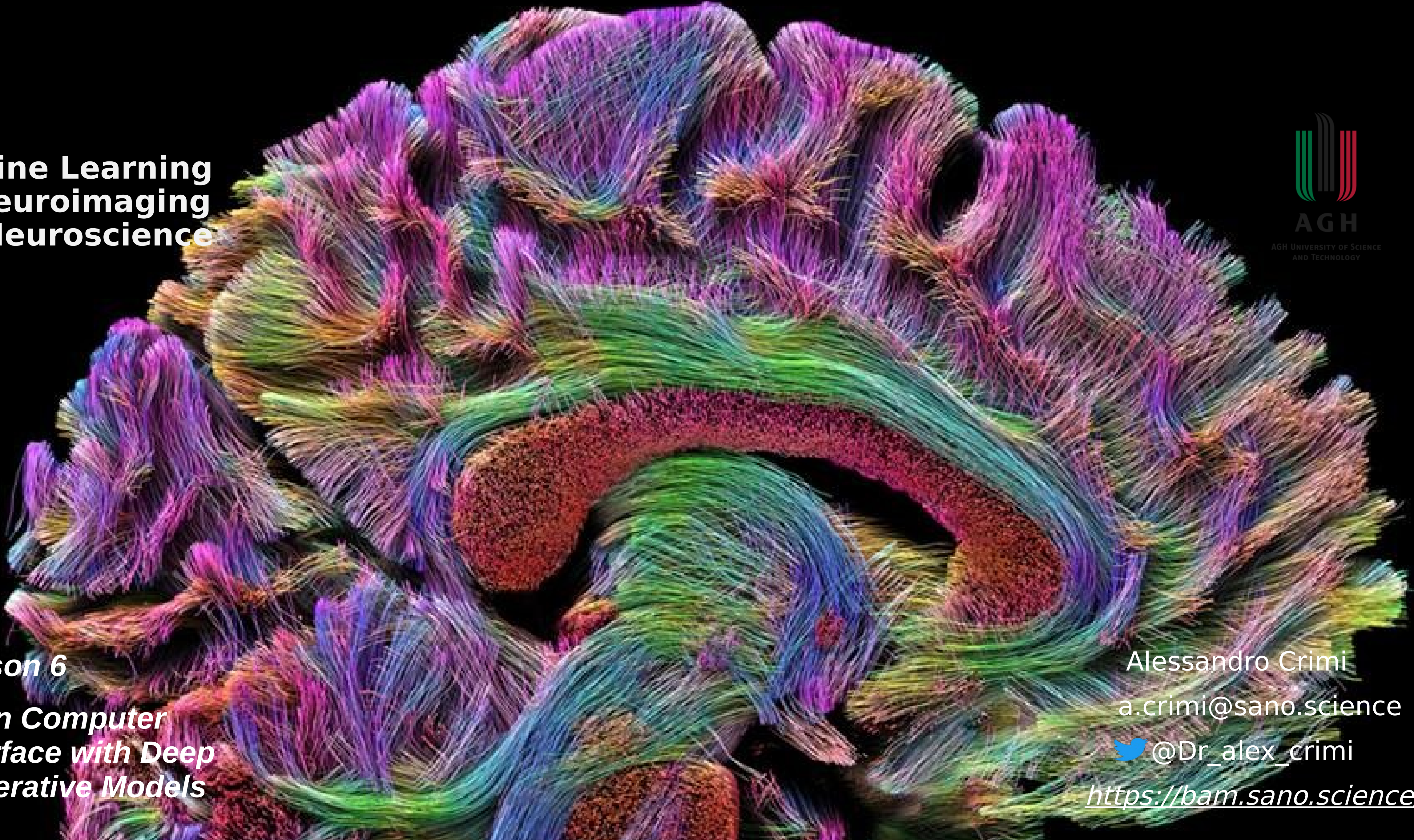


Machine Learning for Neuroimaging and Neuroscience

AGH
AGH University of Science and Technology

Lesson 6

Brain Computer Interface with Deep Generative Models

Alessandro Crimi
a.crimi@sano.science
@Dr_alex_crimi
https://bam.sano.science/

Outline

Deep Convolutional Generative Adversarial

Network (DCGAN) BigBiGAN

Tricks for more realistic image construction

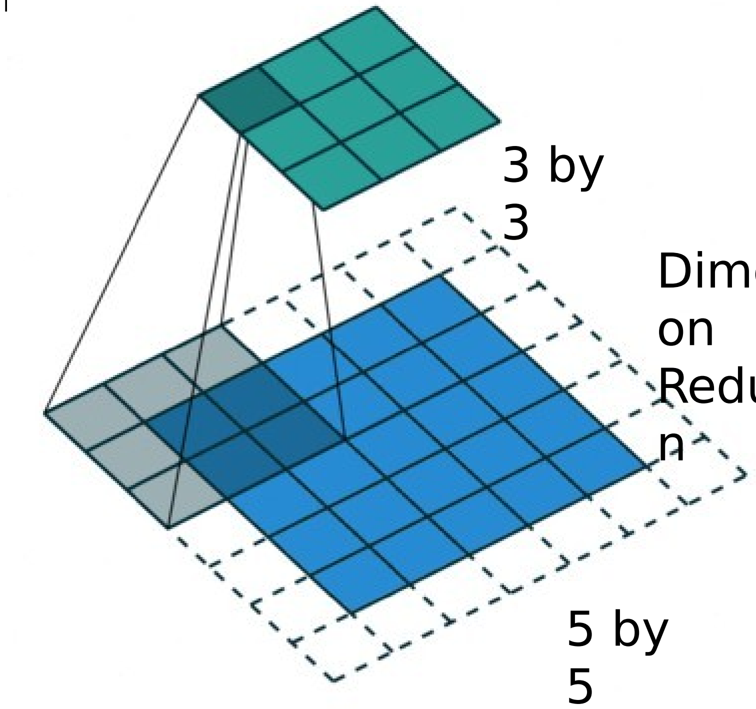
using GANs Image reconstruction from

brain signals

Evaluation of brain readers

Towards Brain Computer Interface (BCI)

Deconvolution



3 by 3

5 by 5

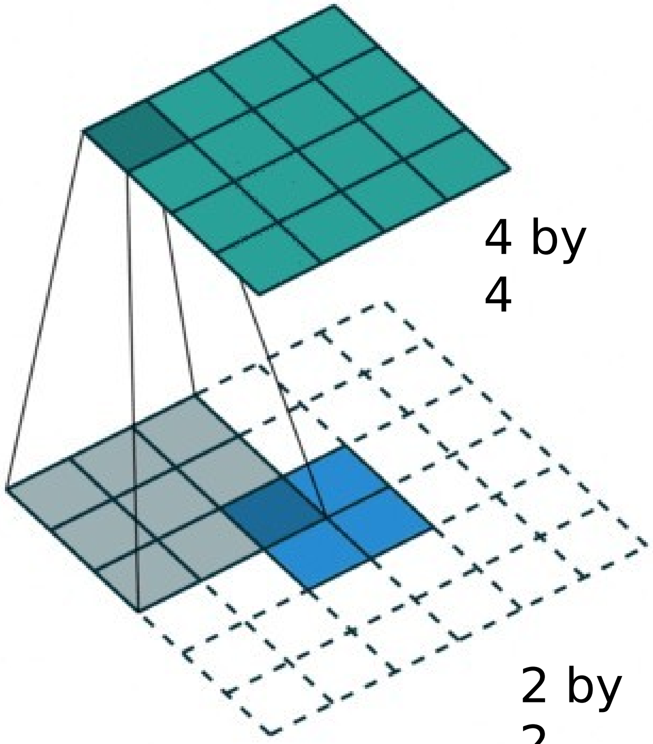
2D Convolution

Output Feature Map

Dimension Reduction

Input Feature Map

Dimension Expansion



4 by 4

2 by 2

2D De-Convolution

Pytorch:

torch.nn.ConvTranspose2d(*in_channels, out_channels, kernel_size, stride, padding)*

torch.nn.ConvTranspose2d(in_channels=*1, out_channels=1, kenel_size=3, stride=0, padding=2)*

Image credits: https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

Deep Convolutional GAN (DCGAN)

Input Image
64x64x3

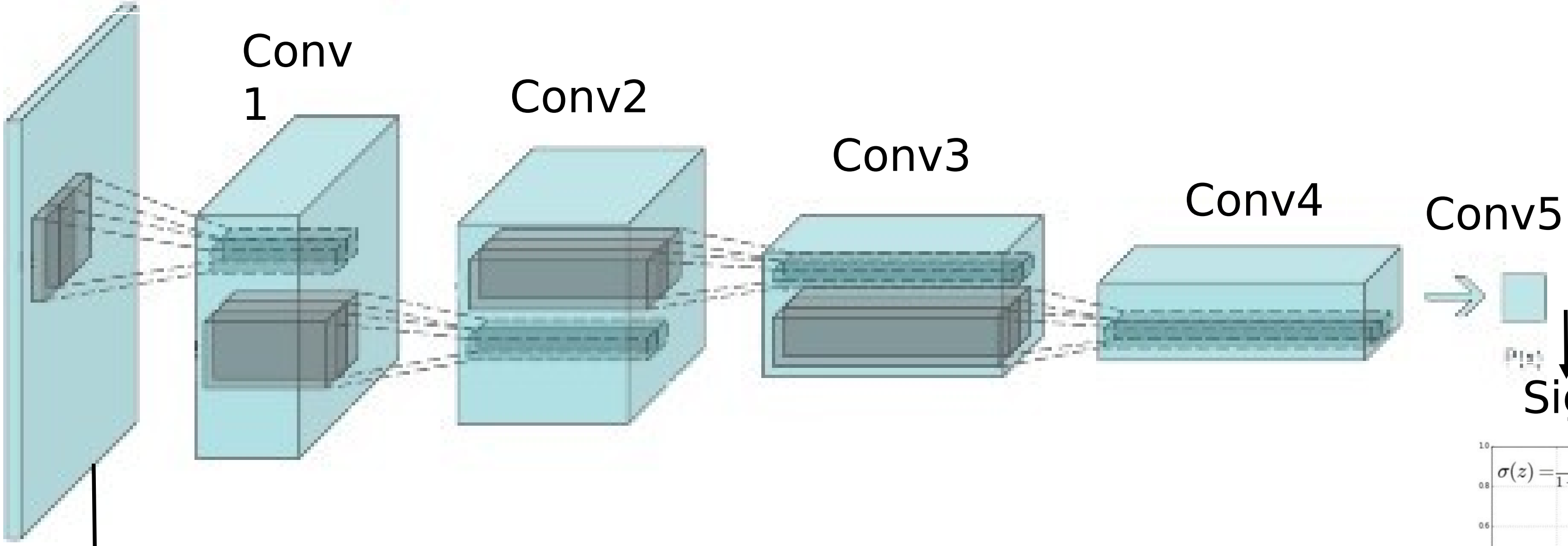
Conv
1

Conv2

Conv3

Conv4

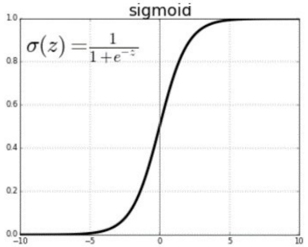
Conv5



0.2 Dog

0.8 non-dog

Sigmoid



Normalize to [0,1]

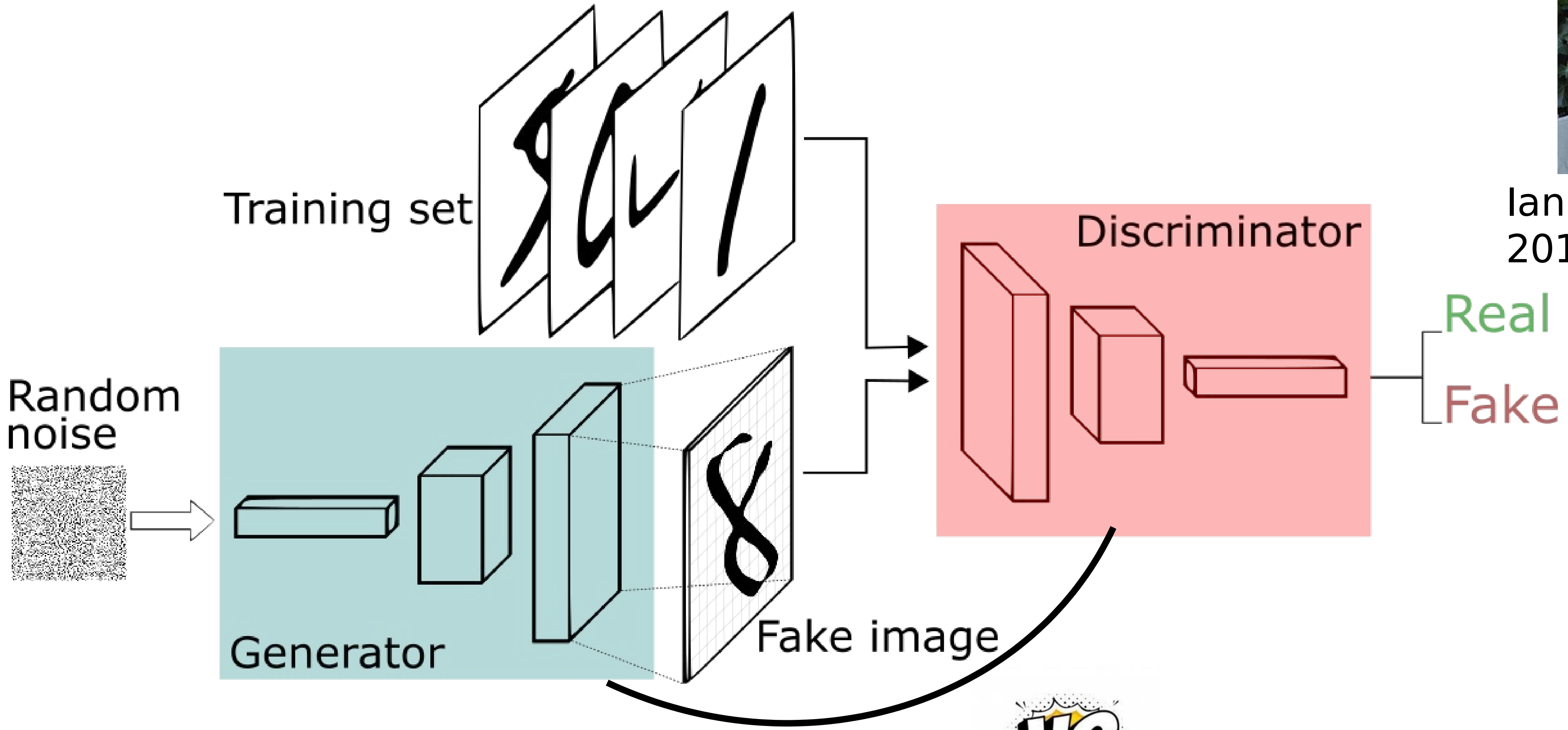
```
# Create the dataset
dataset = dset.ImageFolder(root=dataroot,
                           transform=transforms.Compose([
                               transforms.Resize(image_size),
                               transforms.CenterCrop(image_size),
                               transforms.ToTensor(),
                               transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
                           ]))
```

