

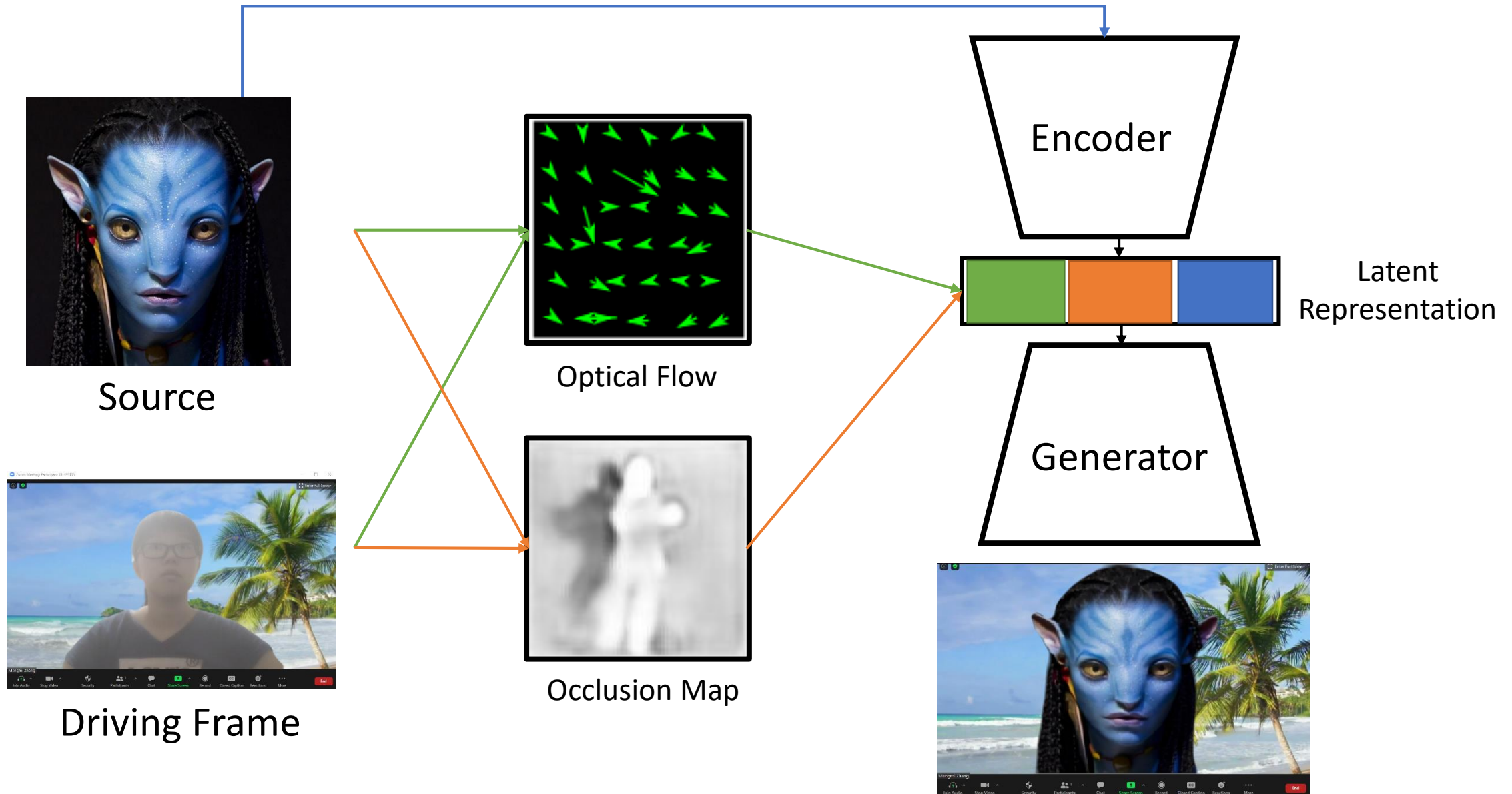
Toward Brain Computer Interface: Deep Generative Models for Brain Reading

Mengmi Zhang

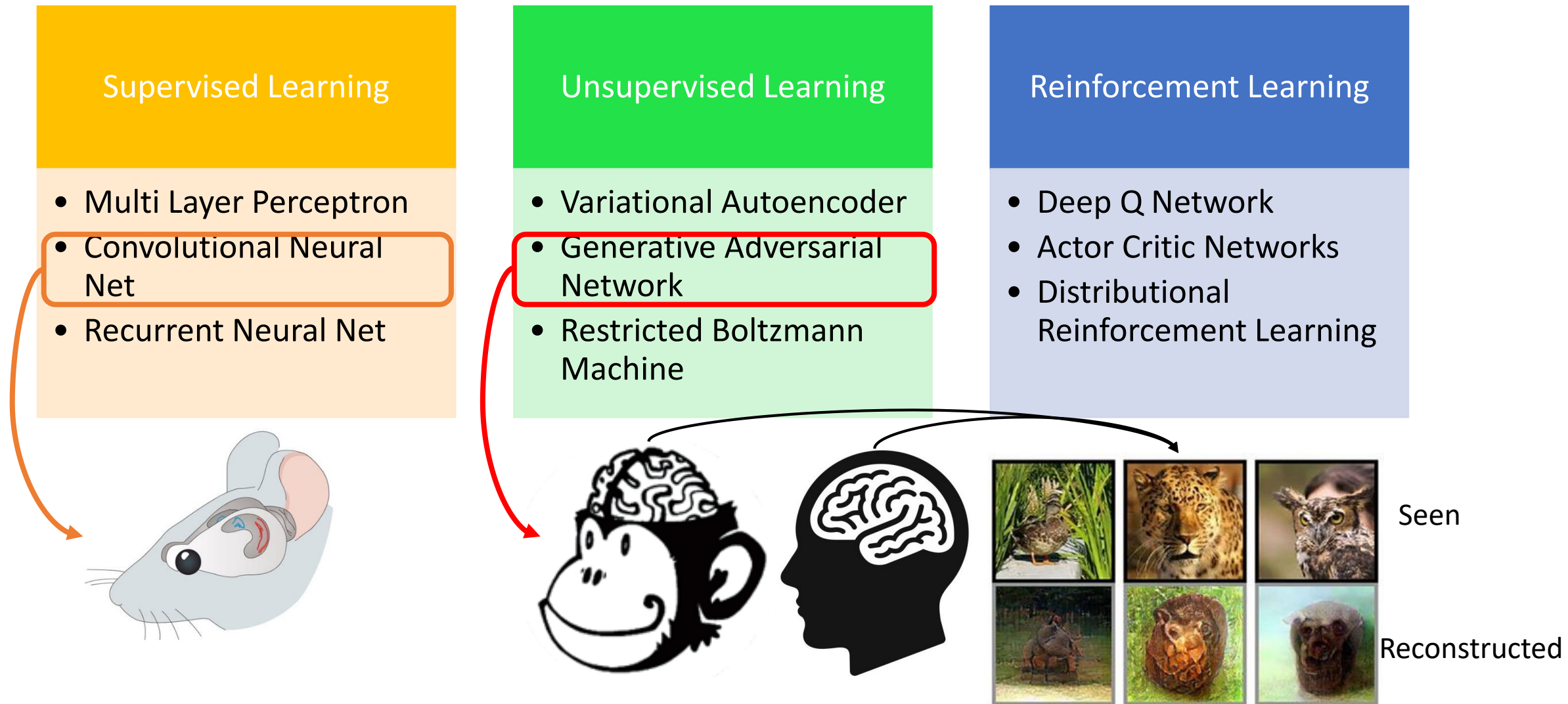
Mengmi.Zhang@childrens.harvard.edu

CBMM Summer Course 2020

DeepFake in Zoom



Deep Learning in Machine Learning



Outline

Deep Convolutional Generative Adversarial Network (DCGAN)

BigBiGAN

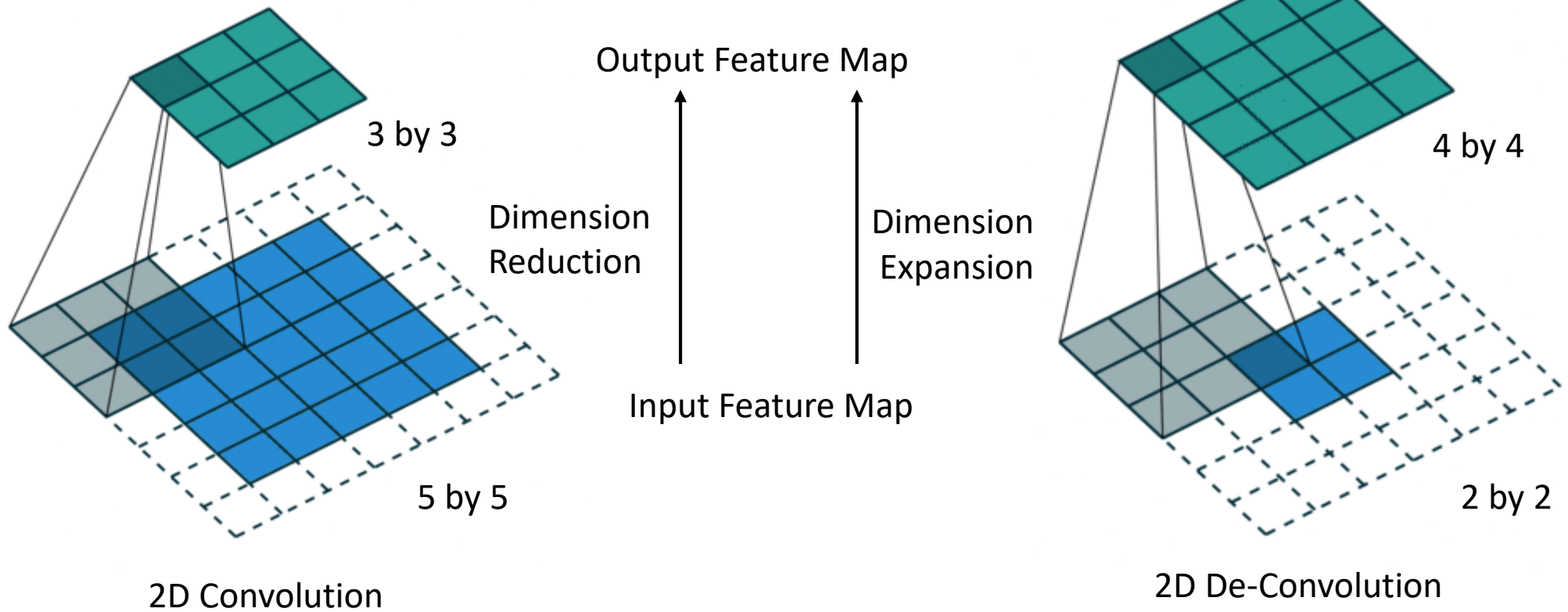
Tricks for more realistic image construction using GANs

Image reconstruction from brain signals

Evaluation of brain readers

Towards Brain Computer Interface (BCI)

Deconvolution



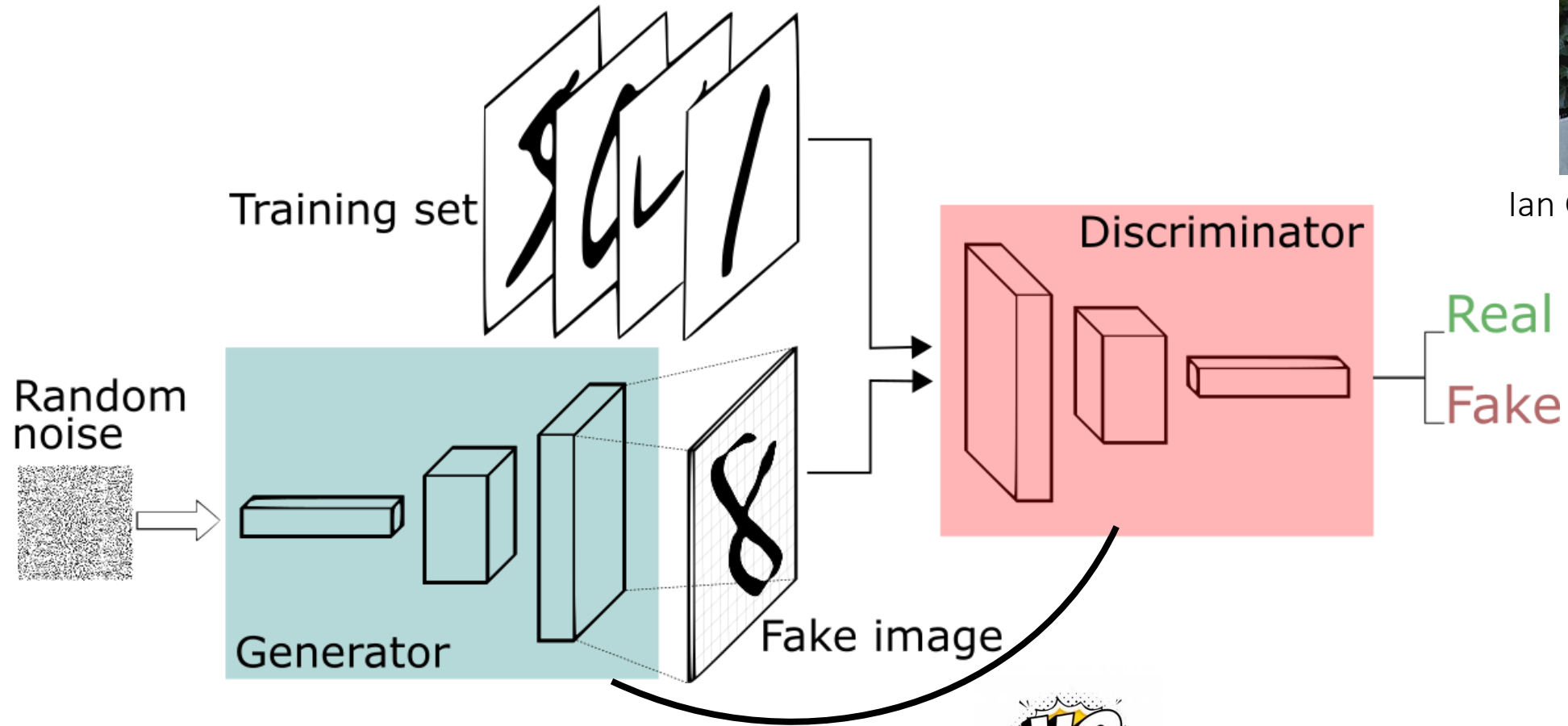
Pytorch: `torch.nn.ConvTranspose2d(in_channels, out_channels, kernel_size, stride, padding)`

