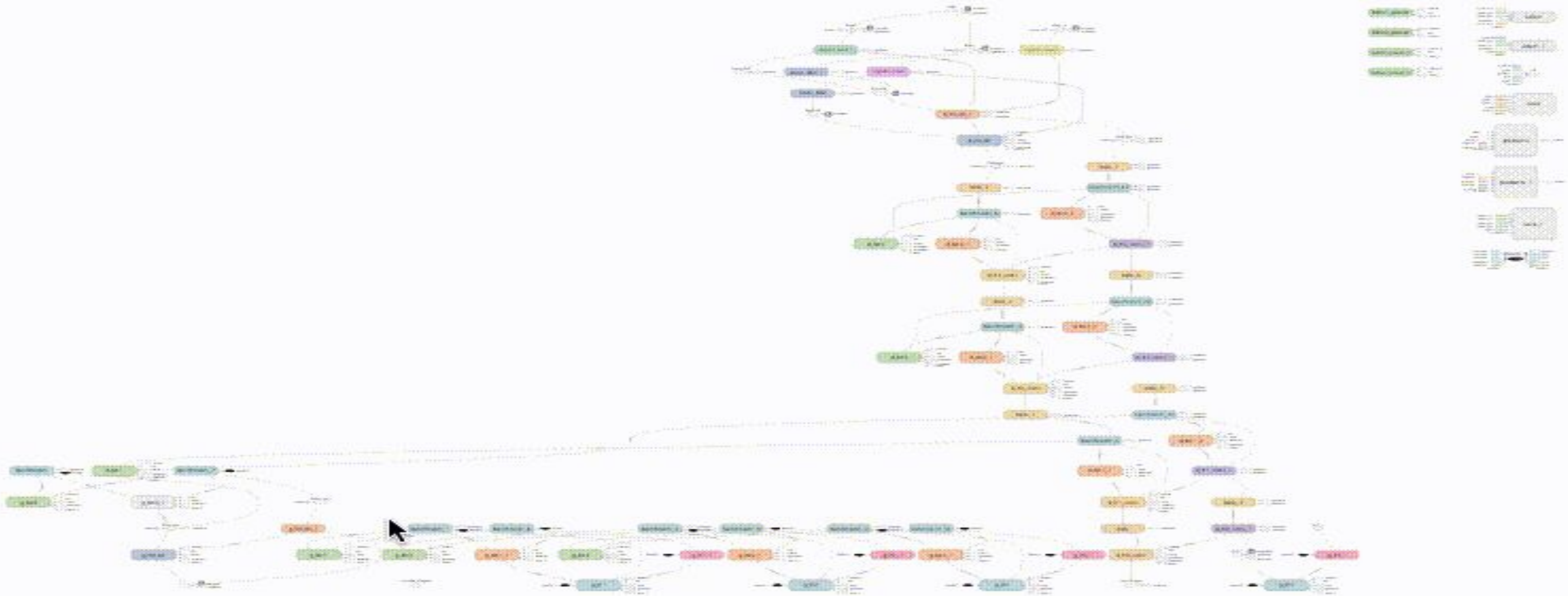
Most probable Image



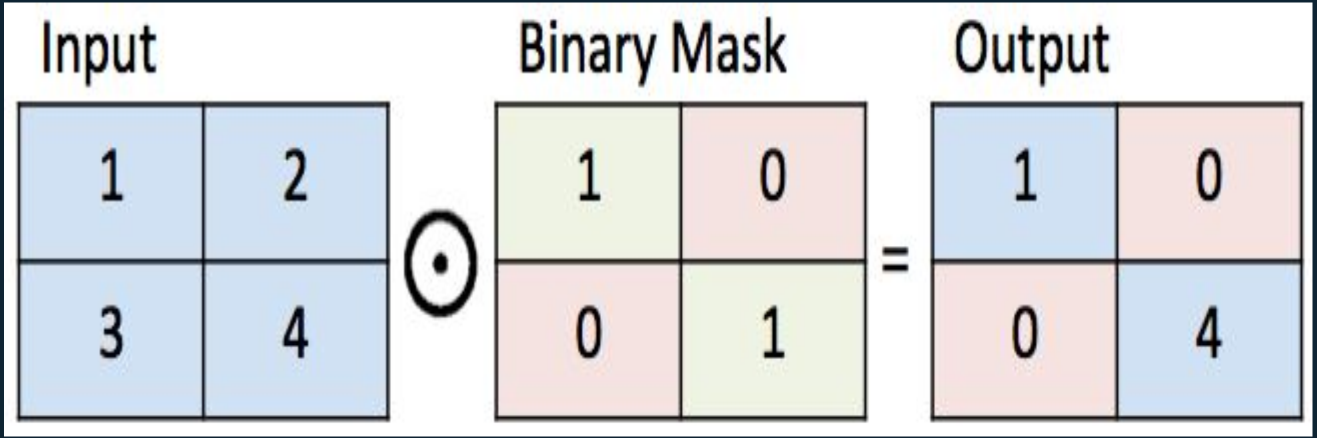
Generating fake Images



(a)

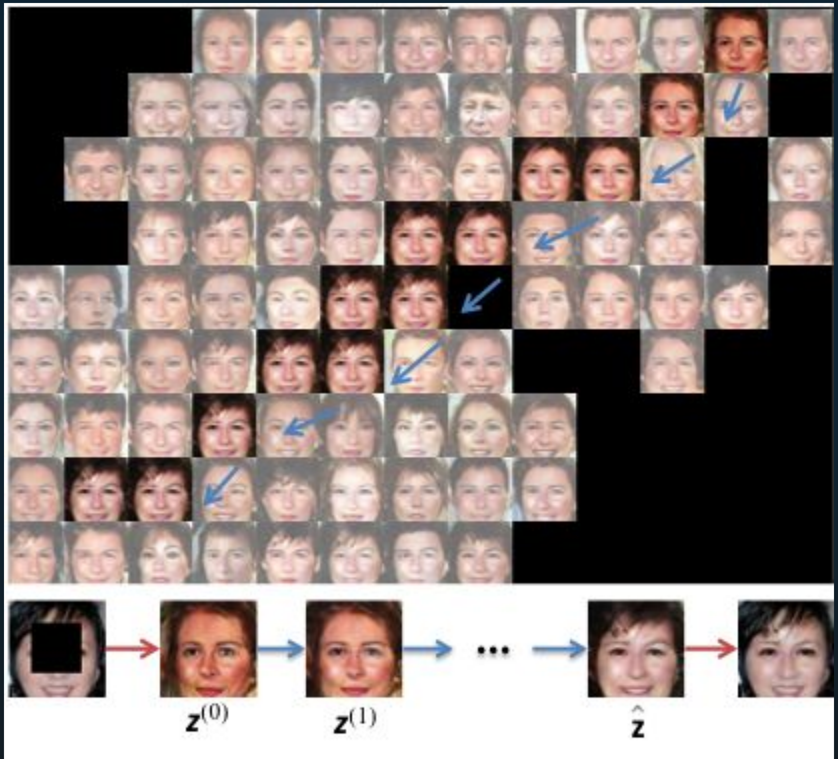


Construction



$x = M \odot y + (1 - M) \odot G(\hat{z})$
L perceptual $(z) = \log(1 - D(G(z)))$

Generating fake Images



Visual Result

Arrange in coulombs: 1- Real Image 2- Input Image 3- Context encoder 4- The paper result

Thank You For your Attention