Normalize image pixel value from [0,1] to [-1,1]

Image credit: https://gluon.mxnet.io/chapter14_generative-adversarial-networks/dcgan.html

Generative Adversarial Networks

Ian Goodfellow, 2014

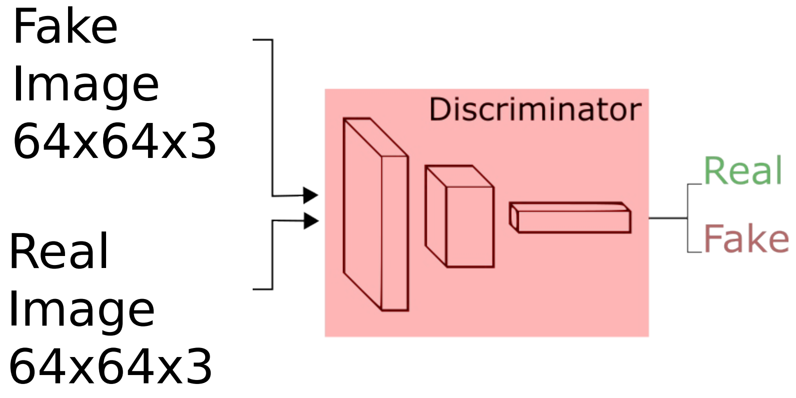


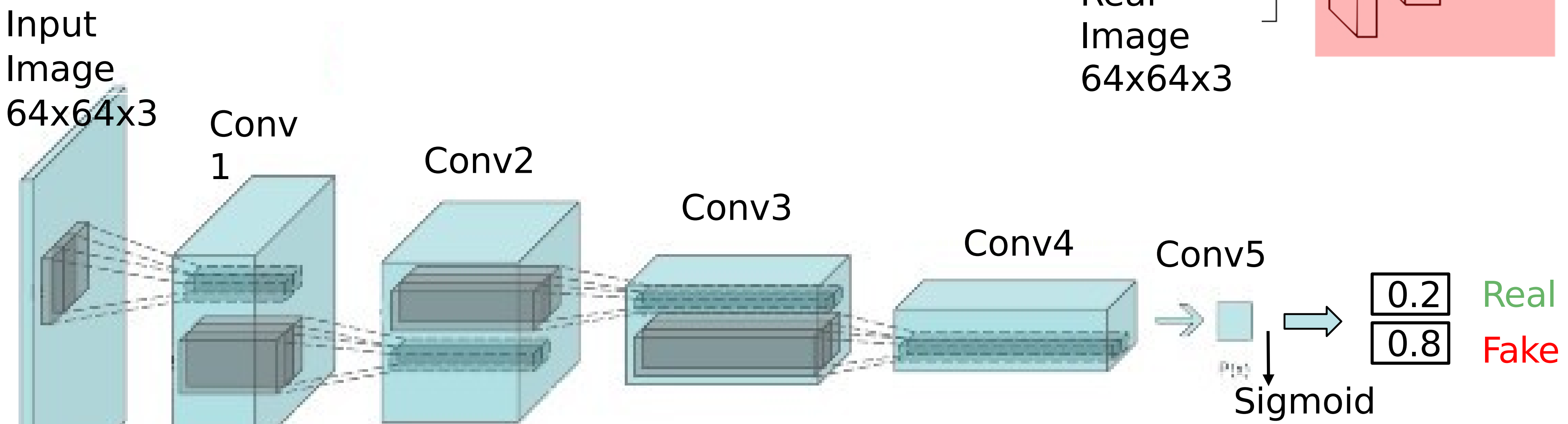
Generator  Discriminator

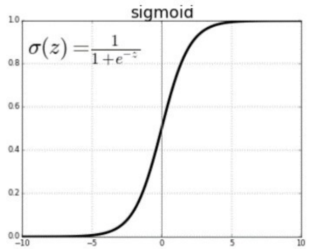
Min-max Game: minimize the possible loss for a worst scenario

Image credit: https://sthalles.github.io/intro-to-gans/

Discriminator







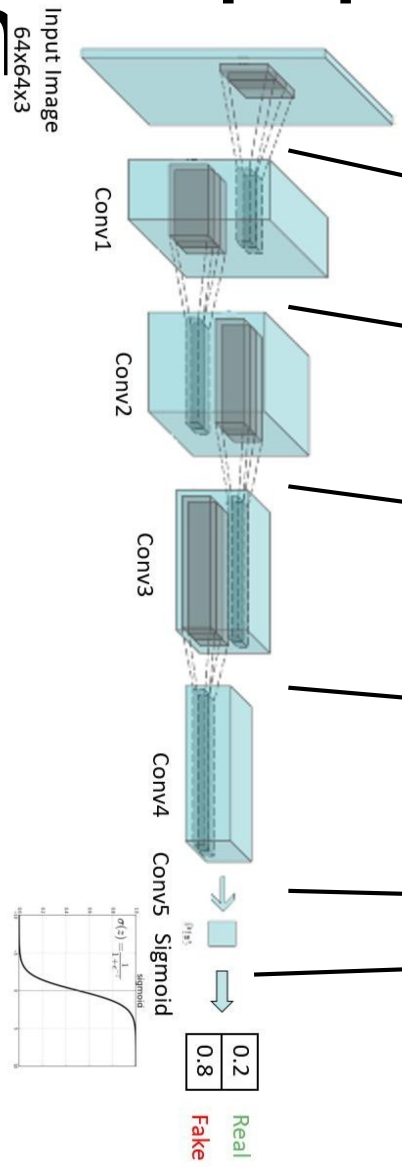
Normalize to [0,1]

```
# Create the dataset
dataset = dset.ImageFolder(root=dataroot,
                           transform=transforms.Compose([
                               transforms.Resize(image_size),
                               transforms.CenterCrop(image_size),
                               transforms.ToTensor(),
                               transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
                           ]))
```

Normalize image pixel value from [0,1] to [-1,1]

Image credit: https://gluon.mxnet.io/chapter14_generative-adversarial-networks/dcgan.html

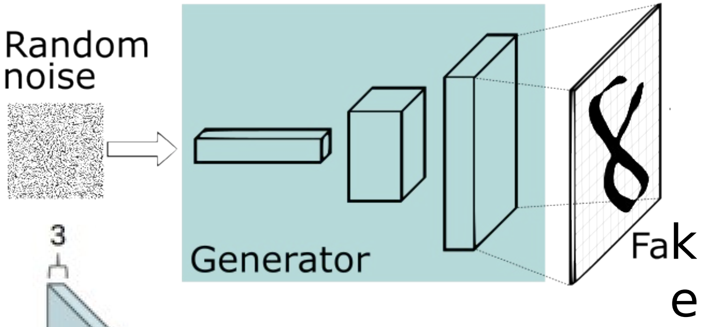
Discriminator – pytorch implementation

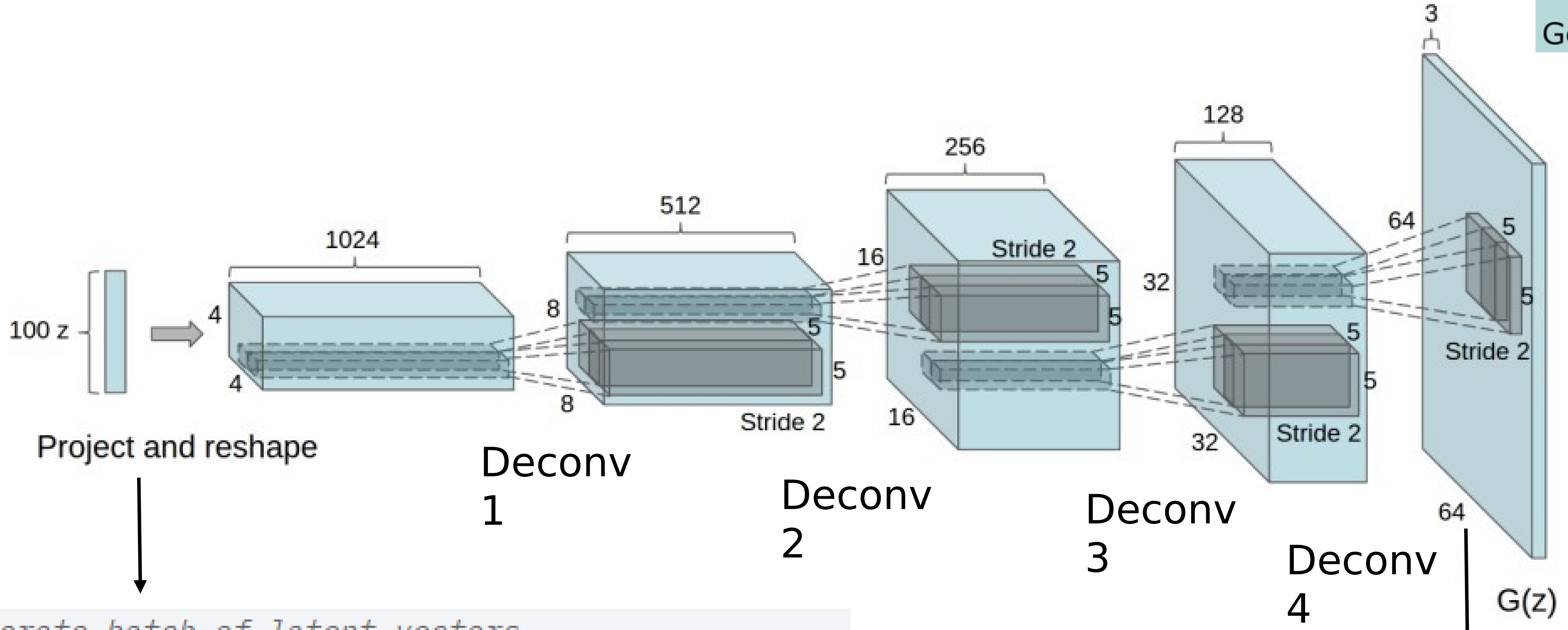



```python
class Discriminator(nn.Module):
    def __init__(self, ngpu):
        super(Discriminator, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            # input is (nc) x 64 x 64
            nn.Conv2d(nc, ndf, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf) x 32 x 32
            nn.Conv2d(ndf, ndf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 2),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf*2) x 16 x 16
            nn.Conv2d(ndf * 2, ndf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 4),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf*4) x 8 x 8
            nn.Conv2d(ndf * 4, ndf * 8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 8),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf*8) x 4 x 4
            nn.Conv2d(ndf * 8, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
        )

    def forward(self, input):
        return self.main(input)
```

[Image credit: https://gluon.mxnet.io/chapter14_generative-adversarial-networks/dcgan.html](https://gluon.mxnet.io/chapter14_generative-adversarial-networks/dcgan.html)

Generator





Deconv 1

Deconv 2

Deconv 3

Deconv 4

```
# Generate batch of latent vectors
noise = torch.randn(b_size, nz, 1, 1, device=device)
```