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

Generative Adversarial Networks



Ian Goodfellow, 2014



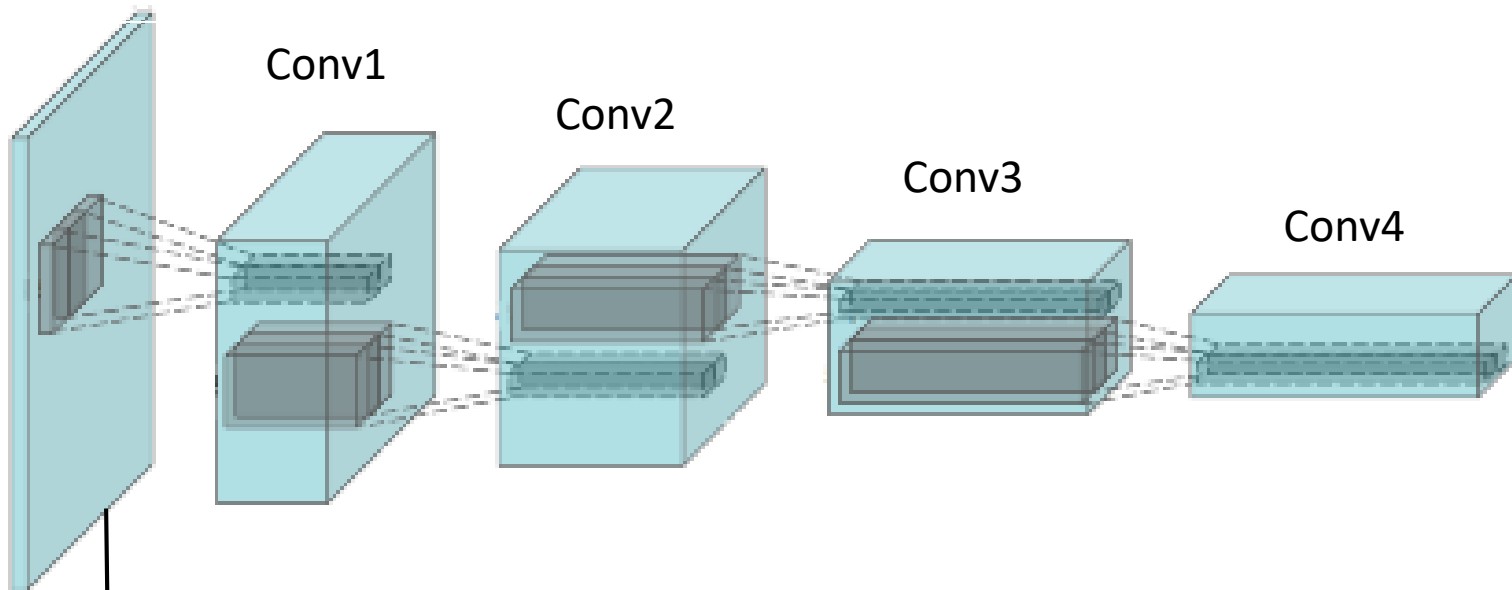
Generator **VS** Discriminator

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

Deep Convolutional GAN (DCGAN)

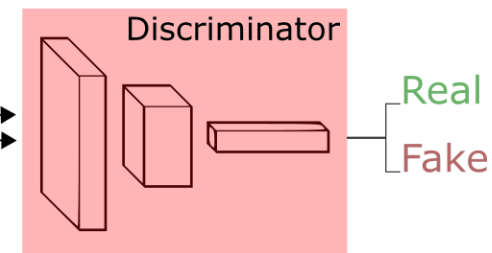
Discriminator

Input Image
64x64x3



Fake Image
64x64x3

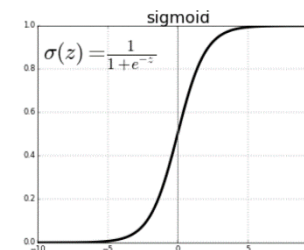
Real Image
64x64x3



Conv5

0.2	Real
0.8	Fake

Sigmoid



Normalize to [0,1]

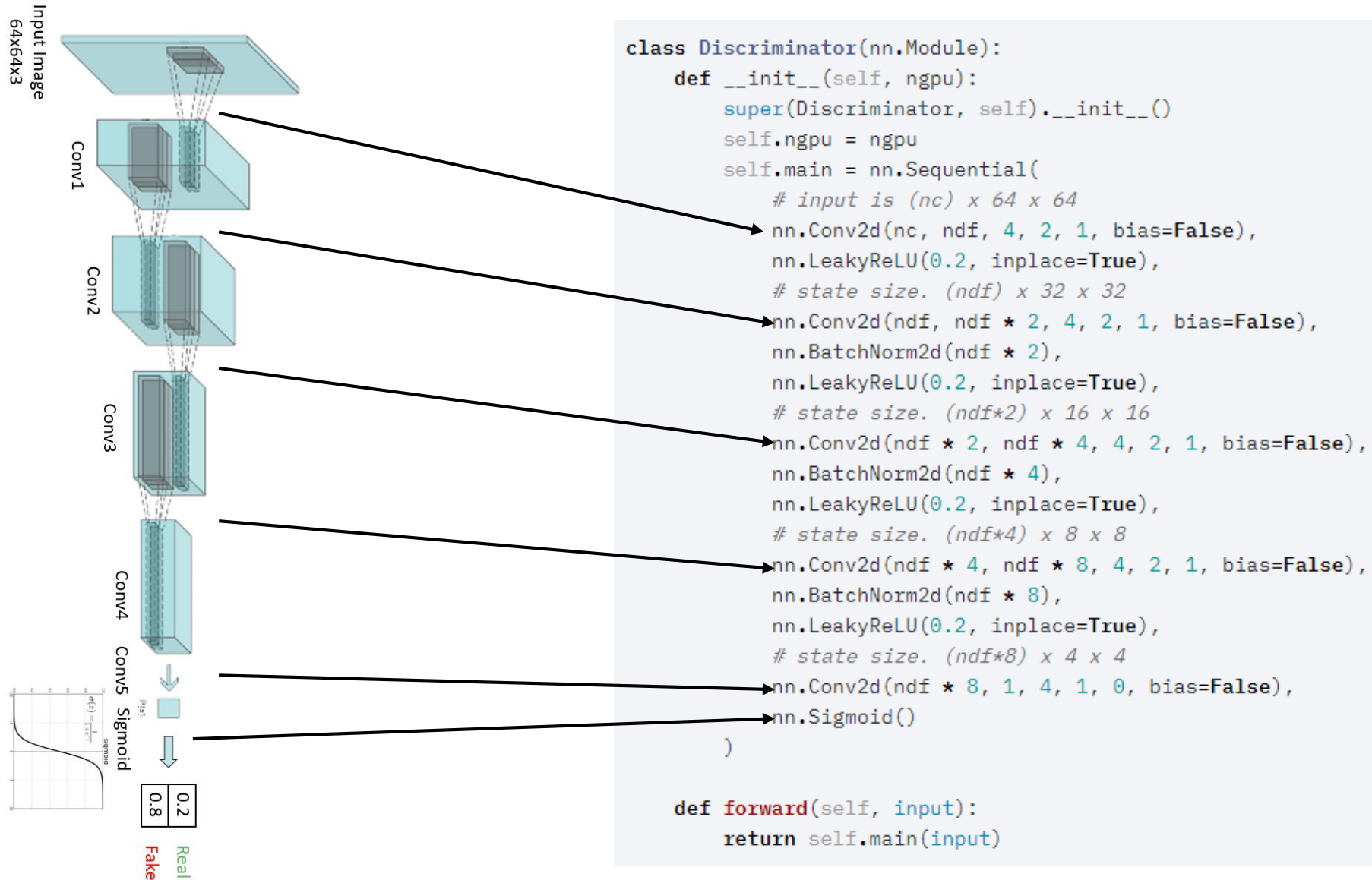
Create the dataset

```
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)),
                           ]))
```

Image augmentation:
scaling, rotation, random cropping, etc

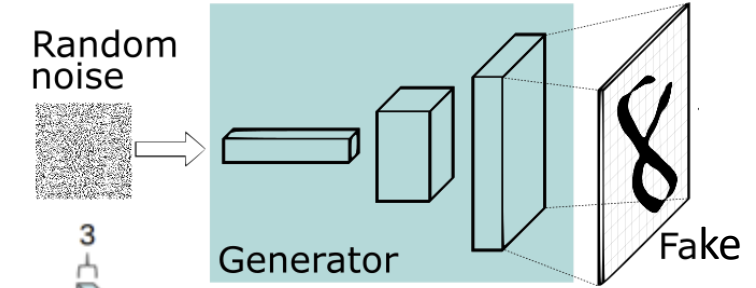
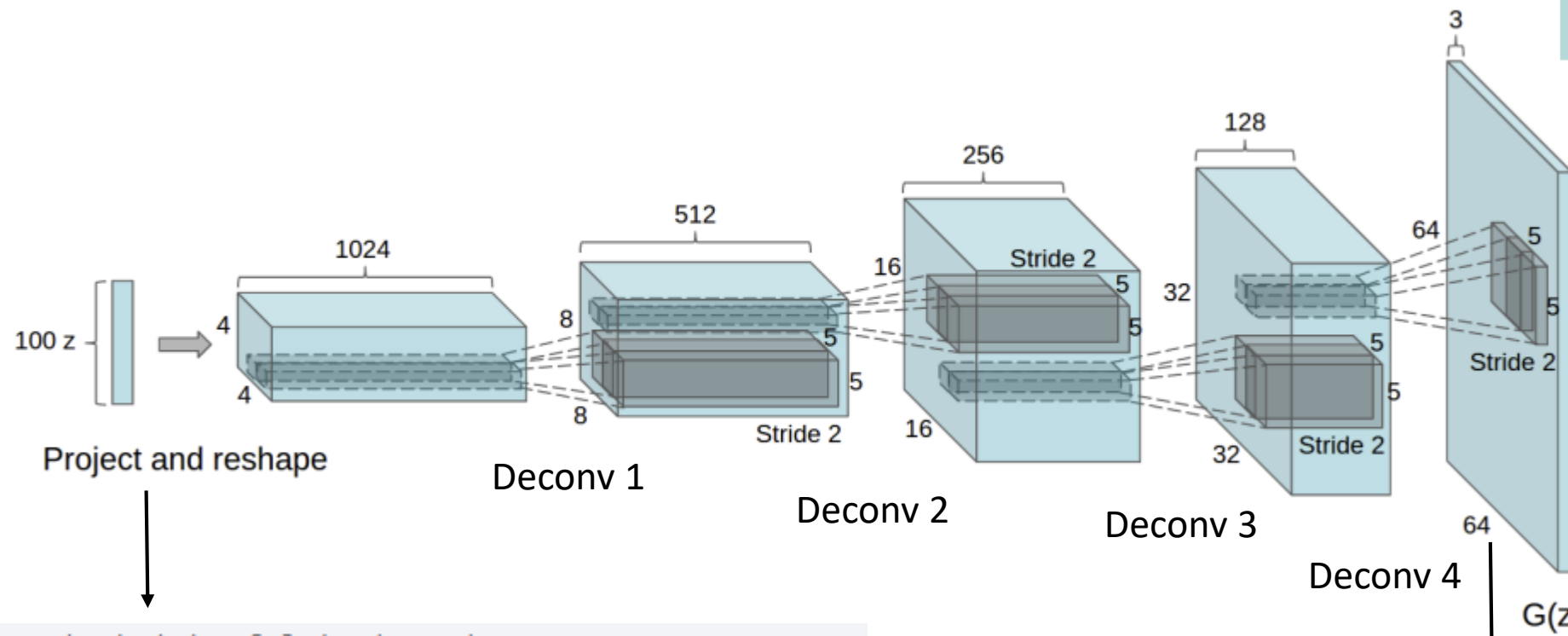
Normalize image pixel value from [0,1] to [-1,1]

Discriminator – pytorch implementation



Deep Convolutional GAN (DCGAN)

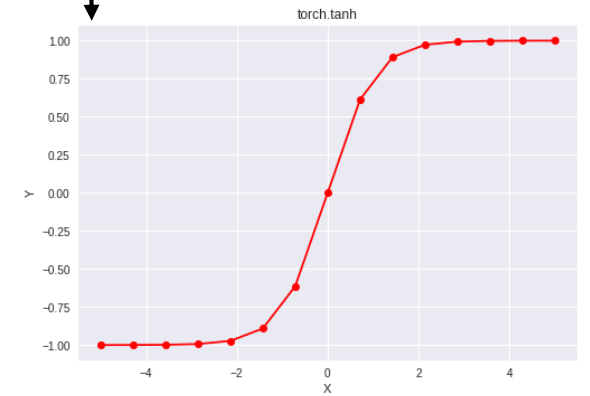
Generator



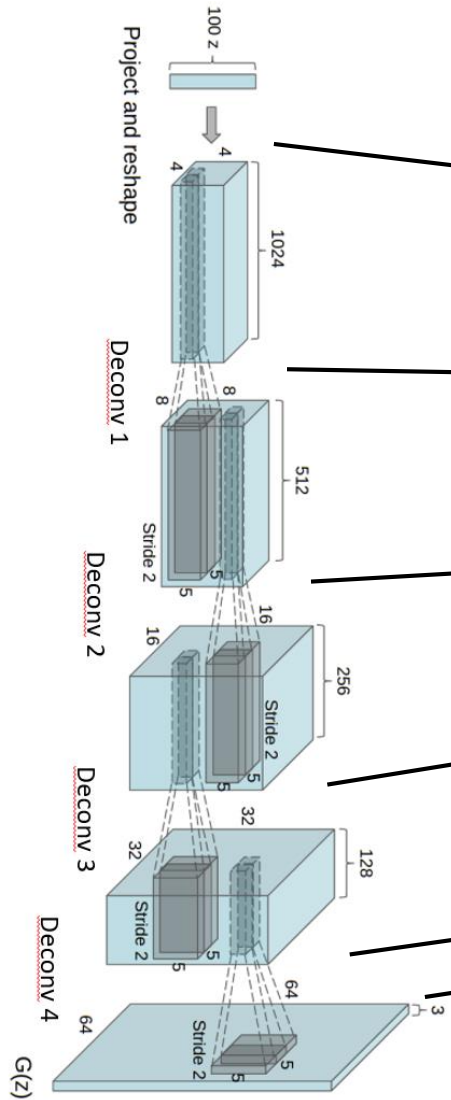
```
# Generate batch of latent vectors  
noise = torch.randn(b_size, nz, 1, 1, device=device)
```