Random numbers from normal distribution with mean 0 and variance 1

Tanh Normalize to [-1,1] Consistent with image pixel input range

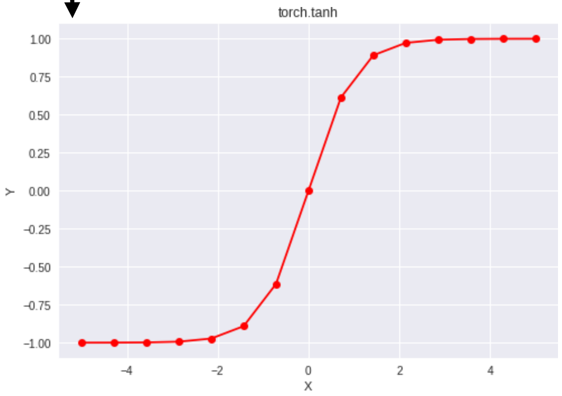
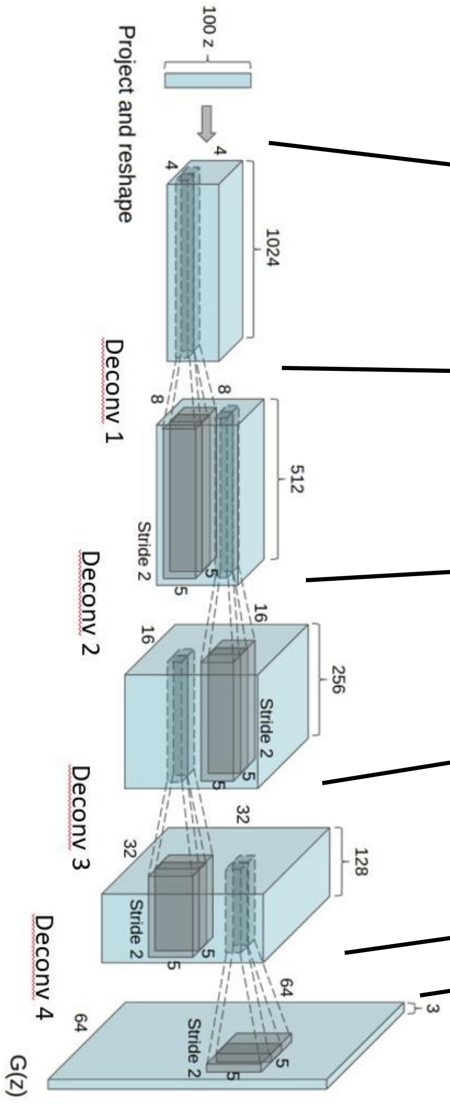


Image credit: https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

Generator – pytorch implementation



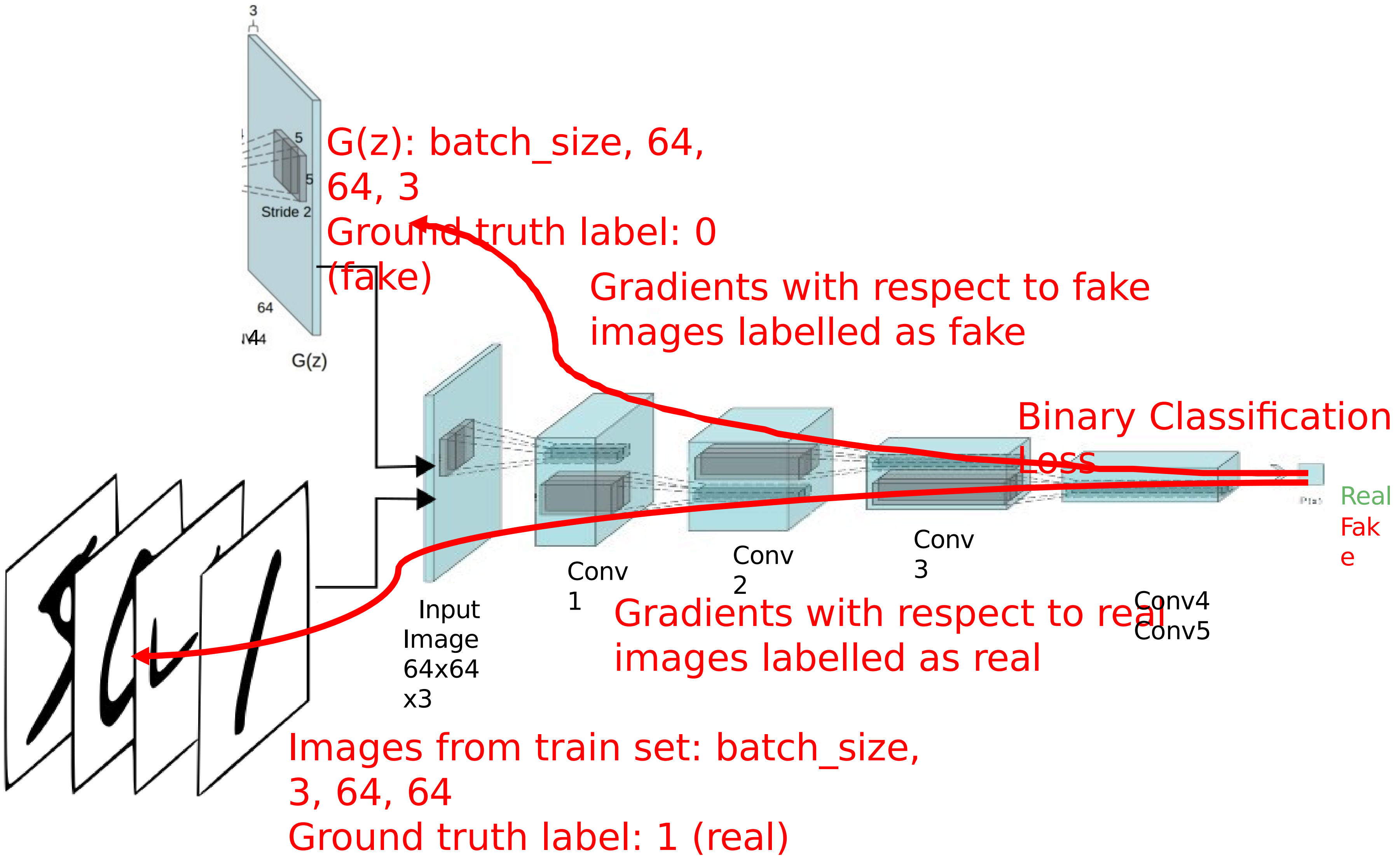

```
class Generator(nn.Module):
    def __init__(self, ngpu):
        super(Generator, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            # input is Z, going into a convolution
            nn.ConvTranspose2d( nz, ngf * 8, 4, 1, 0, bias=False),
            nn.BatchNorm2d(ngf * 8),
            nn.ReLU(True),
            # state size. (ngf*8) x 4 x 4
            nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 4),
            nn.ReLU(True),
            # state size. (ngf*4) x 8 x 8
            nn.ConvTranspose2d( ngf * 4, ngf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 2),
            nn.ReLU(True),
            # state size. (ngf*2) x 16 x 16
            nn.ConvTranspose2d( ngf * 2, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(True),
            # state size. (ngf) x 32 x 32
            nn.ConvTranspose2d( ngf, nc, 4, 2, 1, bias=False),
            nn.Tanh()
            # state size. (nc) x 64 x 64
        )

    def forward(self, input):
        return self.main(input)
```

Training GAN – Part 1

Training part 1: train

Generator VS Discriminator

Min-max Game

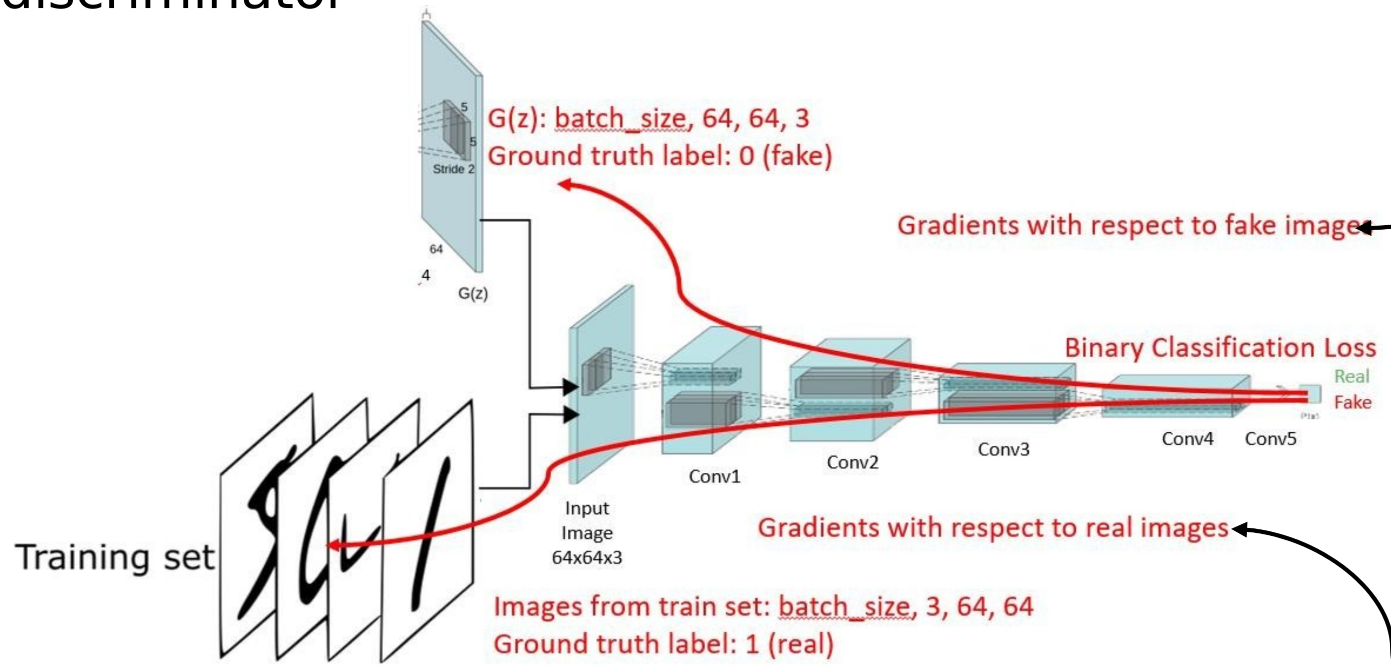


Training GAN – Part 1

Training part 1: train discriminator

Generator VS Discriminator

Min-max Game




```
## Train with all-fake batch
# Generate batch of latent vectors
noise = torch.randn(b_size, nz, 1, 1, device=device)
# Generate fake image batch with G
fake = netG(noise)
label.fill_(fake_label)
# Classify all fake batch with D
output = netD(fake.detach()).view(-1)
# Calculate D's loss on the all-fake batch
errD_fake = criterion(output, label)
# Calculate the gradients for this batch
errD_fake.backward()
D_G_z1 = output.mean().item()
# Add the gradients from the all-real and all-fake batches
errD = errD_real + errD_fake
# Update D
optimizerD.step()
```

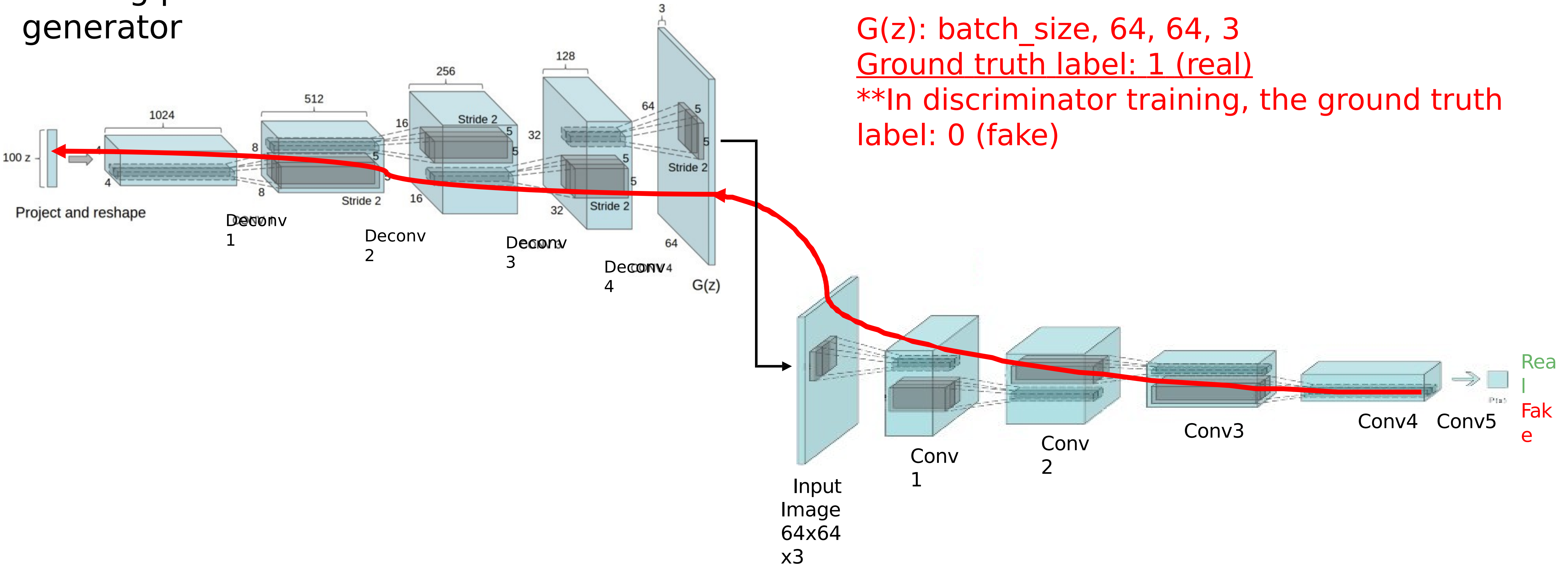
```
## Train with all-real batch
netD.zero_grad()
# Format batch
real_cpu = data[0].to(device)
b_size = real_cpu.size(0)
label = torch.full((b_size,), real_label, device=device)
# Forward pass real batch through D
output = netD(real_cpu).view(-1)
# Calculate loss on all-real batch
errD_real = criterion(output, label)
# Calculate gradients for D in backward pass
errD_real.backward()
D_x = output.mean().item()
```

Training GAN – Part 2

Generator Discriminator
Min-max Game

Training part 2: train generator

G(z): batch_size, 64, 64, 3
<u>Ground truth label: 1 (real)</u>
**In discriminator training, the ground truth label: 0 (fake)



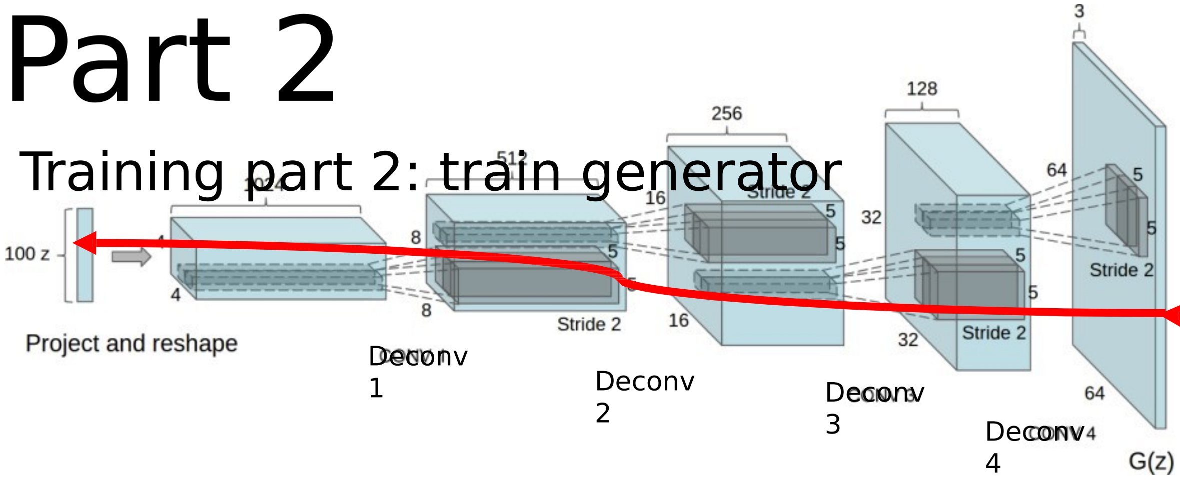
Gradients with respect to fake images but labelled as “real”

Training GAN – Part 2

Generator VS Discriminator

Min-max Game

Training part 2: train generator

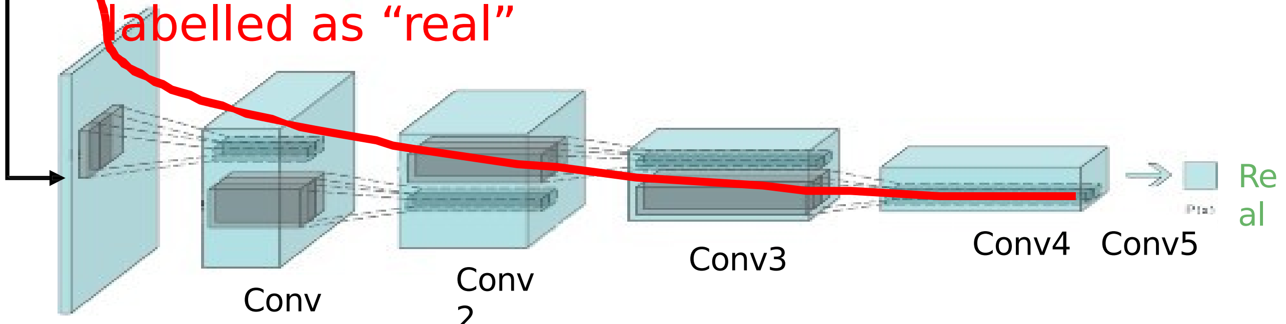


G(z): batch_size, 64, 64, 3

<u>Ground truth label: 1 (real)</u>

**In discriminator training, the ground truth label: 0 (fake)

Gradients with respect to fake images but labelled as “real”




```
##############################
# (2) Update G network: maximize log(D(G(z)))
##############################
netG.zero_grad()
label.fill_(real_label)  # fake labels are real for generator cost
# Since we just updated D, perform another forward pass of all-fake batch through D
output = netD(fake).view(-1)
# Calculate G's loss based on this output
errG = criterion(output, label)
# Calculate gradients for G
errG.backward()
D_G_z2 = output.mean().item()
# Update G
optimizerG.step()
```

GAN Zoo

StyleGAN

cGAN

ProgressiveGAN

CycleGAN

InforGAN

LapGAN

AC-GAN

BigBiGAN

BiGAN

StackGAN

EBGAN

BEGAN

WGAN

BigGAN

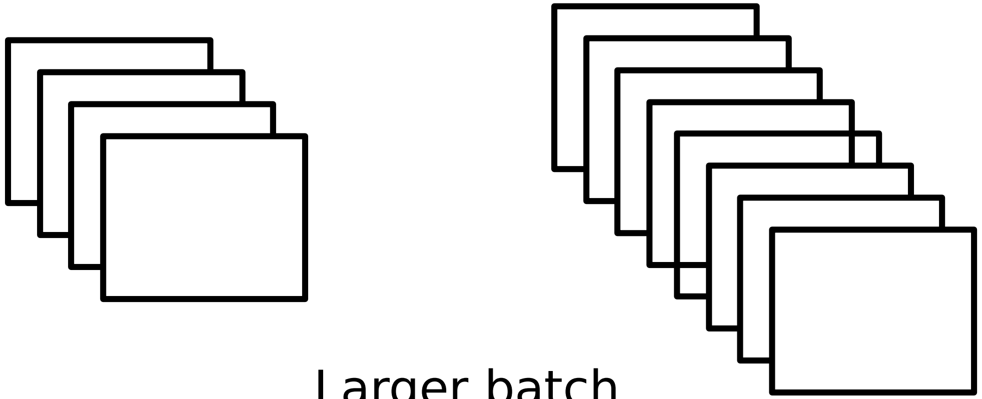
Problems with Deep Convolutional GAN (DCGAN)

- Generated images are very small, 64x64, 128x128
- One generator only corresponds with one class of images (no control
  over random vector z)

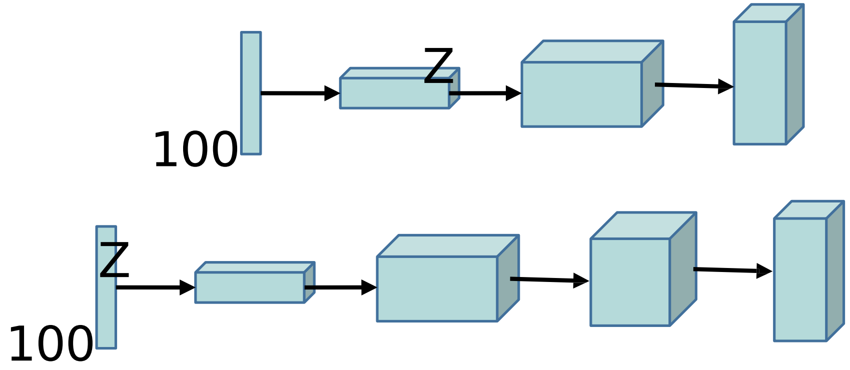
- Generated image quality is bad
- Training GAN is brittle:
- Non-convergence: Model parameter oscillate and never converge
- Model collapse: Produce limited number of samples
- Diminished gradients: discriminator is too perfect and generator always fails
- Highly sensitive to hyperparameters

BigBiGAN

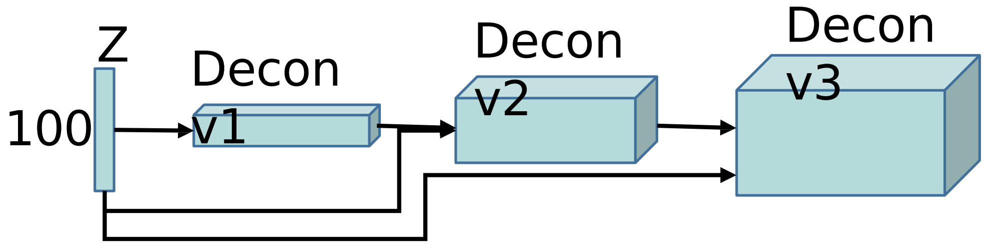
Larger batch size, more network parameters, network architecture designs (skip z-connections, self-attention)



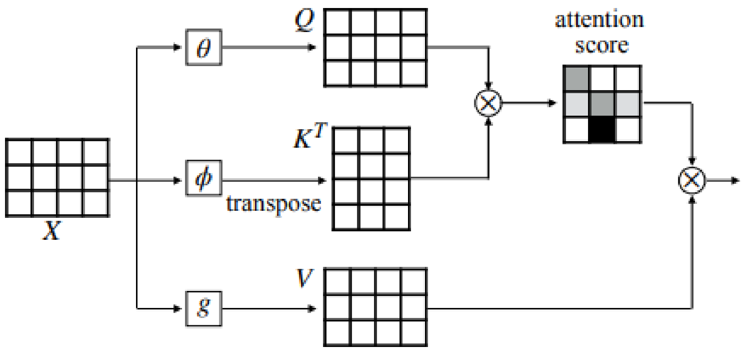
Larger batch sizes



More network parameters



Skip z-connections in generator

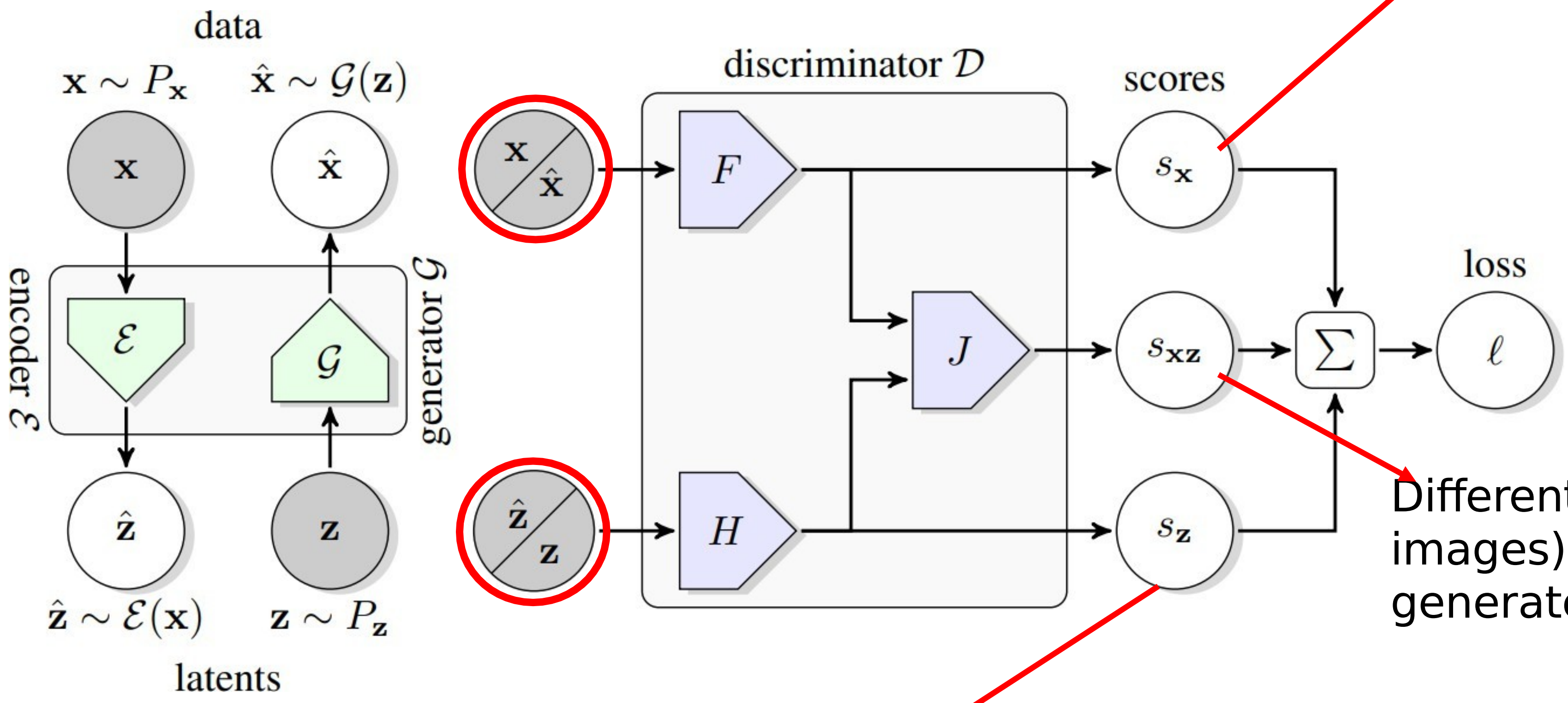


Self-attention Module in GAN

Donahue et al, Large Scale Adversarial Representation Learning, NIPS, 2019

BigBiGAN

Differentiate generated images from real images;



Differentiate pair (encoded z, real images) versus (randomly sampled z, generated images)

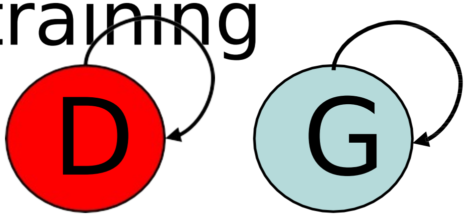
X: input image
Z: latent code

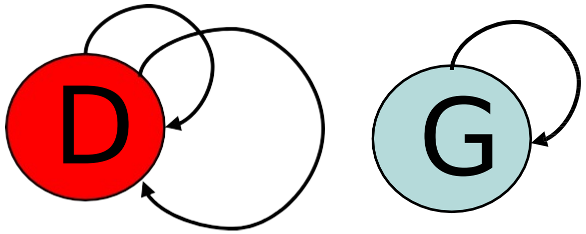
Differentiate encoded z from real images and randomly sampled z