Random numbers from normal distribution with mean 0 and variance 1

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



Generator – pytorch implementation



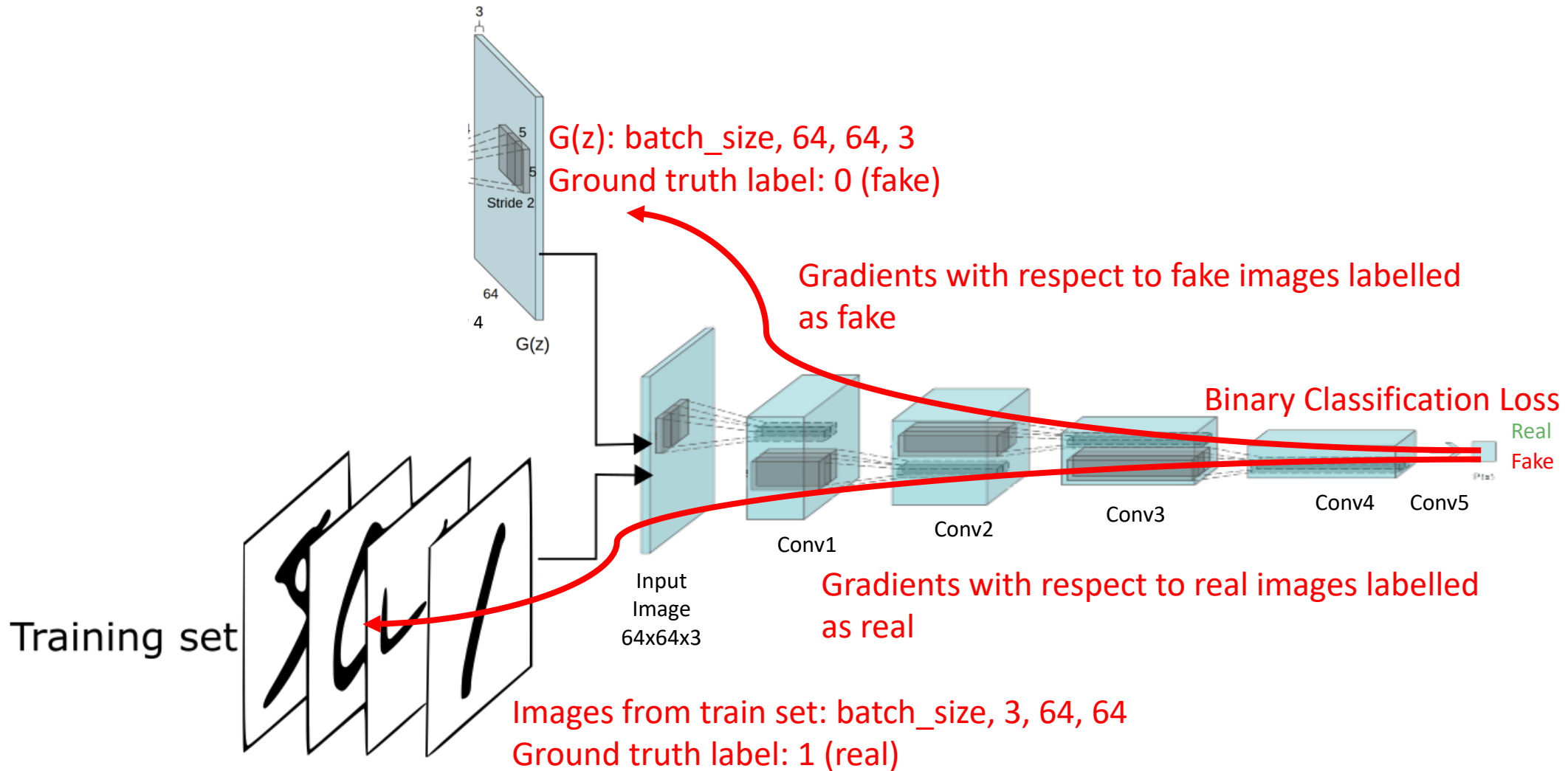
```
class Generator(nn.Module):
    def __init__(self, ngpu):
        super(Generator, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            # input is Z, going into a convolution
            nn.ConvTranspose2d( nz, ngf * 8, 4, 1, 0, bias=False),
            nn.BatchNorm2d(ngf * 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

Generator **VS** Discriminator
Min-max Game

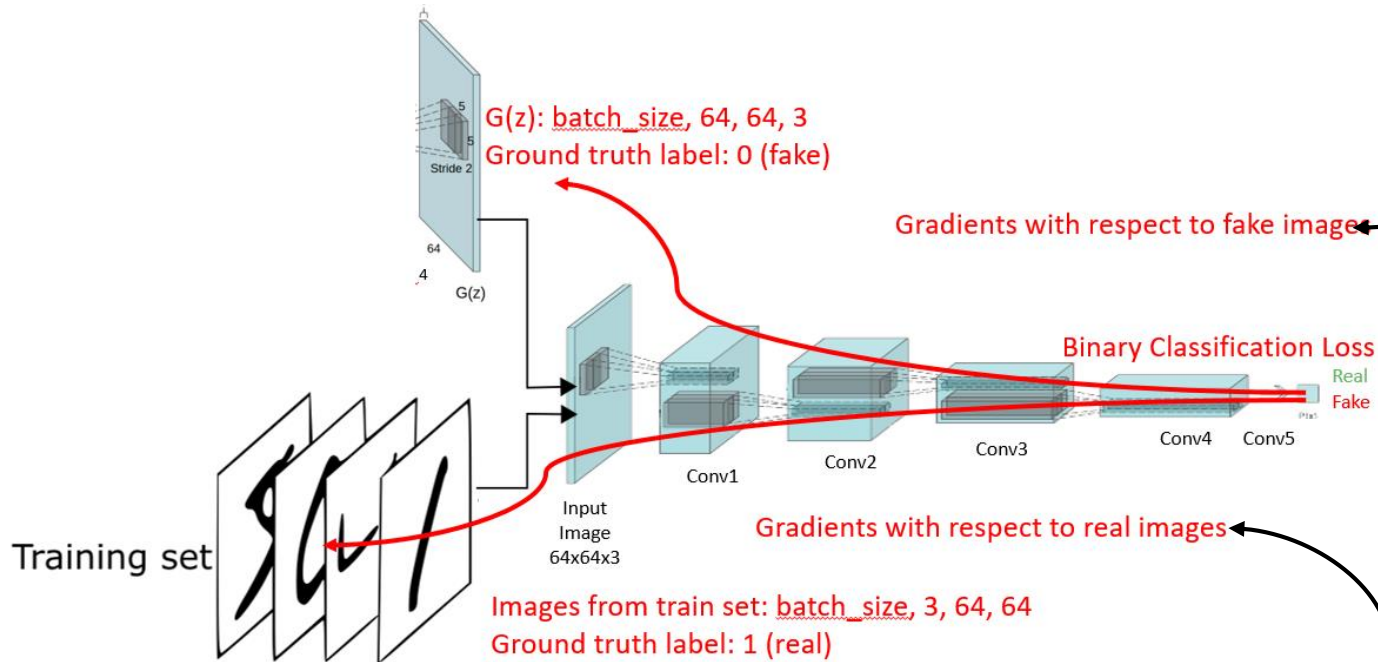
Training part 1: train discriminator



Training GAN – Part 1

Generator Discriminator Min-max Game

Training part 1: train discriminator



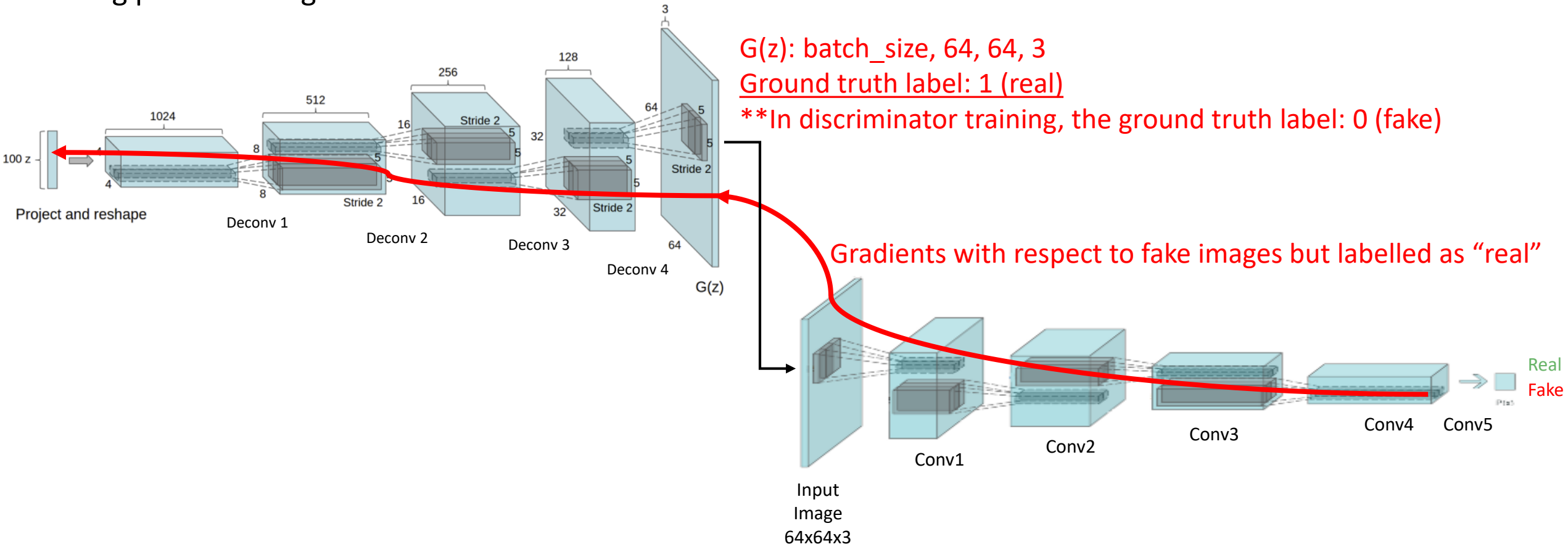
```
## Train with all-fake batch
# Generate batch of latent vectors
noise = torch.randn(b_size, nz, 1, 1, device=device)
# Generate fake image batch with G
fake = netG(noise)
label.fill_(fake_label)
# Classify all fake batch with D
output = netD(fake.detach()).view(-1)
# Calculate D's loss on the all-fake 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 **VS** Discriminator
Min-max Game

Training part 2: train generator

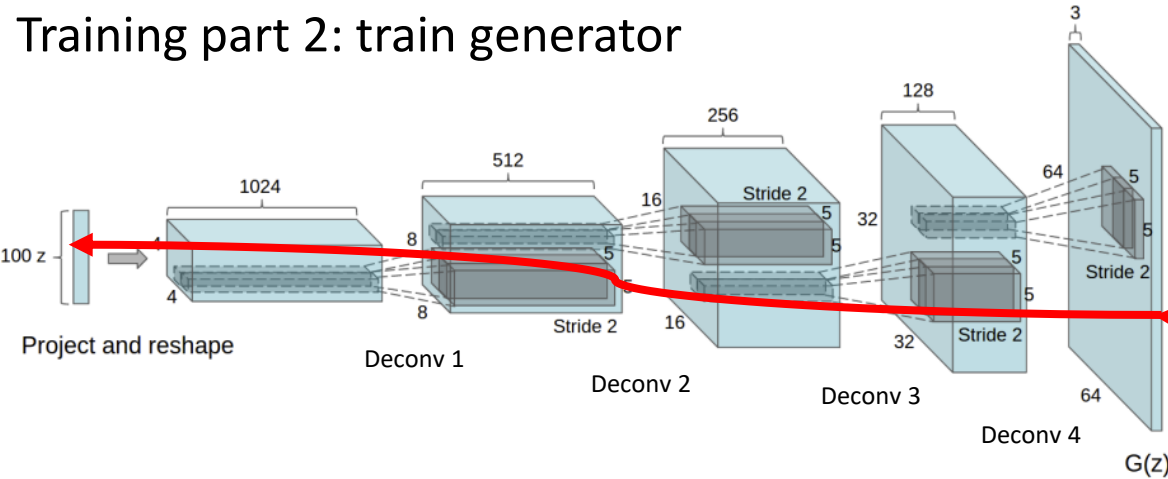


Training GAN – Part 2

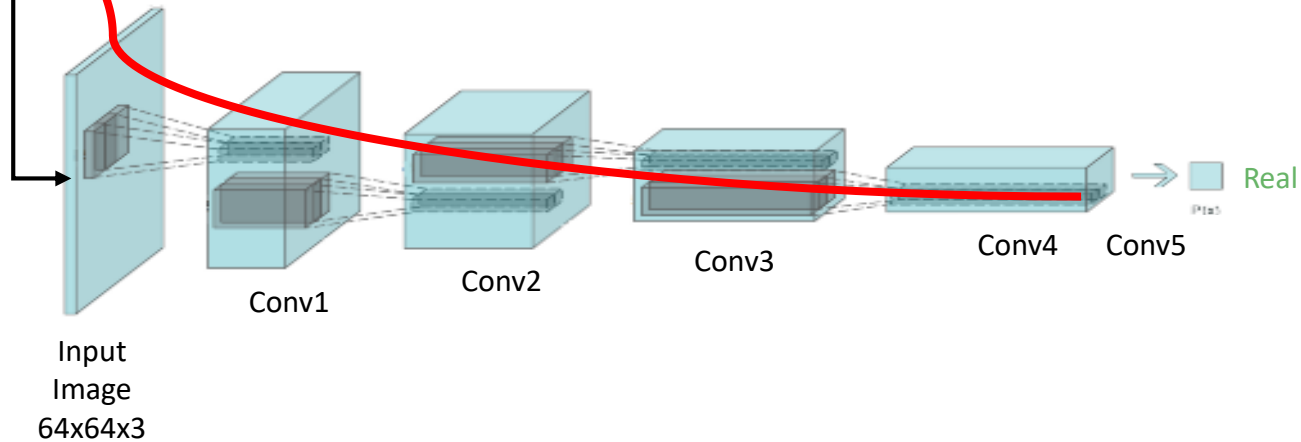
Generator **VS** Discriminator
Min-max Game

Training part 2: train generator

$G(z)$: batch_size, 64, 64, 3
Ground truth label: 1 (real)
**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

cGAN

StyleGAN

ProgressiveGAN

CycleGAN

InforGAN

LapGAN

BigBiGAN

AC-GAN

BiGAN

EBGAN

StackGAN

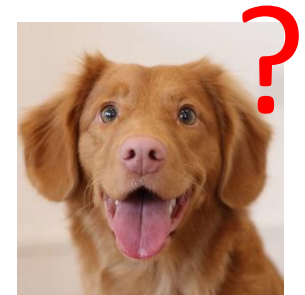
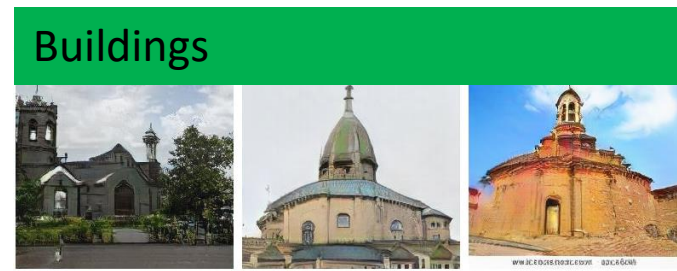
WGAN

BigGAN

BEGAN

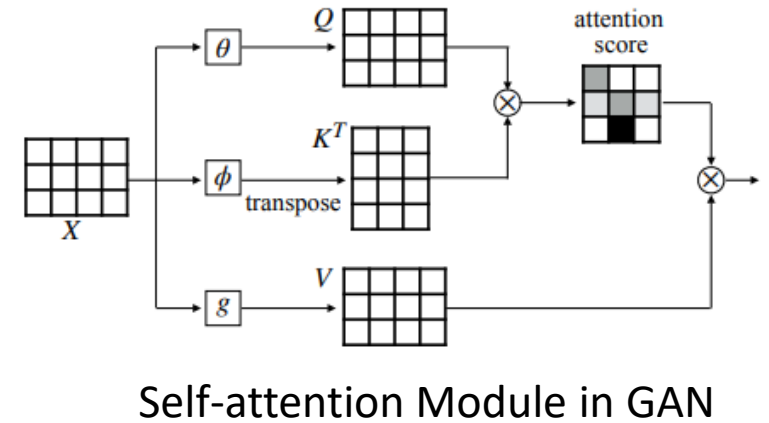
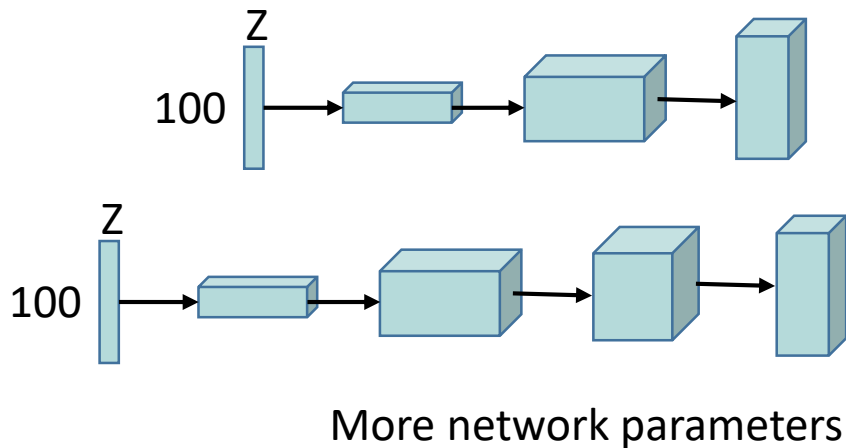
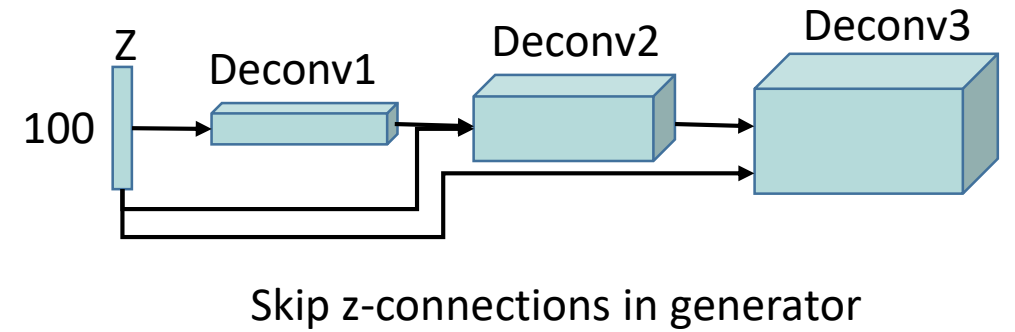
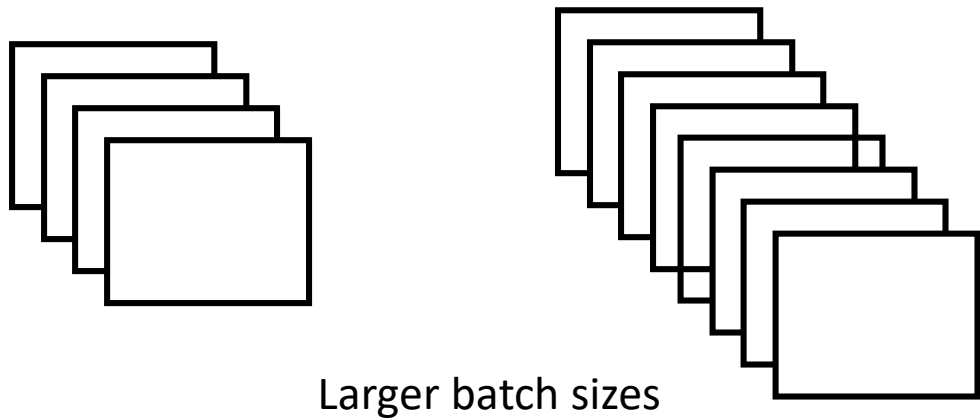
Problems with Deep Convolutional GAN (DCGAN)

- 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

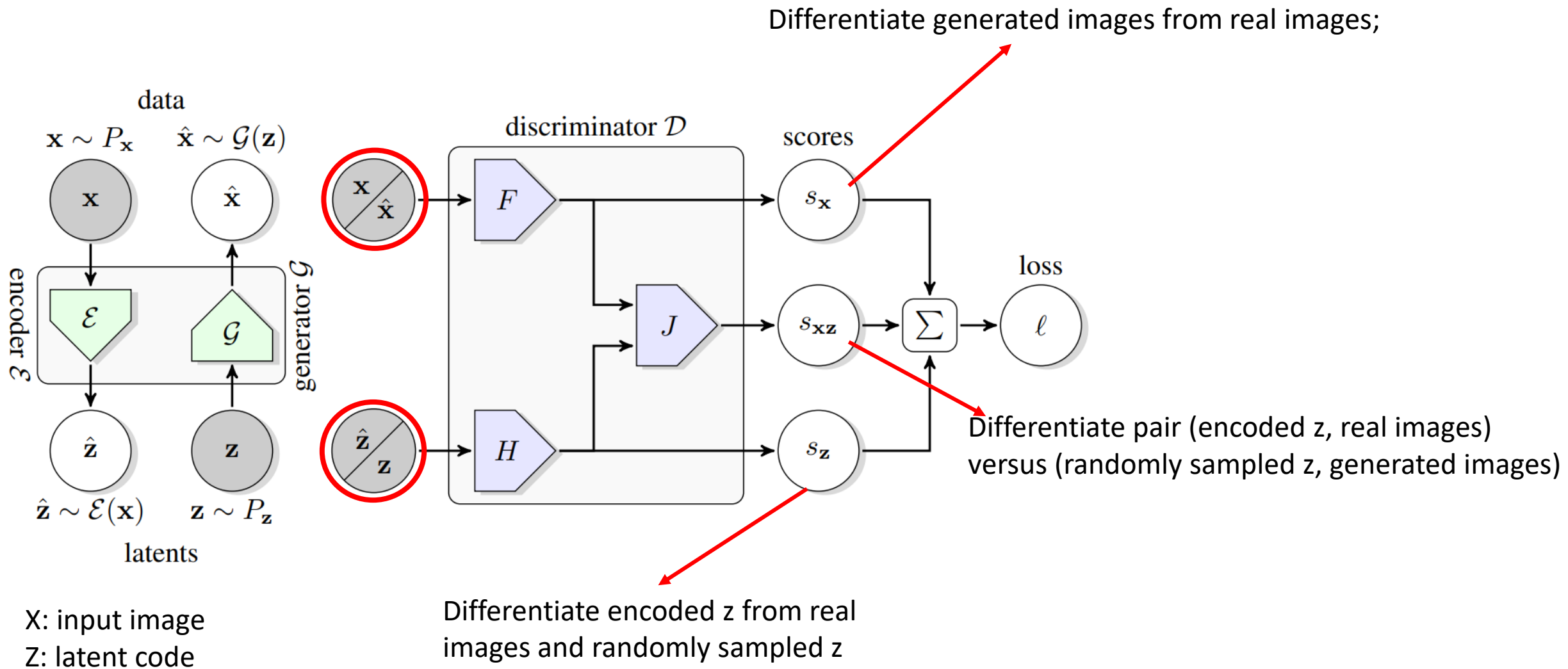


BigBiGAN

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



BigBiGAN

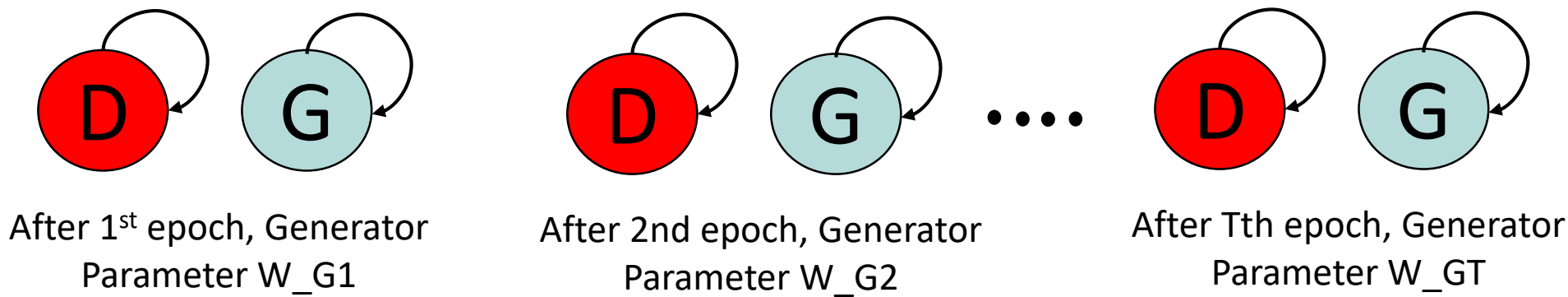


Tricks for More Realistic Image Reconstruction

- Update discriminators more often than generators during training



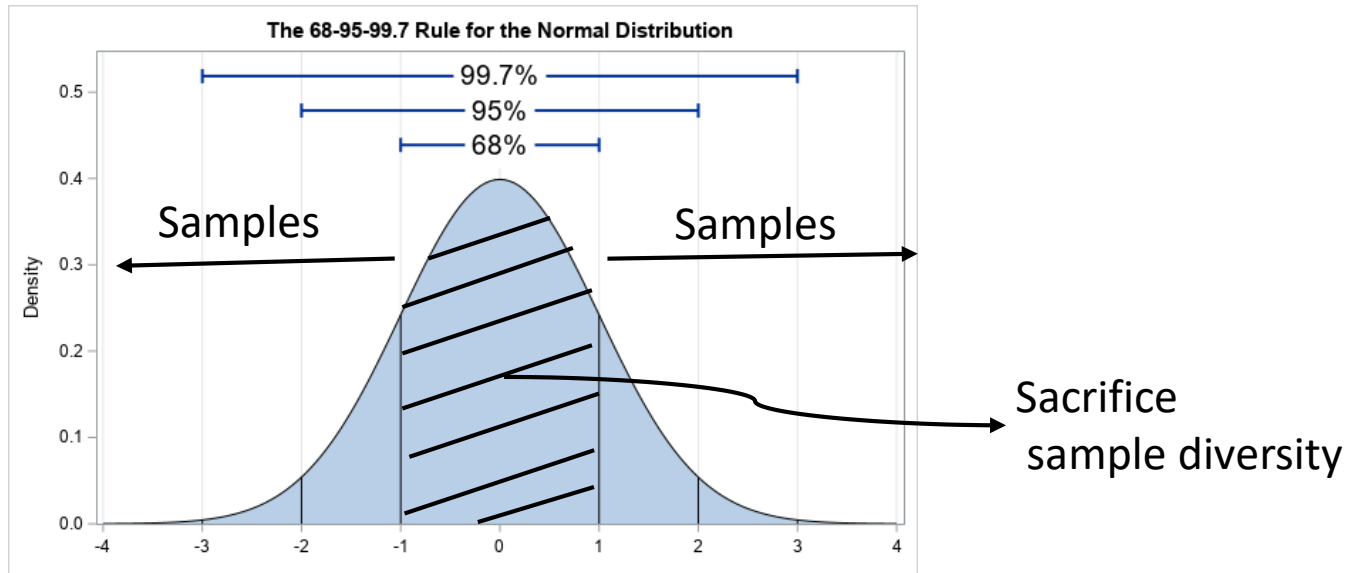
- Moving average of model weights (Progressive GAN)



$$W_{Gfinal} = \text{average}(W_{G1}, W_{G2}, \dots, W_{GT})$$

Tricks for More Realistic Image Reconstruction

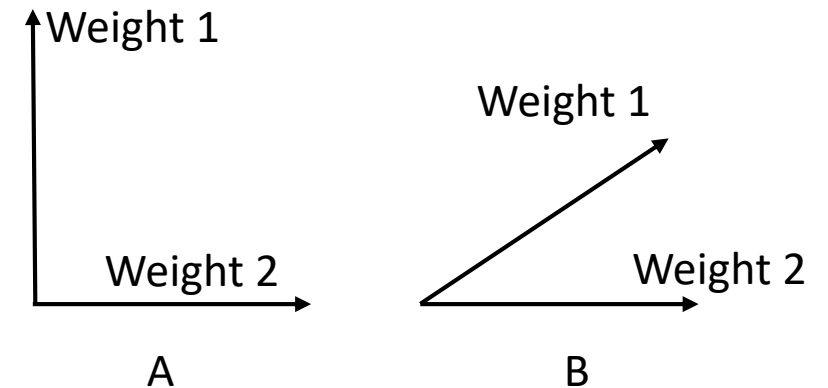
- Truncate z resampling at test stage



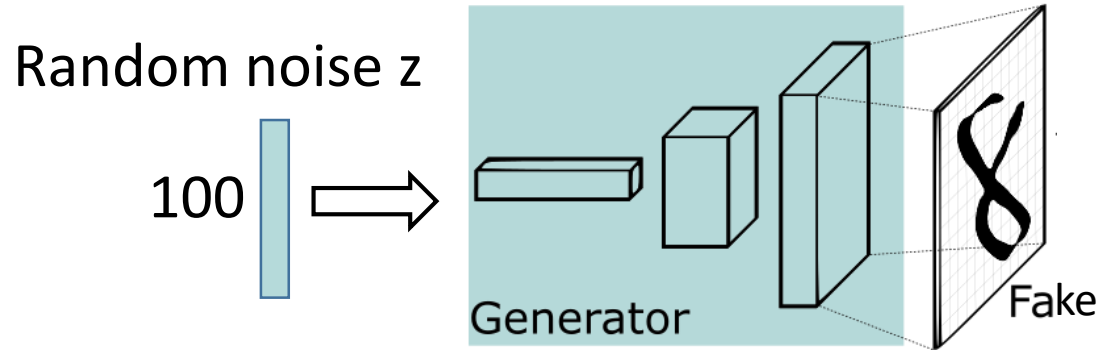
100 Z

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

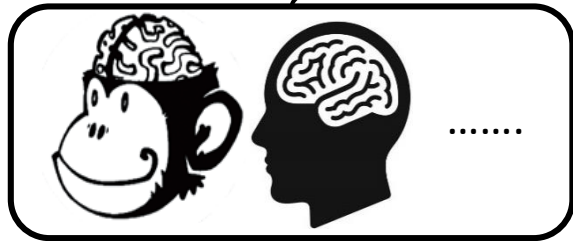
- Orthogonal weight initialization
- Orthogonal weight regularization