Donahue etal, Large Scale Adversarial Representation Learning, NIPS, 2019

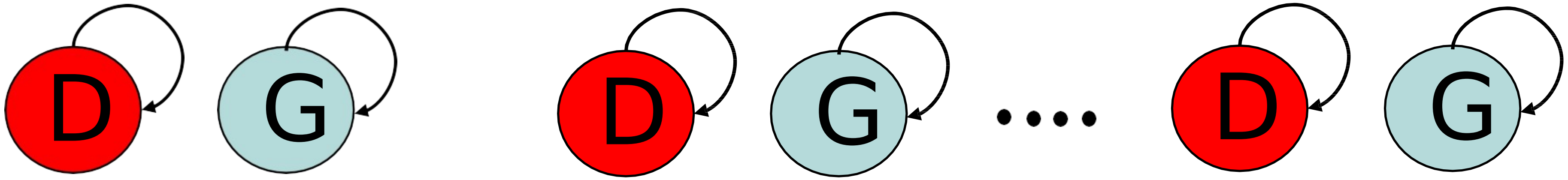
Tricks for More Realistic Image Reconstruction

- Update discriminators more often than generators during training





- Moving average of model weights (Progressive GAN)



After 1st epoch, Generator Parameter W_G1

After 2nd epoch, Generator Parameter W_G2

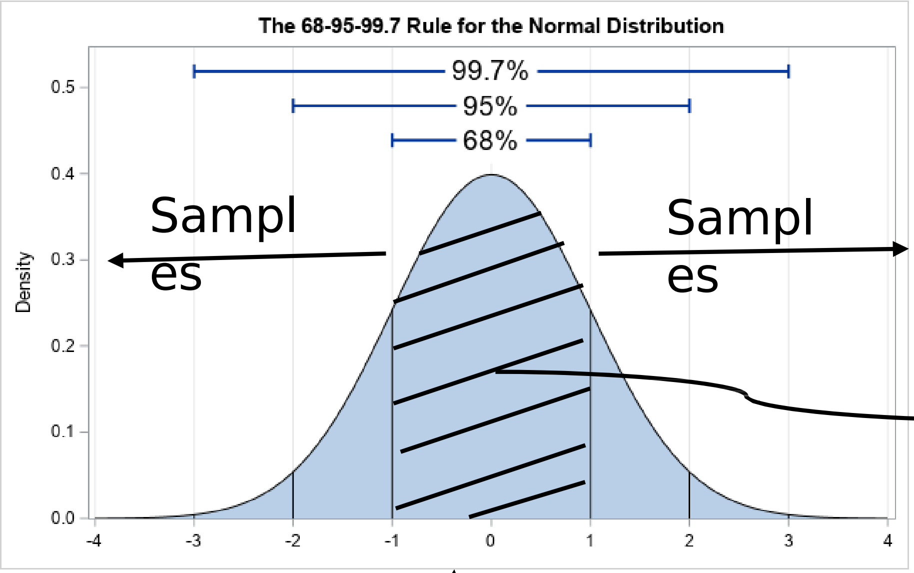
After Tth epoch, Generator Parameter W_GT

W_Gfinal = average(W_G1, W_G2, … ,W_GT)

https://machinelearningmastery.com/a-gentle-

Tricks for More Realistic Image Reconstruction

- Truncate z resampling at test stage

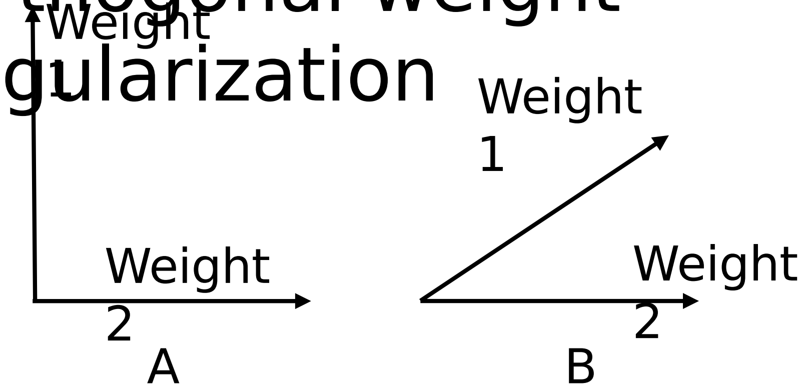


Sacrifice
sample
diversity

Z
10
0

```
# Generate batch of latent vectors
noise = torch.randn(b_size, nz, 1, 1, device=device)
```

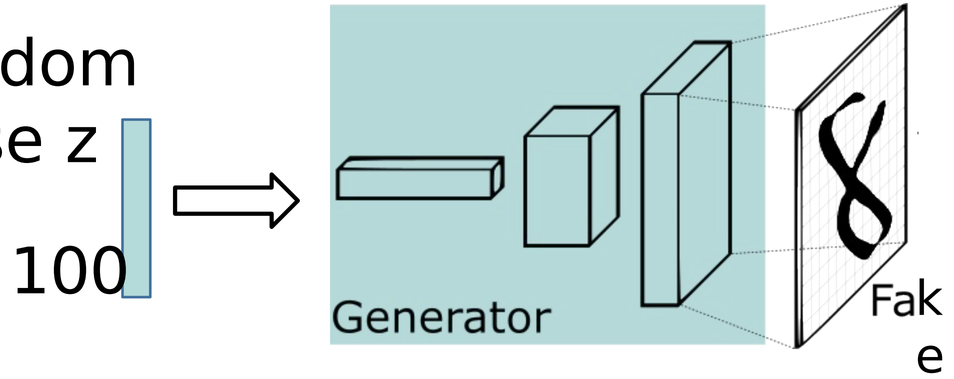
- Orthogonal weight initialization
- Orthogonal weight regularization



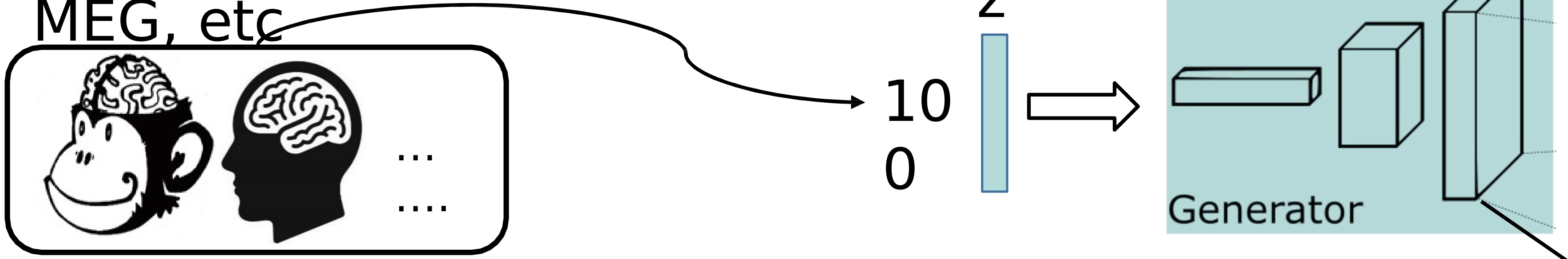
https://machinelearningmastery.com/a-gentle-

Brain Reading

Random noise z



Brain Codes from ECoG, fMRI, EEG, MEG, etc



All Animal species

Five senses: Smell, Images
Touch, Sound, Taste
High-level functionality: Emotions, Memories, Language

Image Reconstruction Methods from Brain Signals

Overview of Image Reconstruction Methods from Brain Signals

Gradient Backpropogation

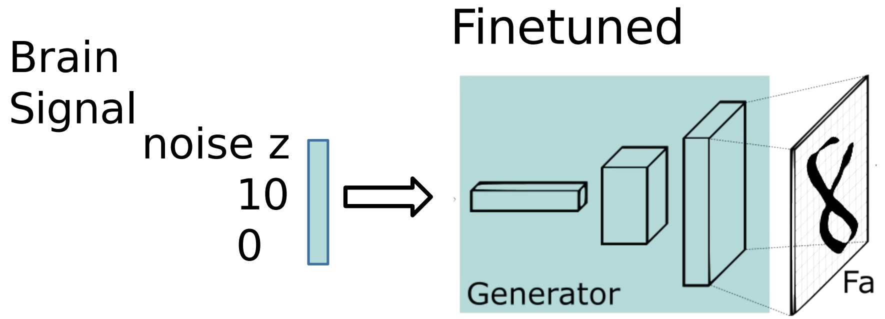
DeepDream

TextureSynthesis, StyleTransfer

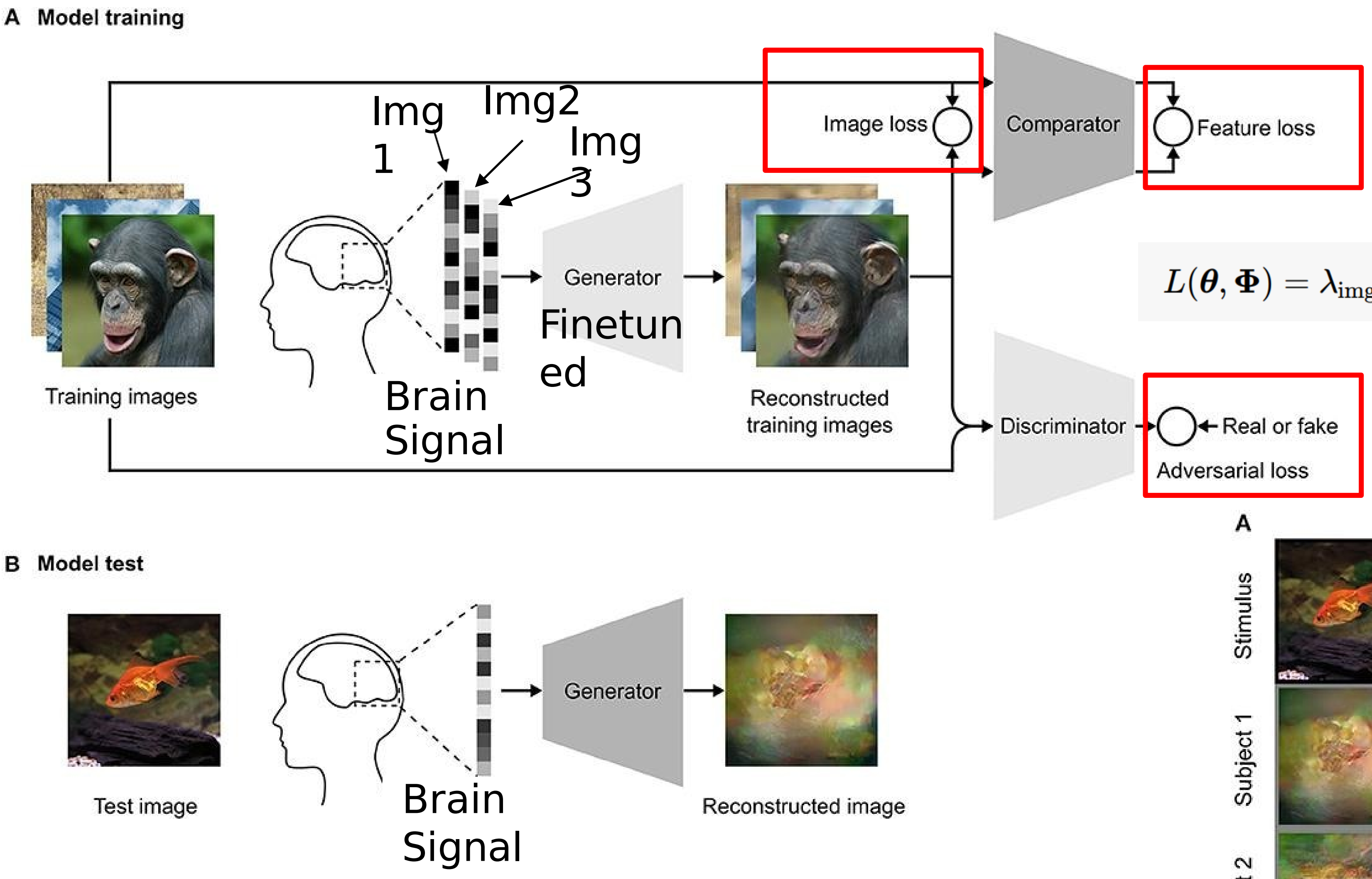
Image Reconstruction Methods from Brain Signals

Overview of Image Reconstruction Methods from Brain Signals

- Gradient Backpropogation (DeepDream, TextureSynthesis, StyleTransfer)
- Generative model (Latent code)
  - Learnable Generative Model (Model parameter is fine-tuned)