Brain Reading



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



All Animal species

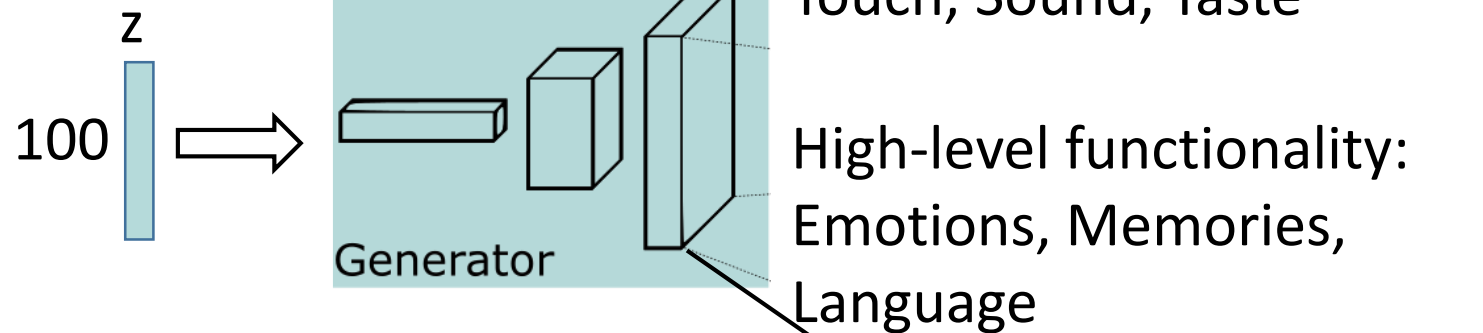
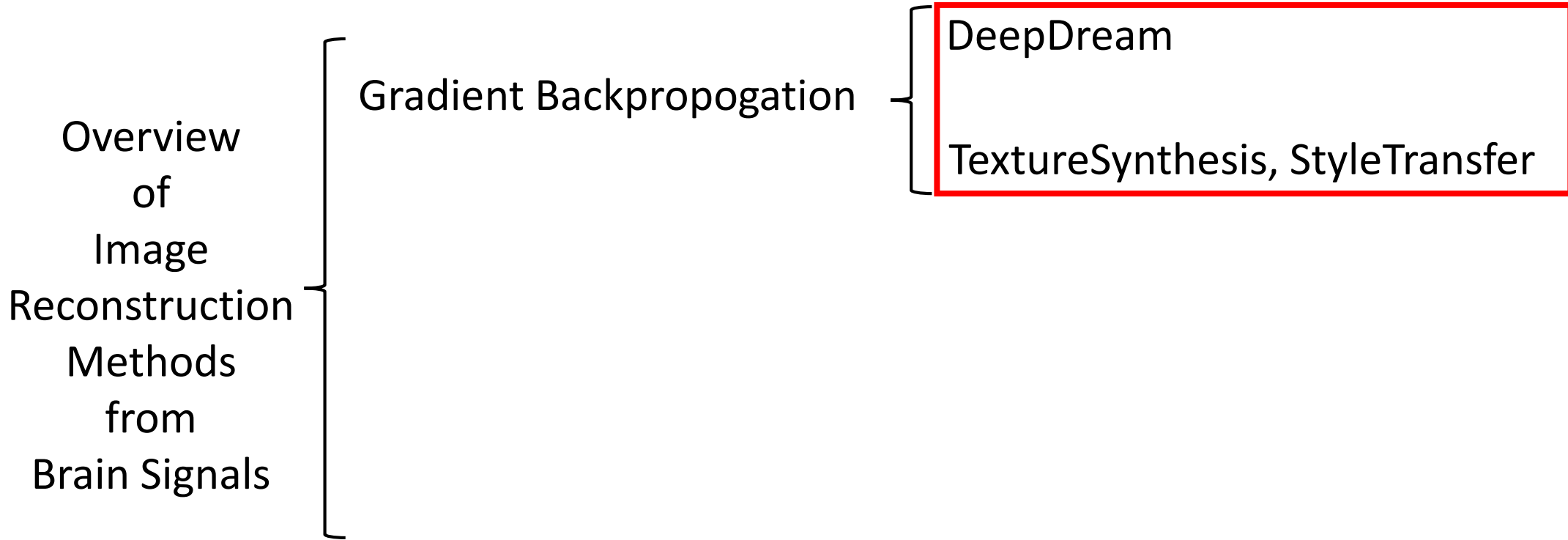
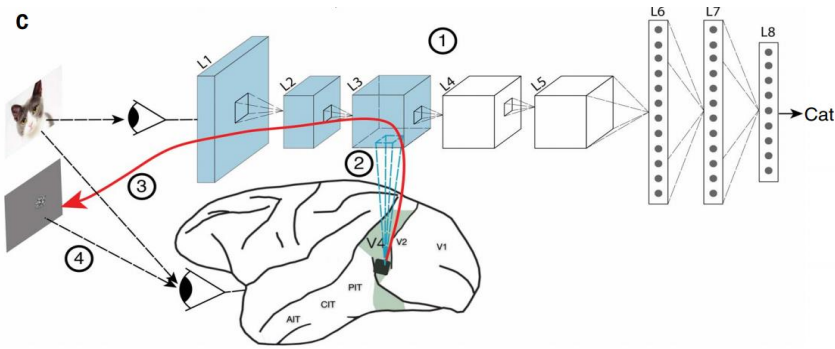


Image Reconstruction Methods from Brain Signals

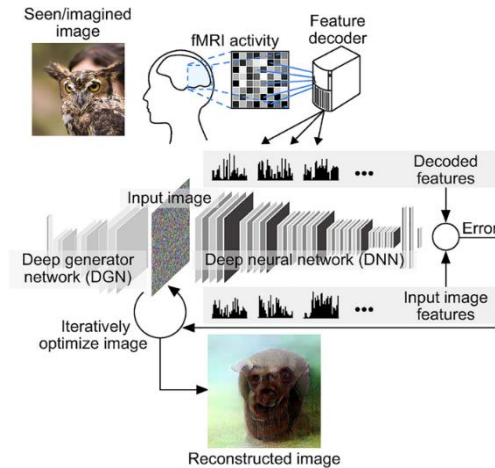


They are NOT generative models; but still cool

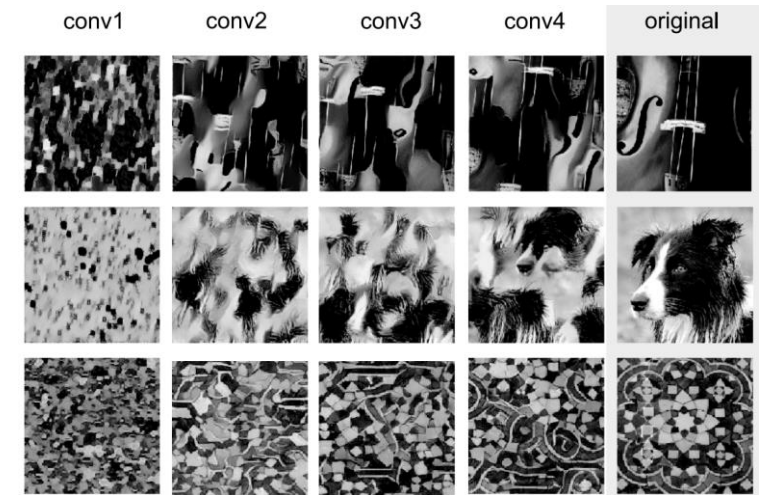
Gradient Backpropagation on pre-trained object recognition networks



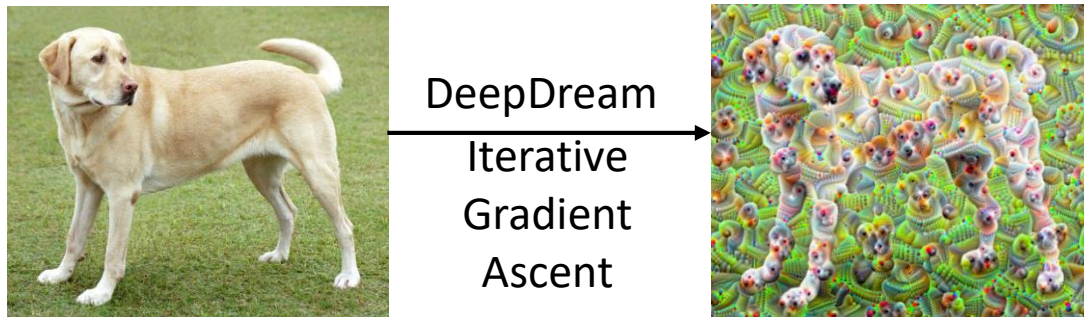
Bashivan et al, Science, 2019



Shen et al, Plos Computational Biology, 2019



Cadena et al, Plos Computational Biology, 2019



Feature MSE Loss at target layer

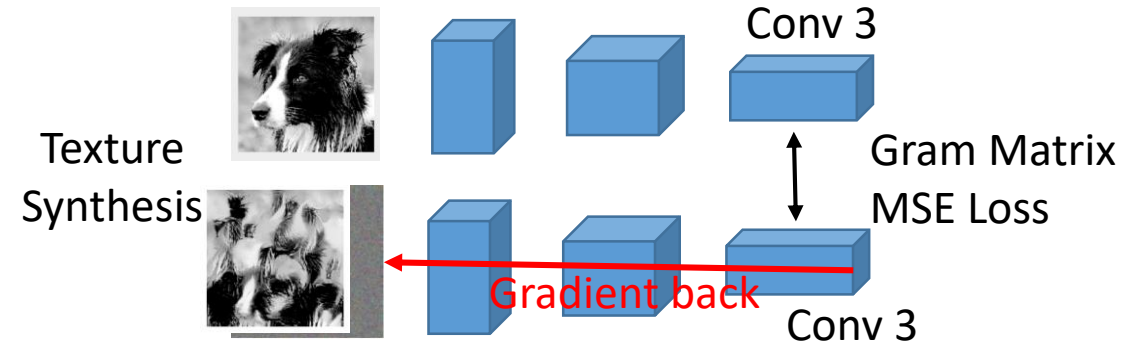
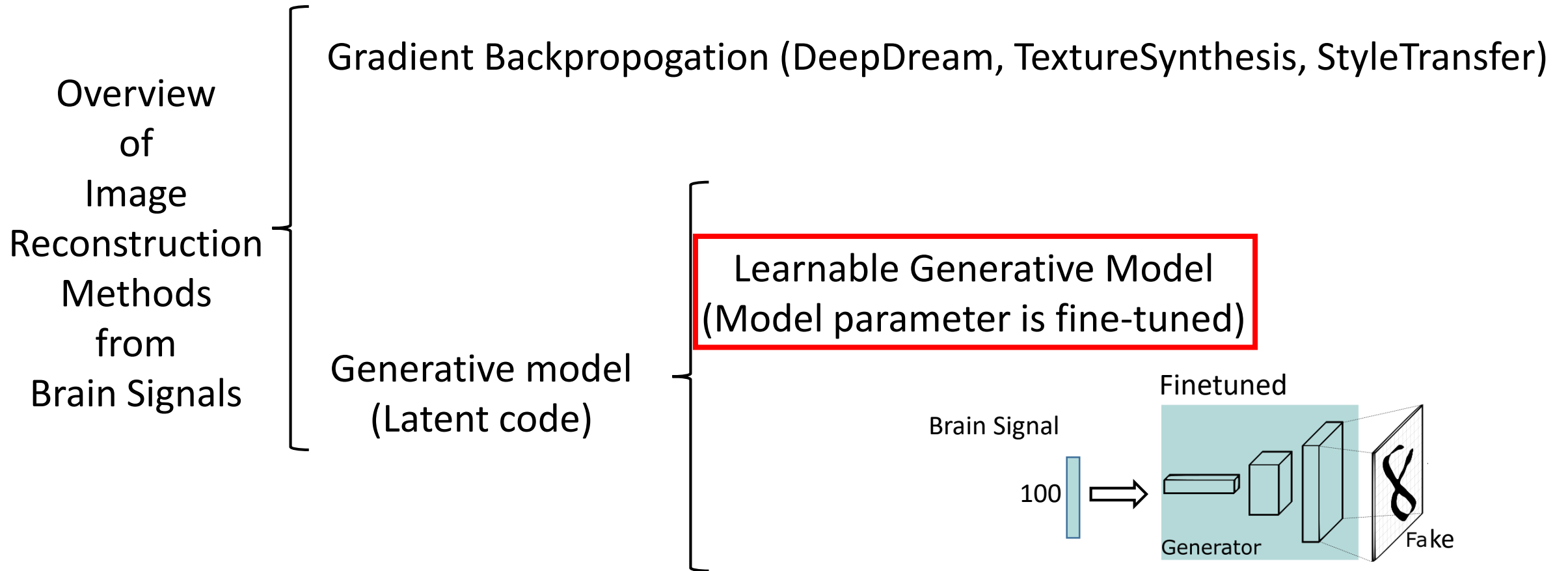
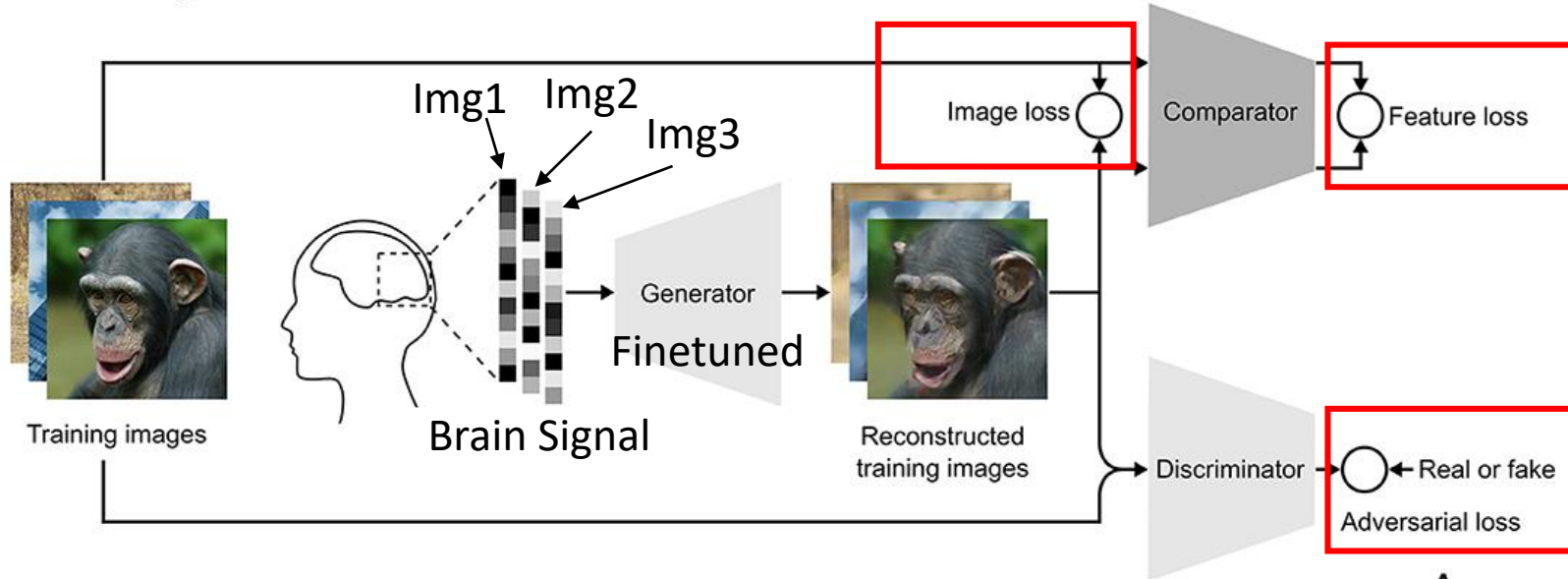


Image Reconstruction Methods from Brain Signals

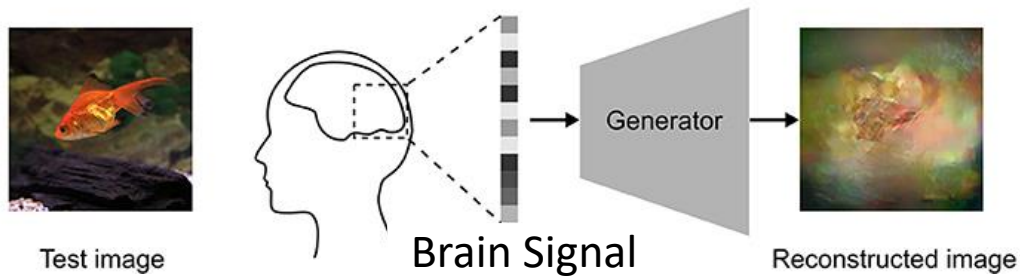


End-to-End Learnable Generative Model

A Model training



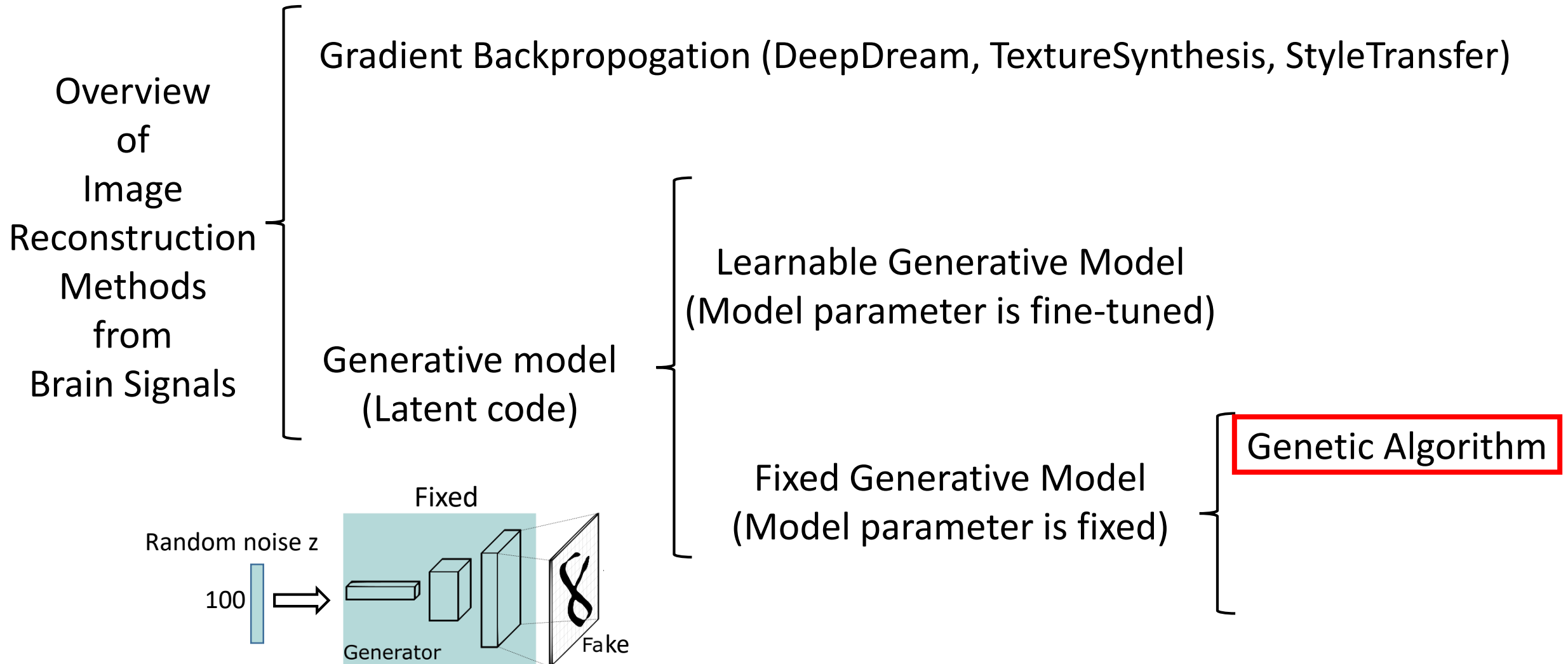
B Model test



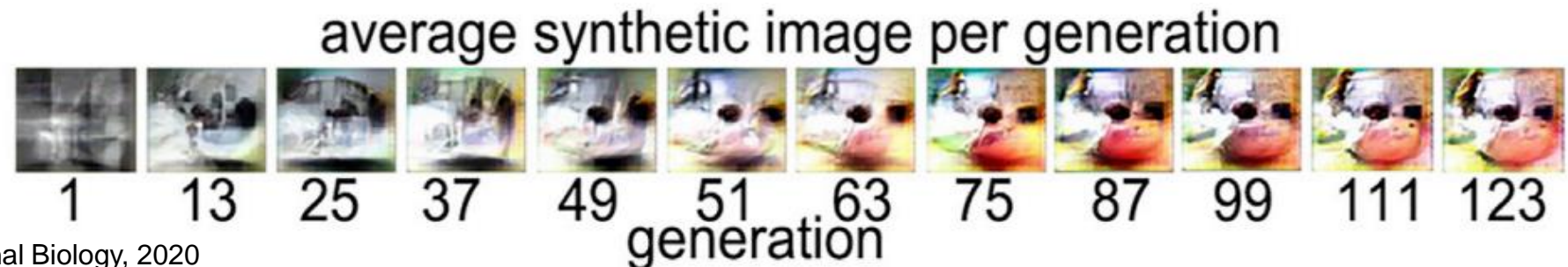
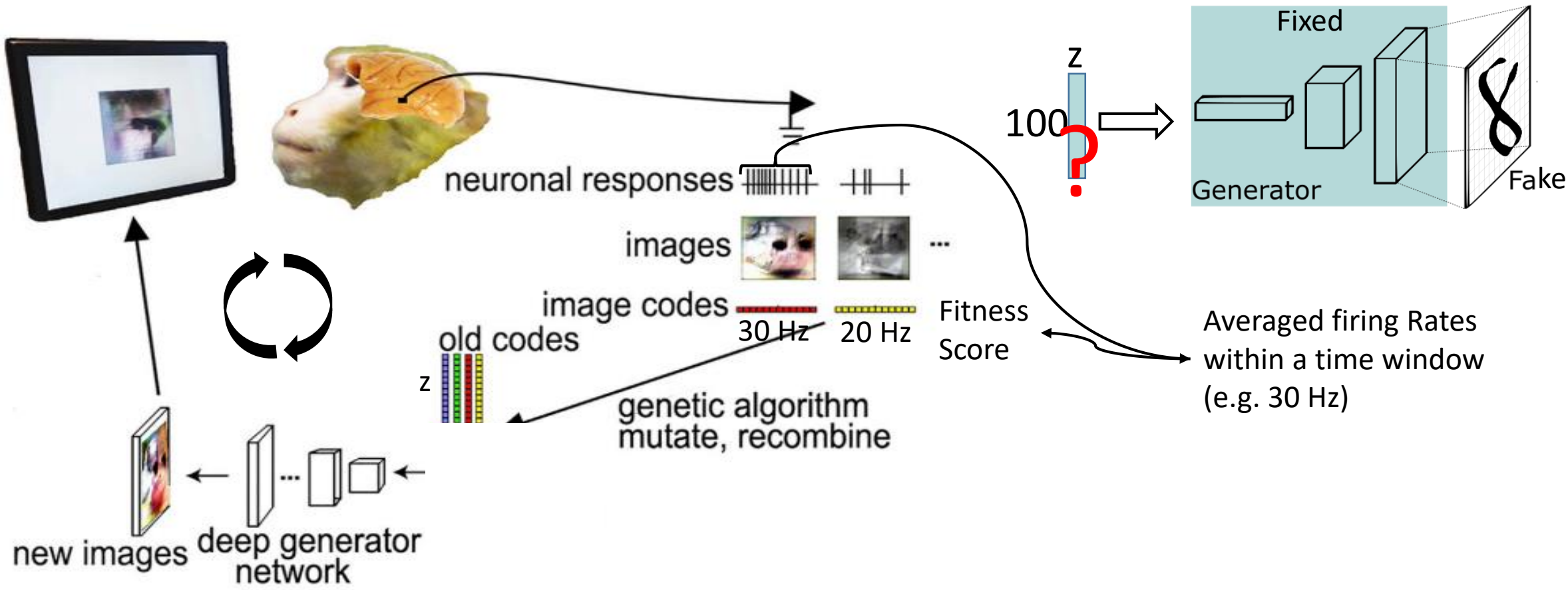
A