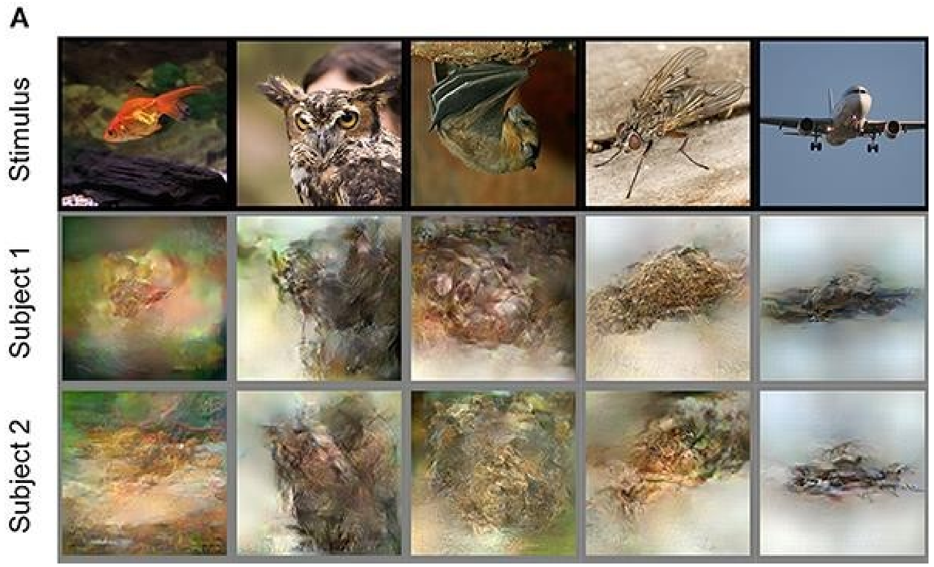


End-to-End Learnable Generative Model



fMRI

$$L(\boldsymbol{\theta}, \boldsymbol{\Phi}) = \lambda_{\mathrm{img}} L_{\mathrm{img}}(\boldsymbol{\theta}) + \lambda_{\mathrm{feat}} L_{\mathrm{feat}}(\boldsymbol{\theta}) + \lambda_{\mathrm{adv}} L_{\mathrm{adv}}(\boldsymbol{\theta}, \boldsymbol{\Phi})$$



Shen et al, Frontiers in Computational Neuroscience, 2019

https://openneuro.org/datasets/ds001506/versions/1.3.1

Image Reconstruction Methods from Brain Signals

Overview of Image Reconstruction Methods from Brain Signals

Gradient Backpropogation (DeepDream, TextureSynthesis, StyleTransfer)

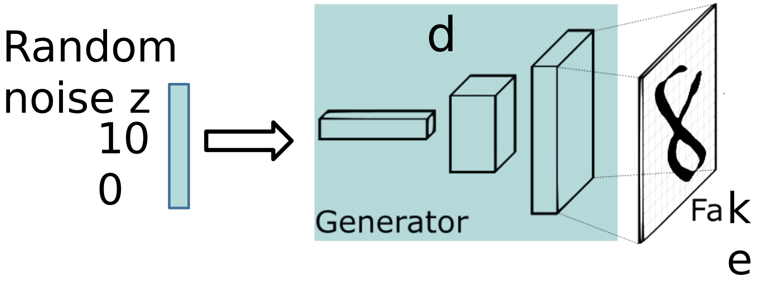
Generative model (Latent code)

Learnable Generative Model
(Model parameter is fine-tuned)

Fixed Generative Model (Model parameter is fixed)

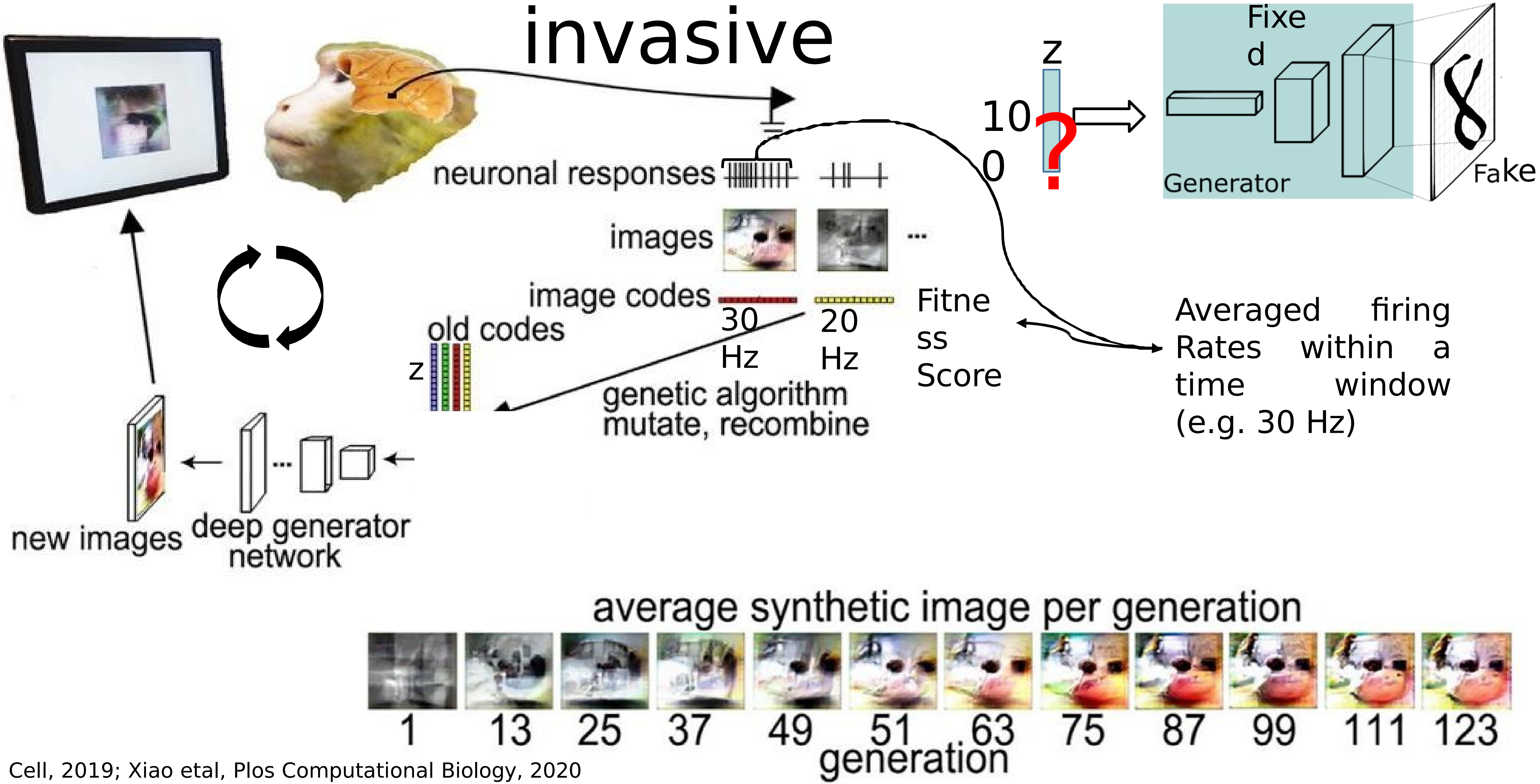
Genetic Algorithm

Fixed



Evolving Latent Code using Genetic Alg.

invasive

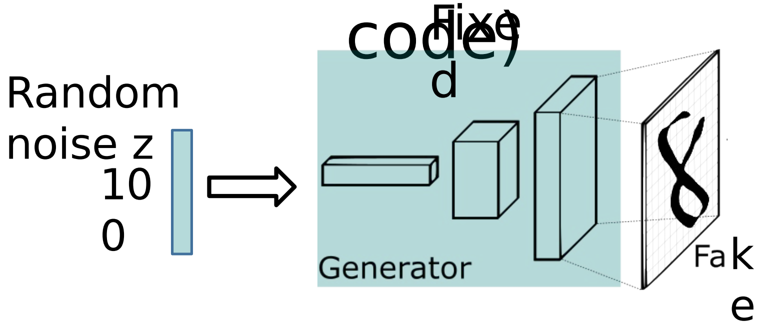


Ponce etal, Cell, 2019; Xiao etal, Plos Computational Biology, 2020

Image Reconstruction Methods from Brain Signals

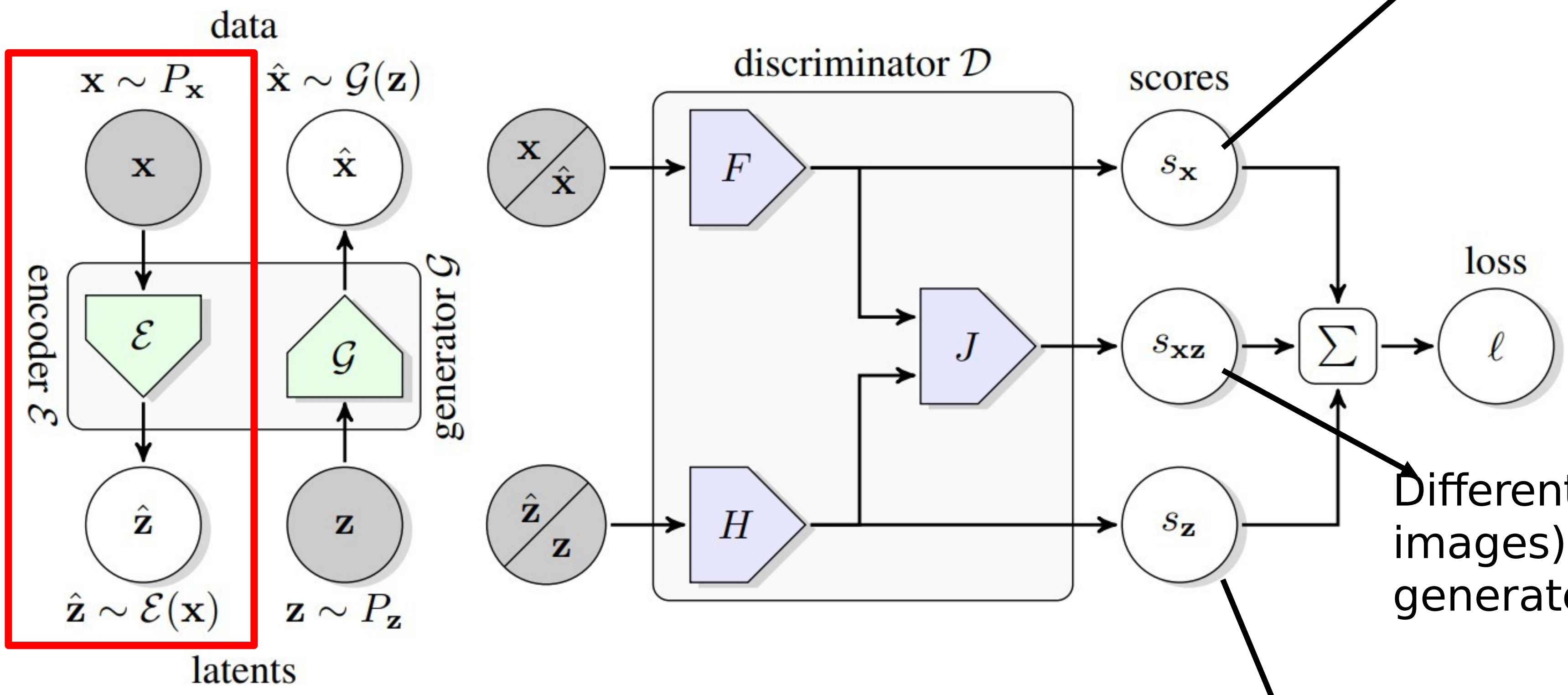
Overview of Image Reconstruction Methods from Brain Signals

- Gradient Backpropogation (DeepDream, TextureSynthesis, StyleTransfer)
- Generative model (Latent code)
  - Learnable Generative Model (Model parameter is fine-tuned)
  - Fixed Generative Model (Model parameter is fixed)
    - Genetic Algorithm
    - Linear Regression



BigBiGAN

Differentiate generated images from real images;



Differentiate pair (encoded z, real images) versus (randomly sampled z, generated images)

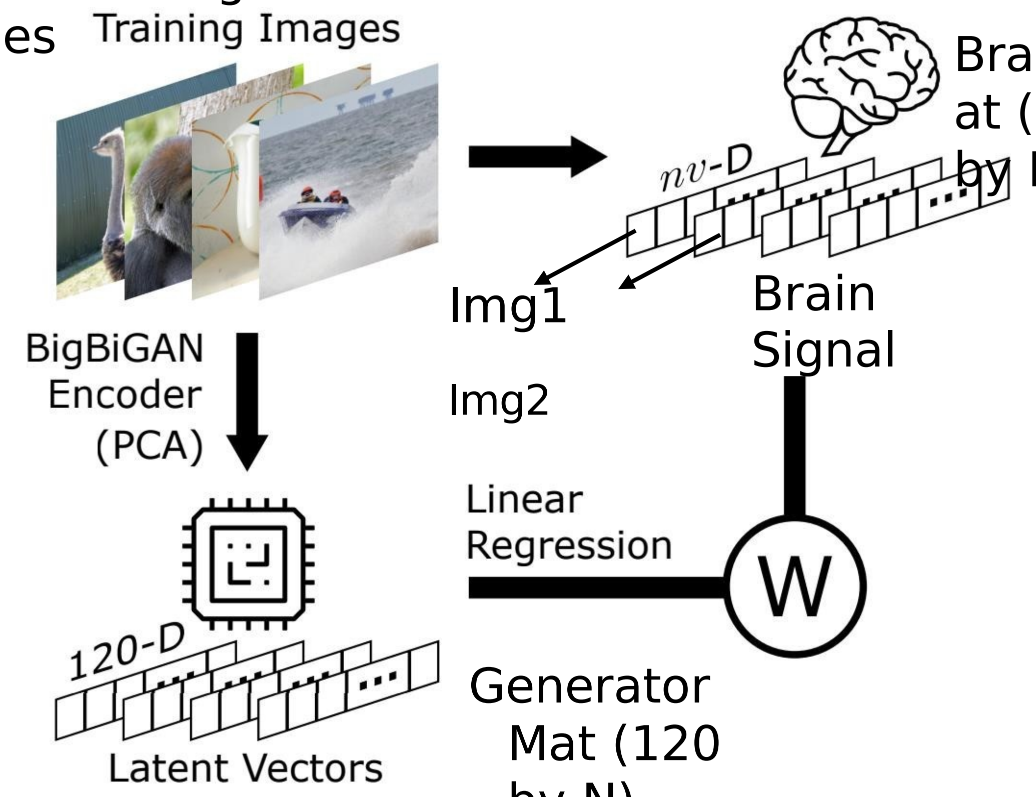
Differentiate encoded z from real images and randomly sampled z