Image Reconstruction Methods from Brain Signals

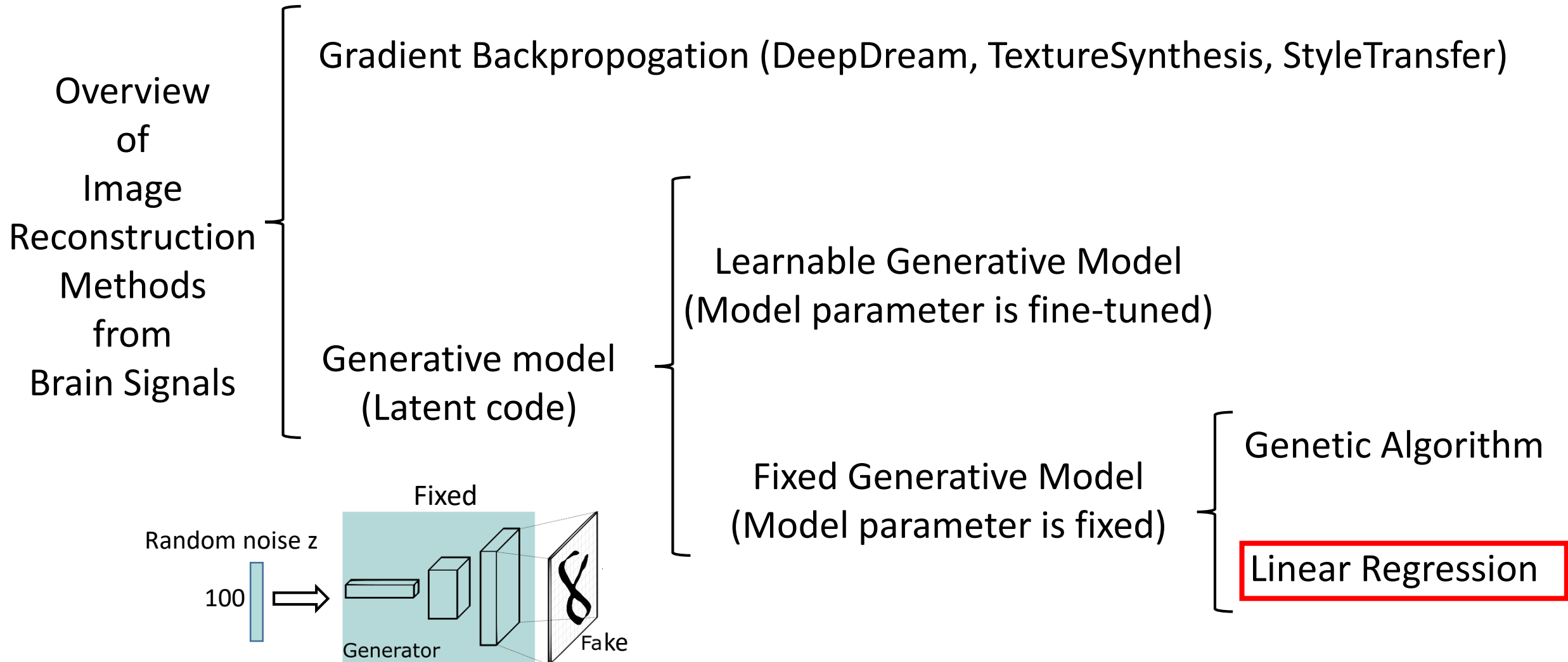


Evolving Latent Code using Genetic Algorithm

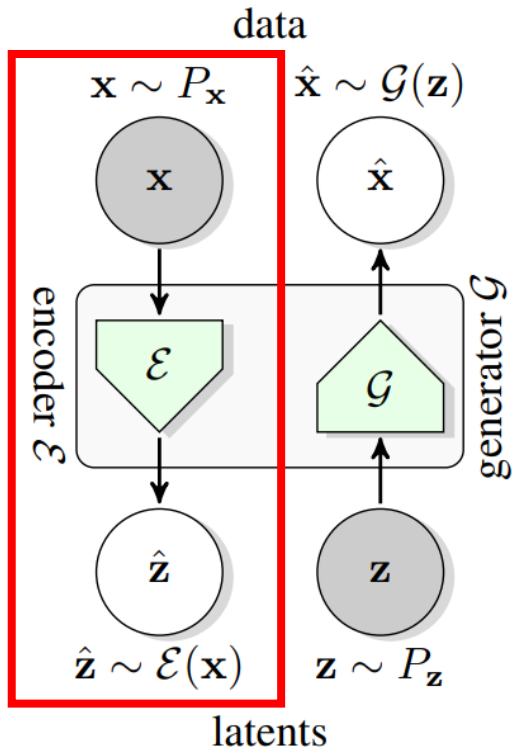


Ponce et al, Cell, 2019; Xiao et al, Plos Computational Biology, 2020

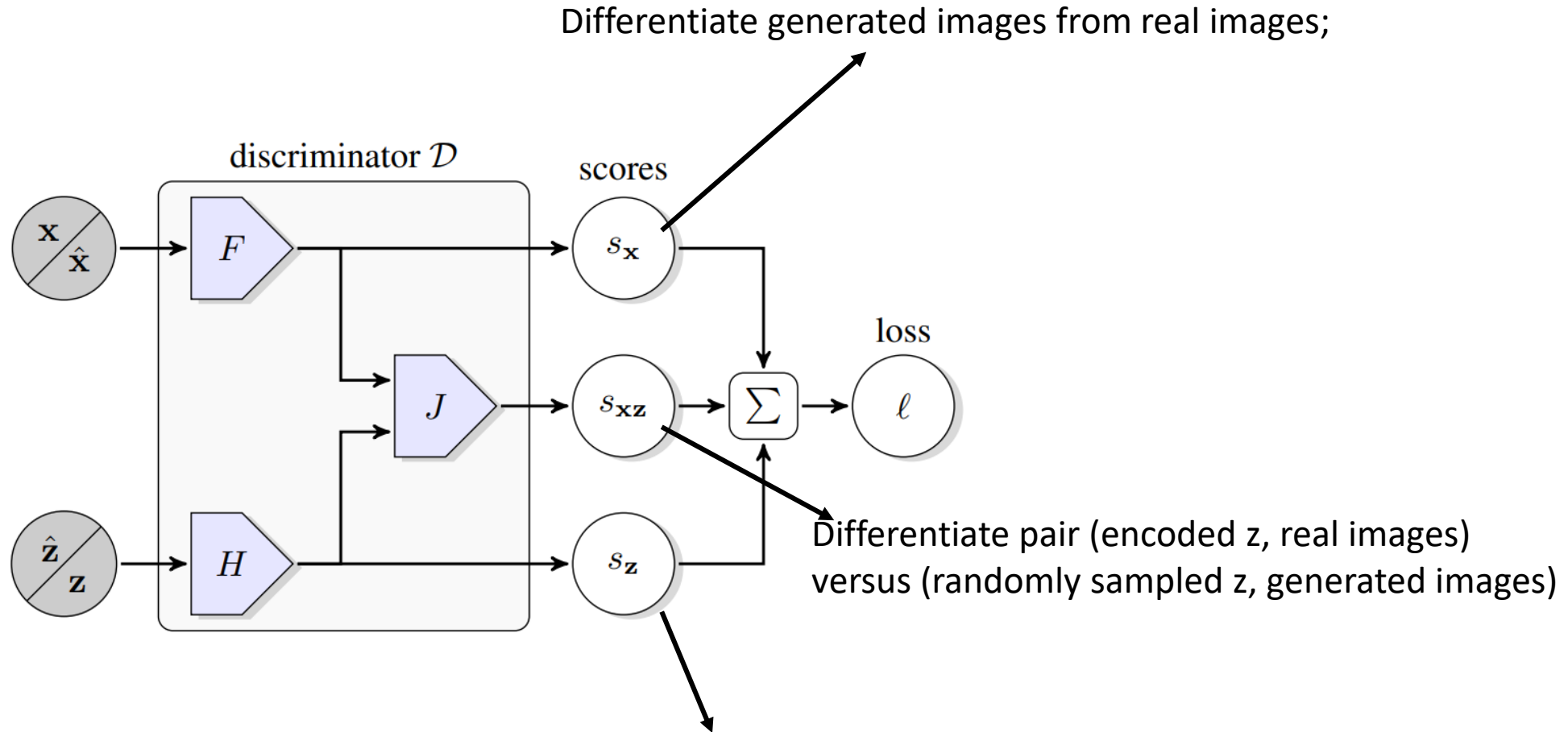
Image Reconstruction Methods from Brain Signals



BigBiGAN

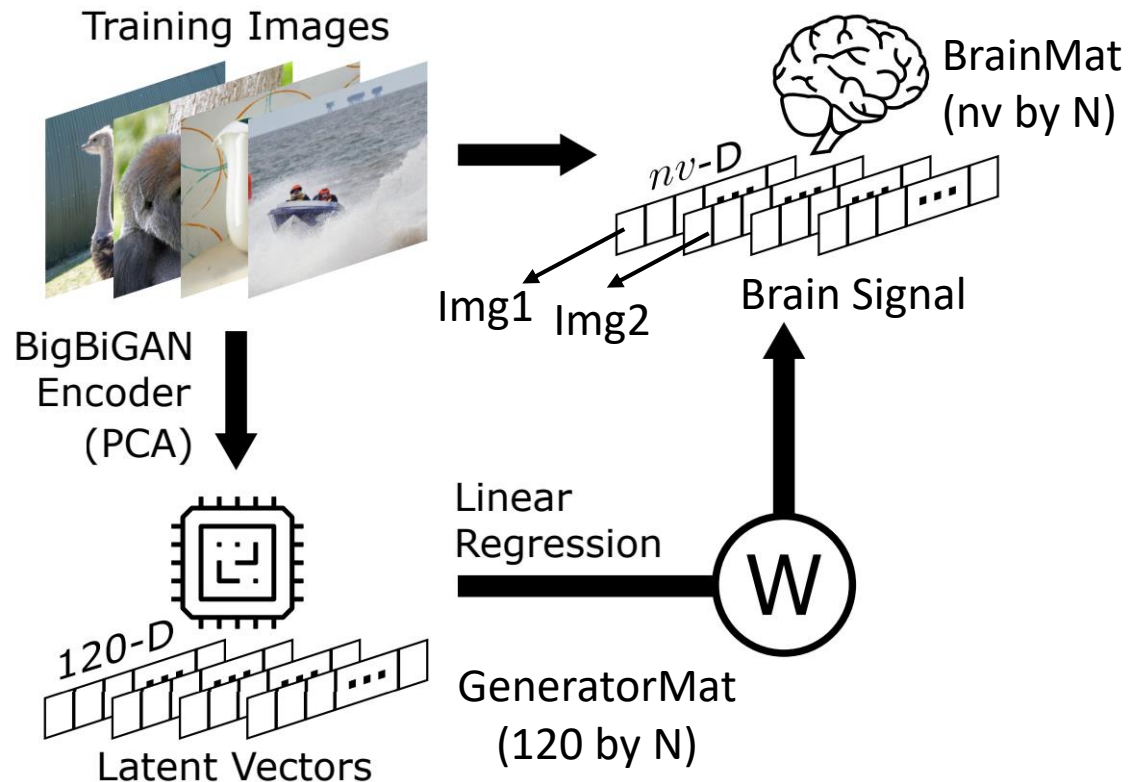


X: input image
Z: latent code



Mapping Latent Code using Linear Regression

N total training images



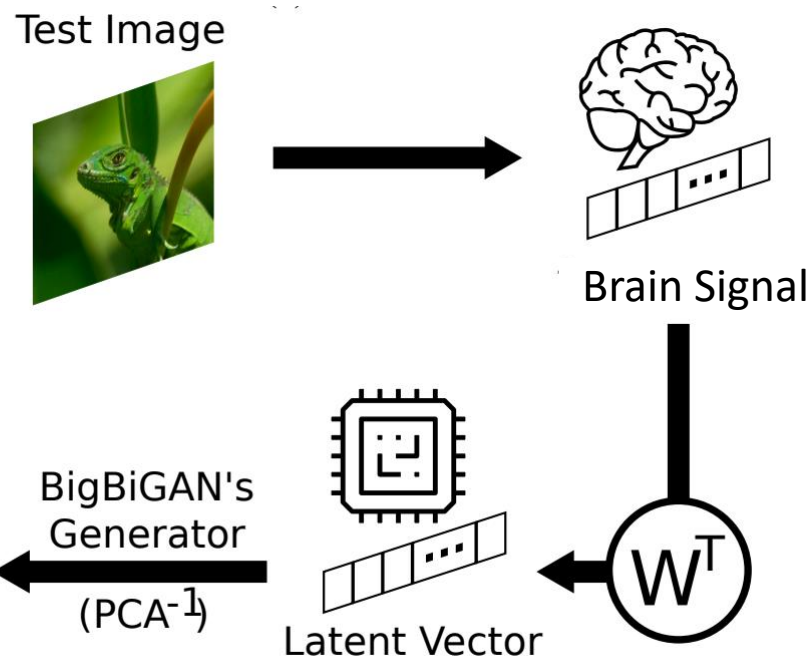
Training stage

$$\begin{matrix} \mathbf{W} \\ (nv \text{ by } 120) \end{matrix} \begin{matrix} \text{GeneratorMat} \\ (120 \text{ by } N) \end{matrix} = \begin{matrix} \text{BrainMat} \\ (nv \text{ by } N) \end{matrix}$$

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

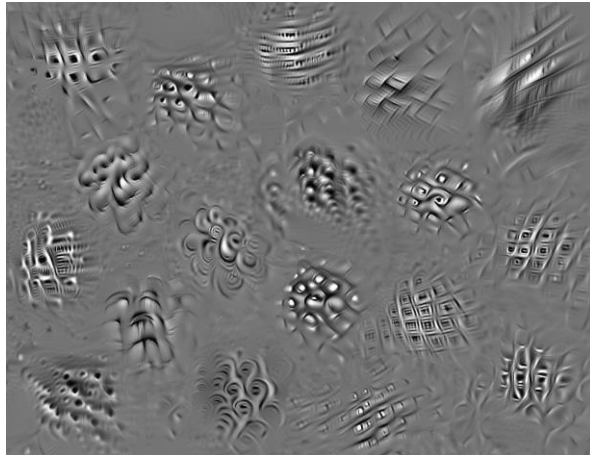
Mapping using Linear Regression



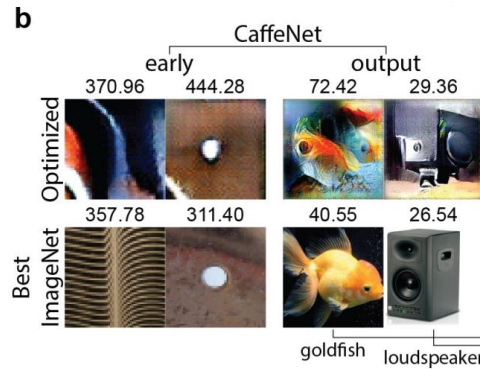
Testing stage

		fMRI Reconstruction					
#	Input Image	BigBiGAN Recon.	sub-01	sub-02	sub-03	sub-04	sub-05
1							
2							
3							
4							
5							

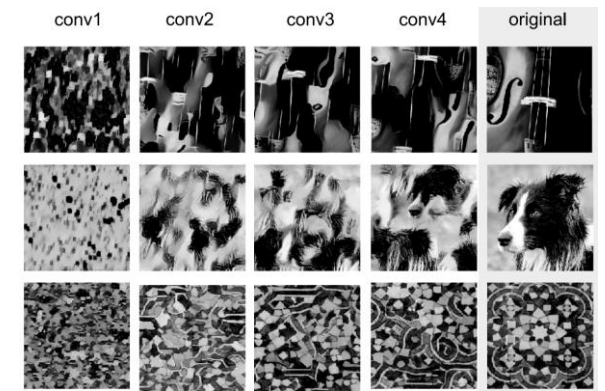
Which brain reader is better?



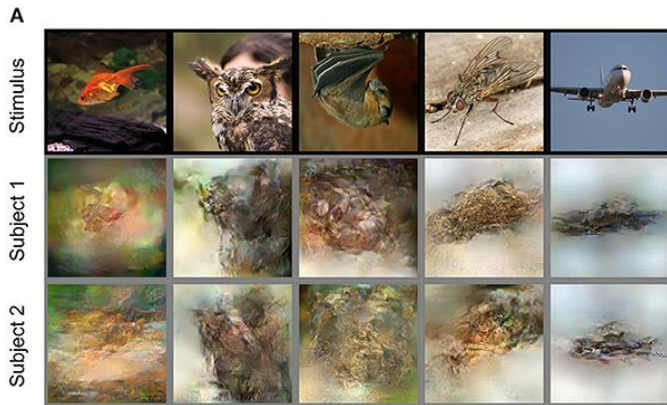
Bashivan et al, Science, 2019



Ponce et al, Cell, 2019;
Xiao et al, Plos Computational Biology, 2020



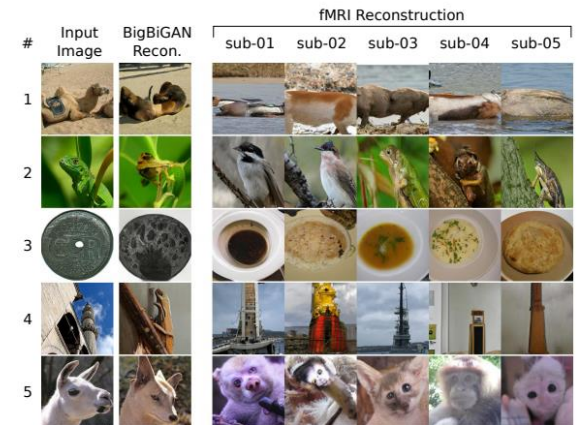
Cadena et al, Plos Computational Biology, 2019



Shen et al, Frontiers in Computational Neuroscience, 2019



Shen et al, Plos Computational Biology, 2019



Mozafari et al, arxiv, 2020

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

More brain scores on generative models



Real Reconstructed

Methods	L2 norm, Cosine SIM, etc At pixel level	Inception Score (IS)	Frechet Inception Distance (FID)	ImageNet Classification Accuracy	Human Behavioral Score	Brain Signal Score (Linear Correlation)
Bashivan etal, Science, 2019						
Shen etal, Frontiers in Computational Neuroscience, 2019						
Mozafari etal, arxiv, 2020						
Ponce etal, Cell, 2019; Xiao etal, Plos Computational Biology, 2020						
Shen etal, Plos Computational Biology, 2019						
Cadena etal, Plos Computational Biology, 2019						
Many more to come ...						