X: input image
Z: latent code

Donahue etal, Large Scale Adversarial Representation

Mapping Latent Code using Linear Regression

N total training images



BrainMat (nv by N)

Img1

Brain Signal

Img2

Generator Mat (120 by N)

Training stage

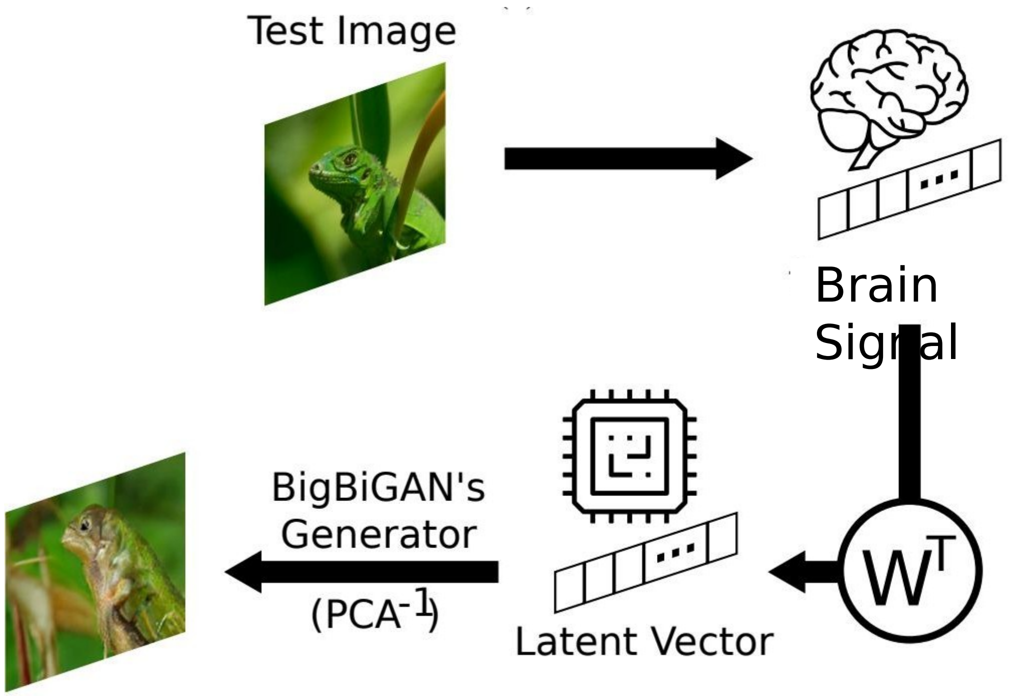
W (nv by 120) Generator Mat (120 by N) = BrainMat (nv by N)

Practice:

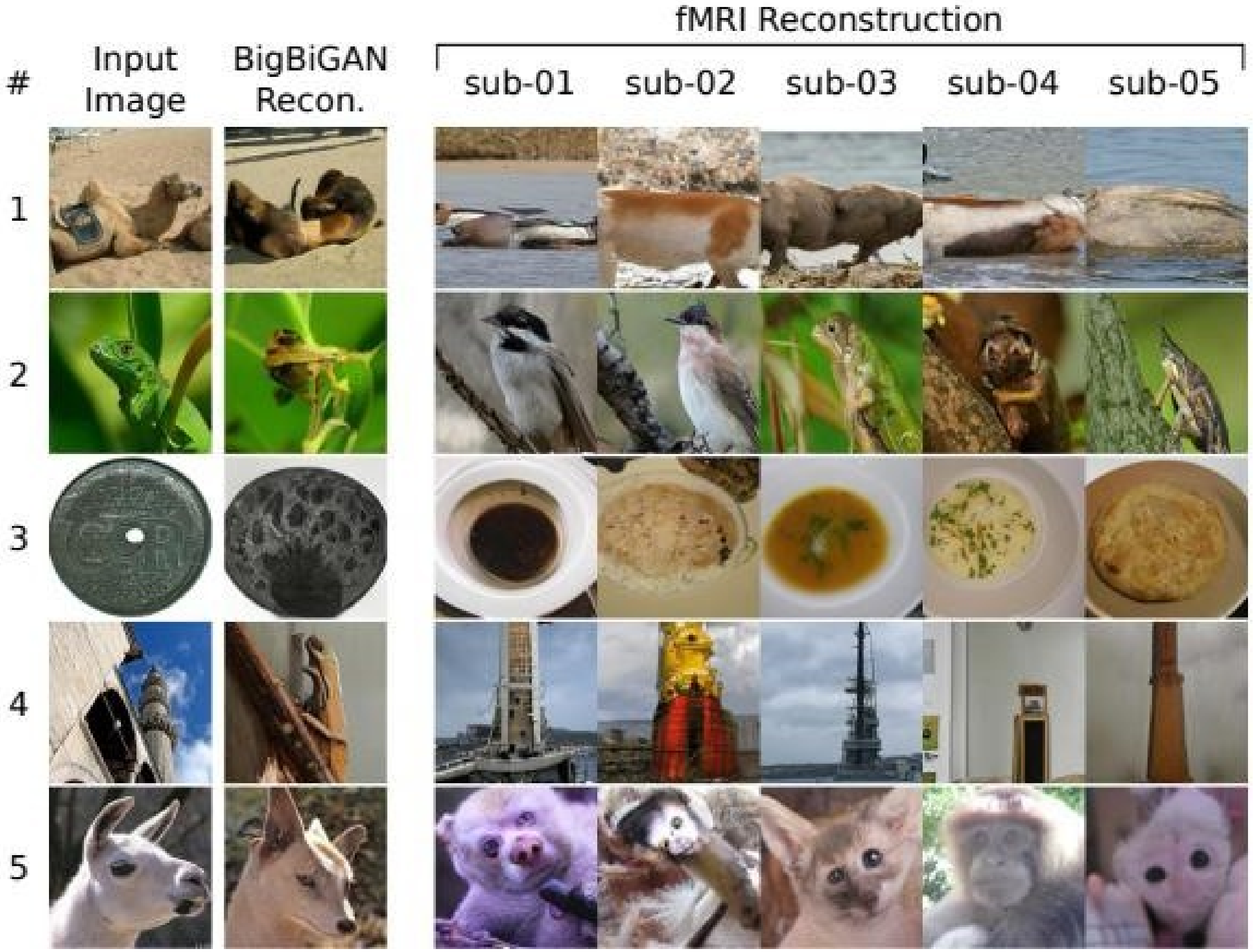
1. W might be invertable: psuedo-inverse
2. BrainMat size is too large: dimension reduction using PCA to pre-process the data
3. Latent code normalization

Mozafari etal, Reconstructing Natural Scenes from fMRI Patterns using BigBiGAN, 2020

Mapping using Linear Regression

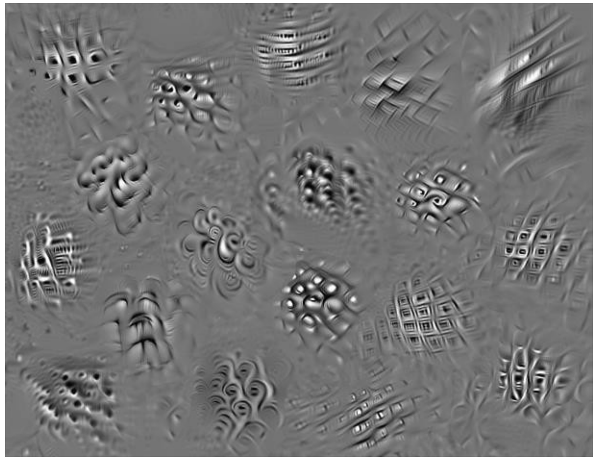


Testing
stage

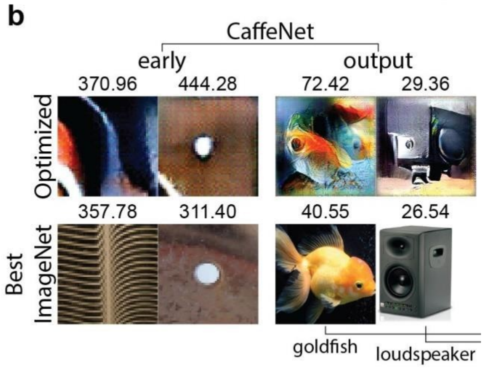


Mozafari et al, Reconstructing Natural Scenes from fMRI Patterns using BigBiGAN, 2020

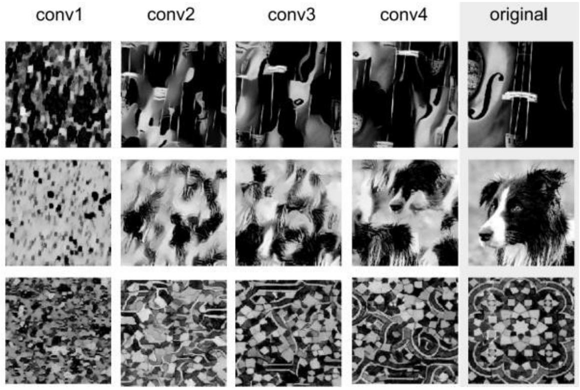
Which brain reader is better?



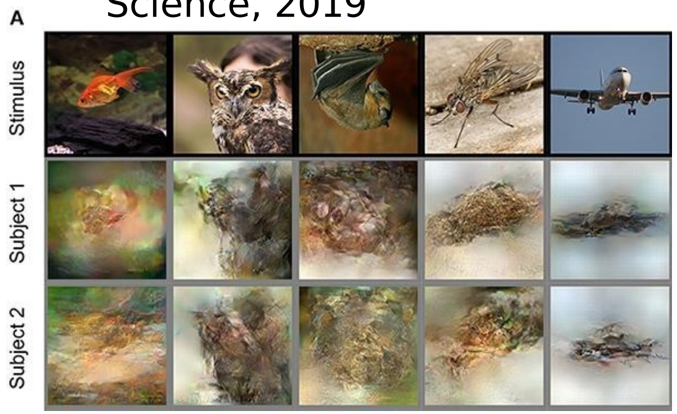
Bashivan etal, Science, 2019



Ponce etal, Cell, 2019;
Xiao etal, Plos Computational Biology, 2020

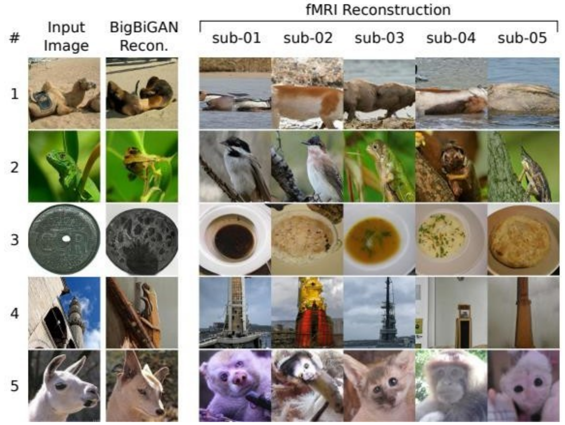


Cadena etal, Plos Computational Biology, 2019



Shen etal, Frontiers in Computational Neuroscience, 2019

Shen etal, Plos Computational Biology, 2019



Mozafari etal, arxiv, 2020

Do not be subjective. Use quantative metrics to evaluate image reconstruction quality

Towards Brain Computer Interface

Latent Codes In Generative Models

Brain Codes

Bijection Deep Brain Stimulation

Machine Codes

Latent Features In Recognition Models

Brain Machine Interface (BCI) for the visually impaired

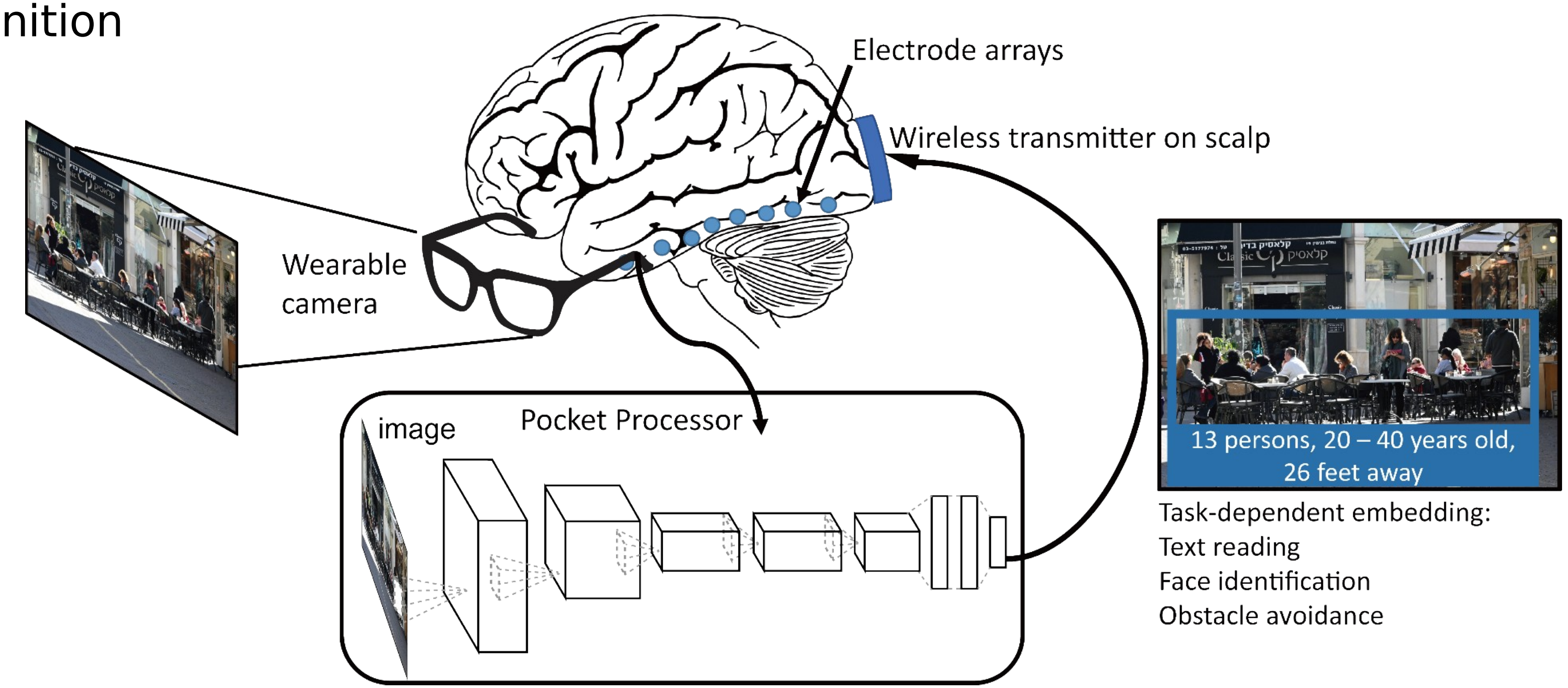
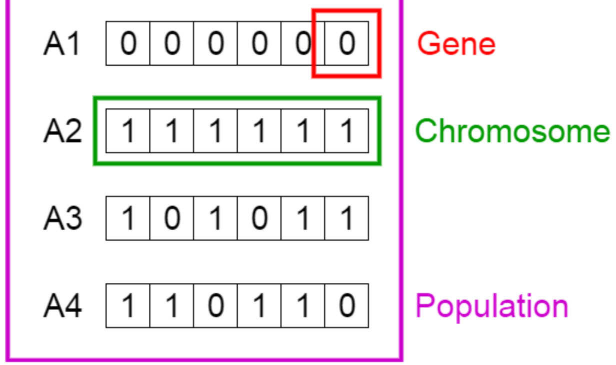


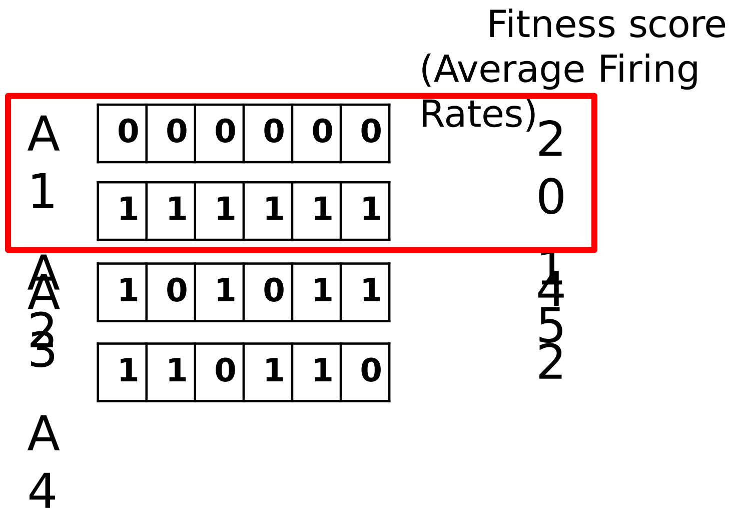
Image credit: Xiao etal, unpublished figure, book chapter on AI and vision

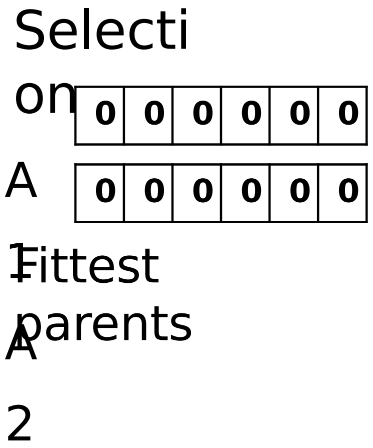
Preliminaries on Genetic Algorithm

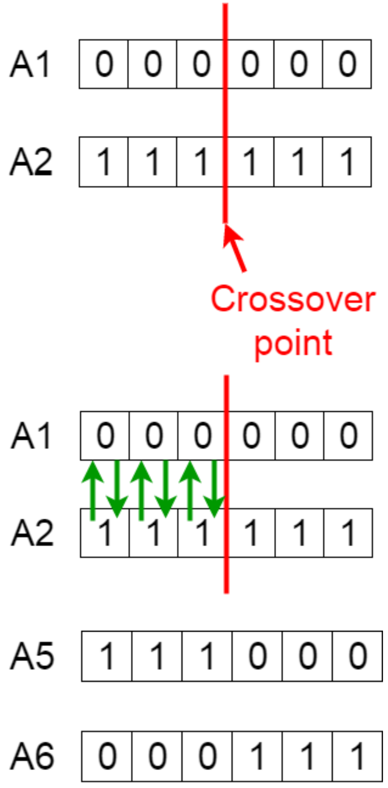
Natural selection process where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. -> the fittest latent code z which drives neurons to fire as much as possible



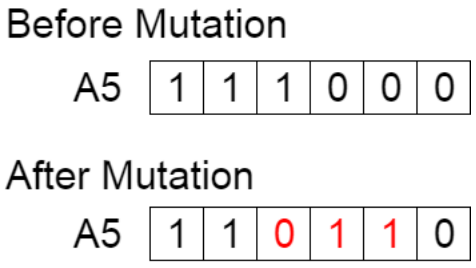
Population, Chromosomes and Genes







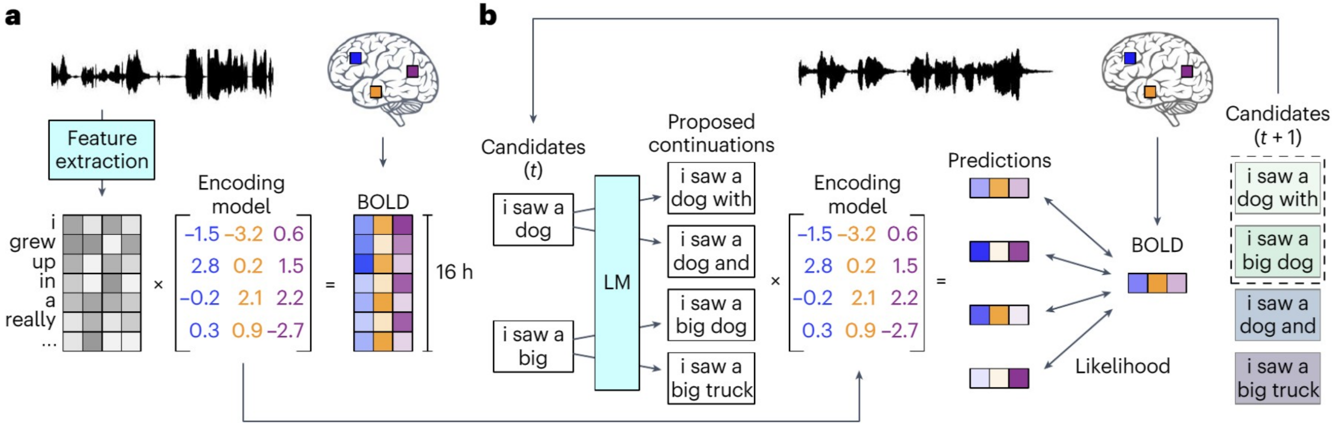
New offspring



Mutation: Before and After

```
START
Generate the initial population
Compute fitness
REPEAT
    Selection
    Crossover
    Mutation
    Compute fitness
UNTIL population has converged
STOP
```

“SEMANTIC RECONSTRUCTION OF CONTINUOUS LANGUAGE FROM NON-INVASIVE BRAIN RECORDINGS” TANG ET AL. NATURE NEUROSCIENCE 2023



c

Actual stimulus	Decoded stimulus
i got up from the air mattress and pressed my face against the glass of the bedroom window expecting to see eyes staring back at me but instead finding only darkness	i just continued to walk up to the window and open the glass i stood on my toes and peered out i didn’t see anything and looked up again i saw nothing
i didn't know whether to scream cry or run away instead i said leave me alone i don't need your help adam disappeared and i cleaned up alone crying	started to scream and cry and then she just said i told you to leave me alone you can't hurt me anymore i'm sorry and then he stormed off i thought he had left i started to cry
that night i went upstairs to what had been our bedroom and not knowing what else to do i turned out the lights and lay down on the floor	we got back to my dorm room i had no idea where my bed was i just assumed i would sleep on it but instead i lay down on the floor
i don't have my driver's license yet and i just jumped out right when i needed to and she says well why don't you come back to my house and i'll give you a ride i say ok	she is not ready she has not even started to learn to drive yet i had to push her out of the car i said we will take her home now and she agreed

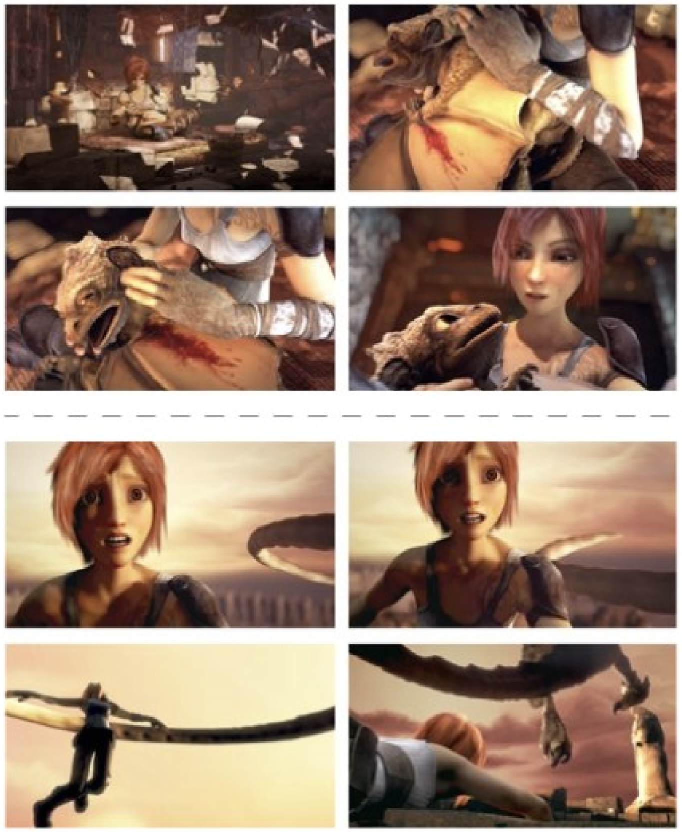
Exact

Gist

Error

c

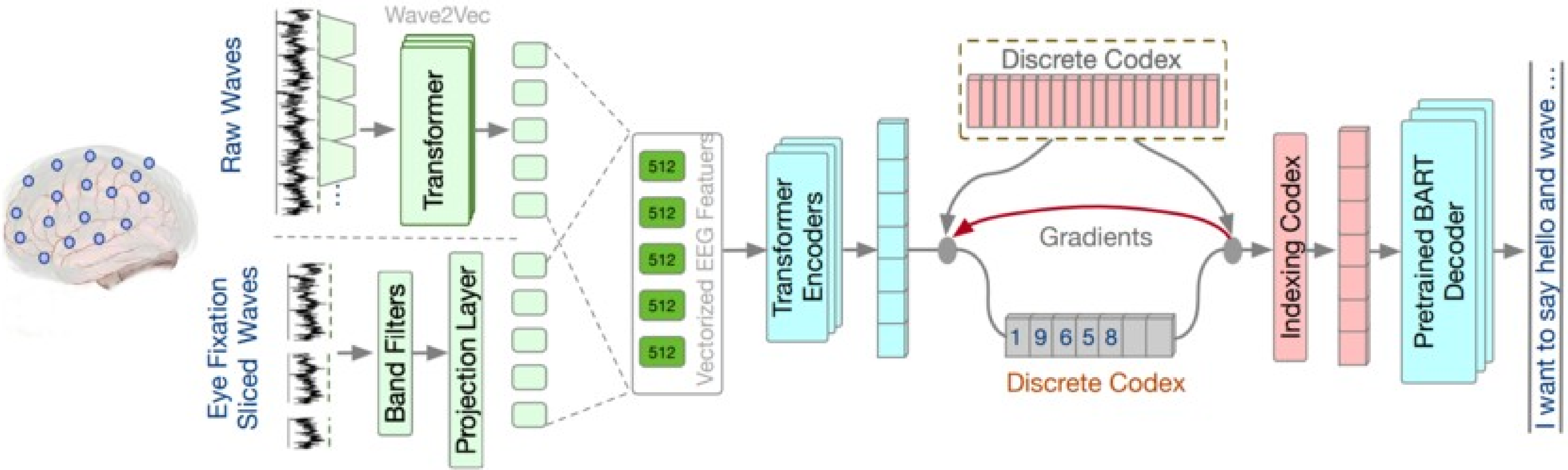
Actual stimulus | Decoded



she was very weak i held her neck to get her breathing under control

i see a girl that looks just like me get hit on her back and then she is knocked off

"DeWave: Discrete #EEG Waves Encoding for #Brain Dynamics to Text Translation"
https://arxiv.org/abs/2309.14030



Questions?