More brain scores - IS

* Inception Score (measuring image quality + diversity)

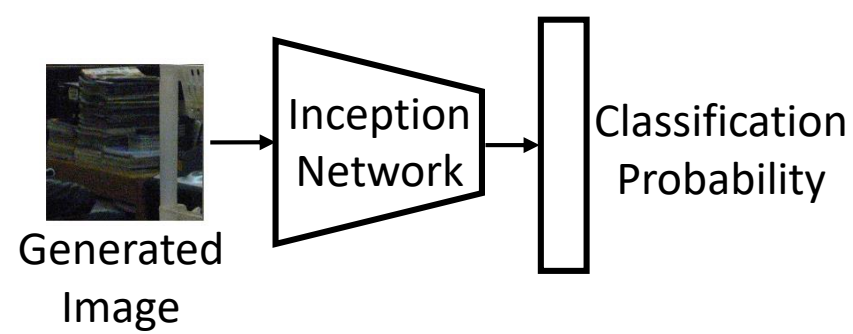


Image Quality (more focused label distribution -> better; lower entropy)

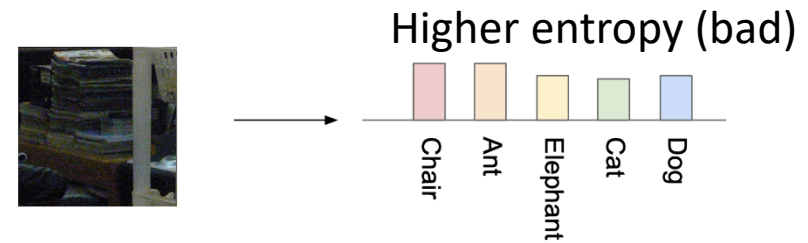
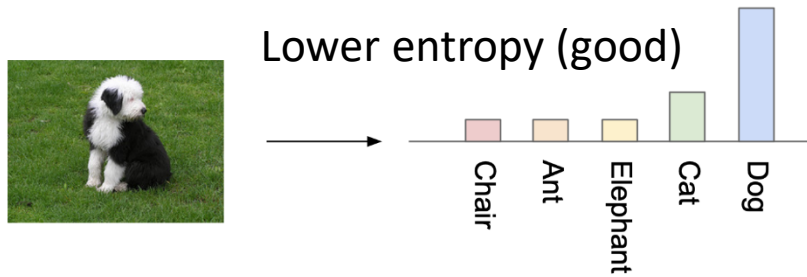
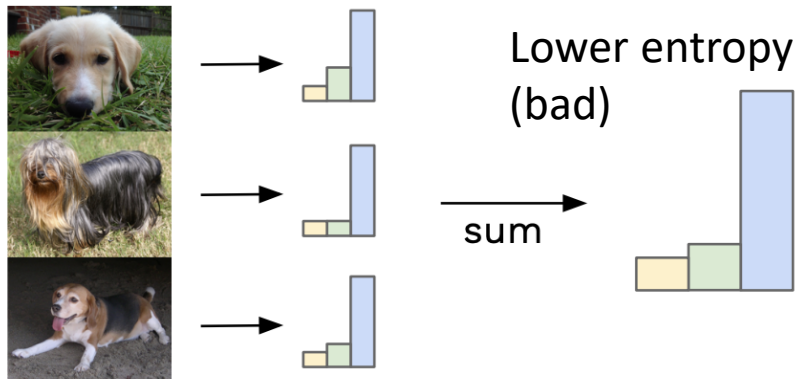
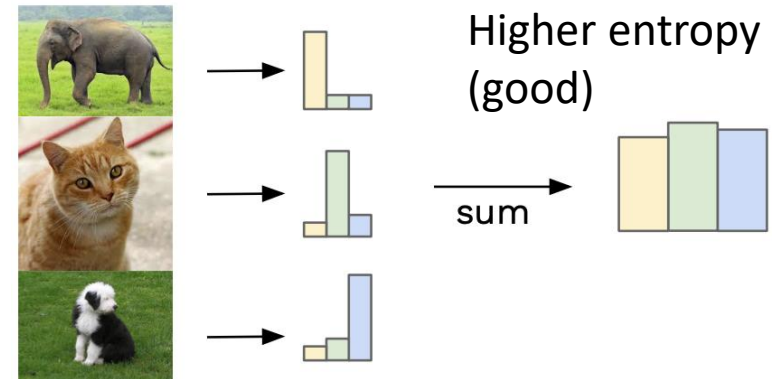


Image Diversity (more uniform marginal distribution -> better; higher entropy)

Similar labels sum to give focussed distribution

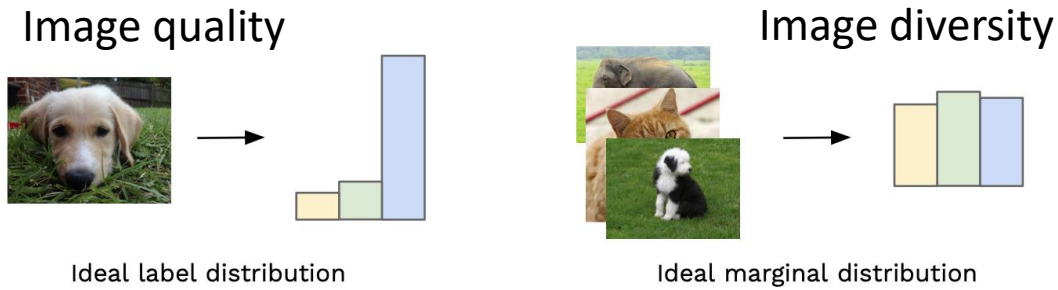


Different labels sum to give uniform distribution



More brain scores - FID

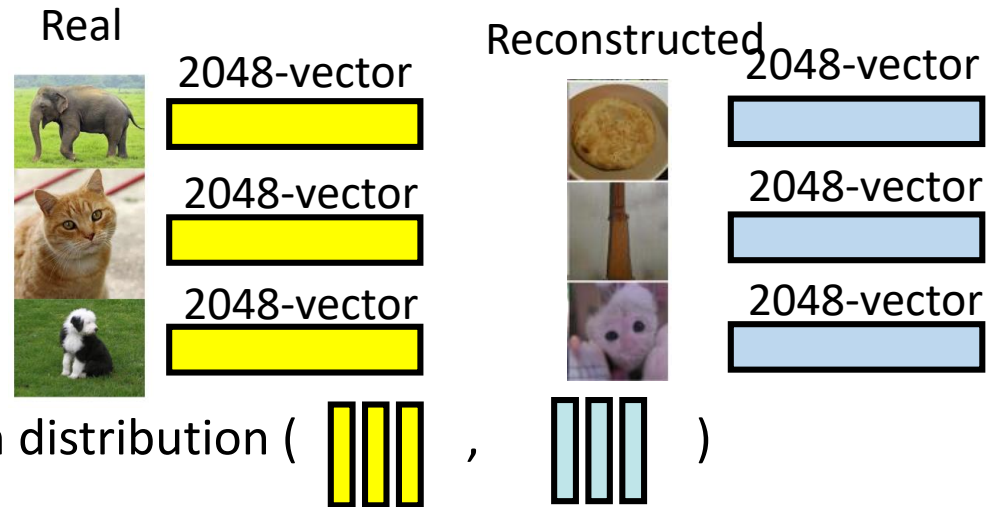
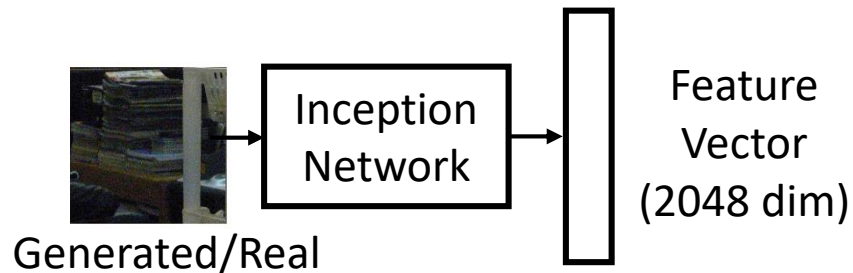
* Inception Score (measuring image quality + diversity)



Inception Score (IS) = KL Divergence ( , )

The higher IS score, the better.

* Fréchet Inception Distance (FID)



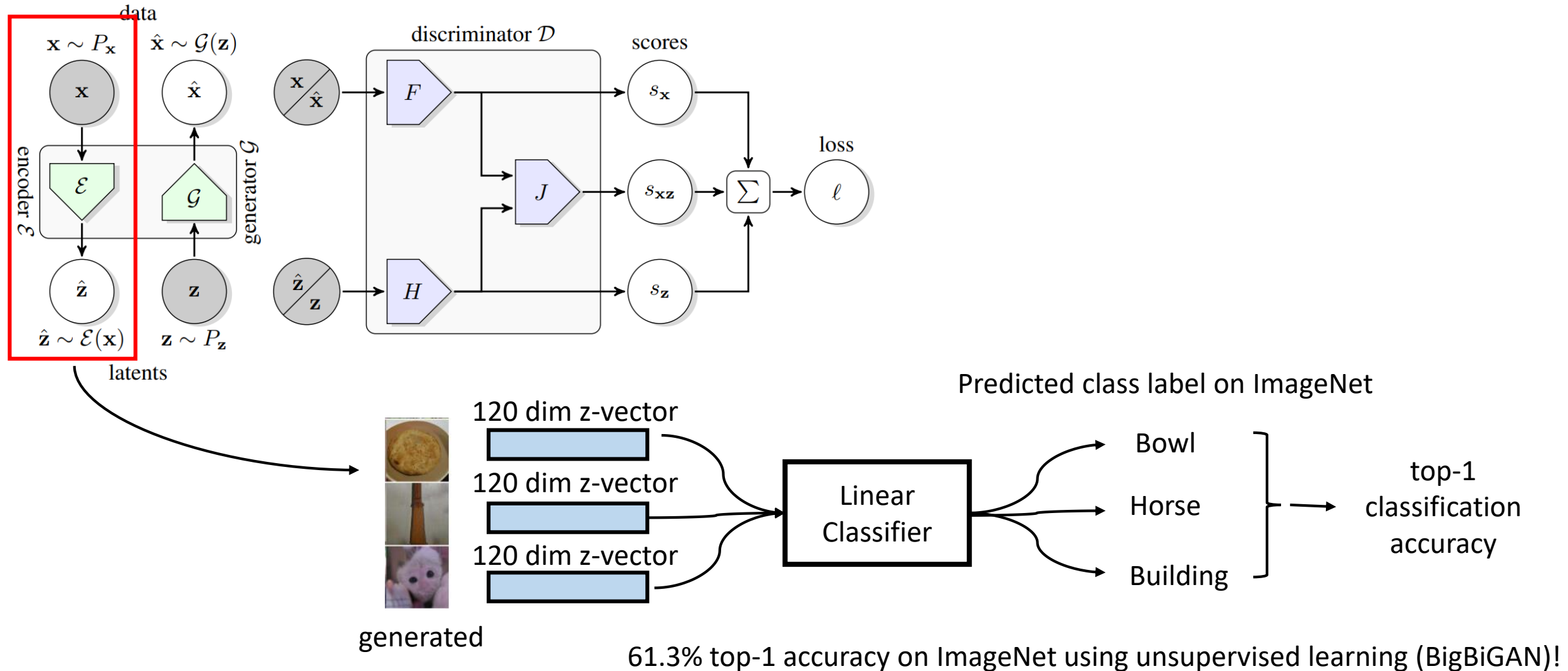
FID= Distance between two multi-variant Gaussian distribution ( , )

The lower FID score, the better.

$$FID = ||\mu_r - \mu_g||^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}),$$

More brain scores – Classification accuracy

* ImageNet top-1 classification accuracy



More brain scores

*Human Judgment at behavioral levels

Reconstructed from brain signal



Real Image as target

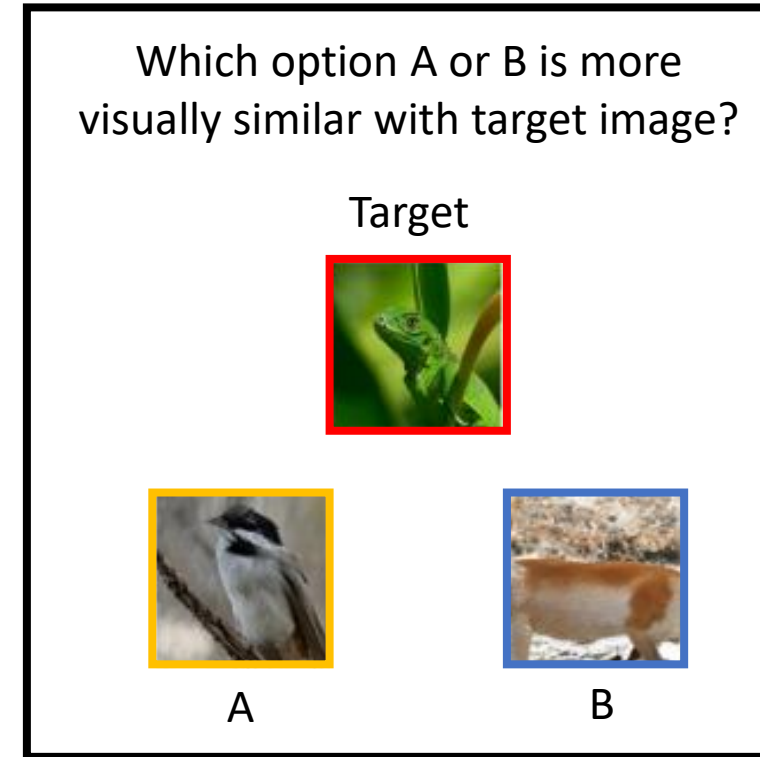


Randomly chosen reconstructed image based on brain activity from real image

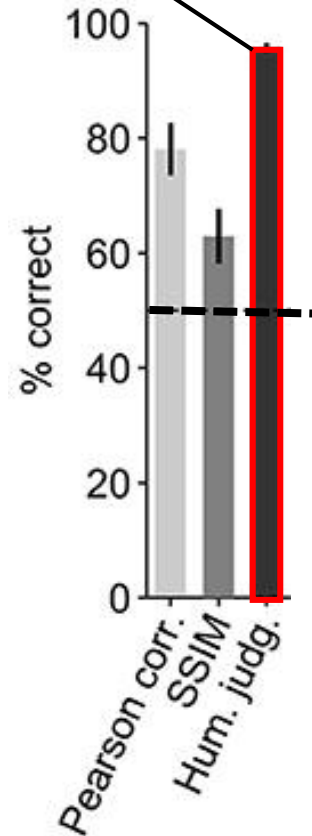


Randomly chosen reconstructed image from other irrelevant images

~90% people chose options which are reconstructed based on brain signals generated after seeing target image



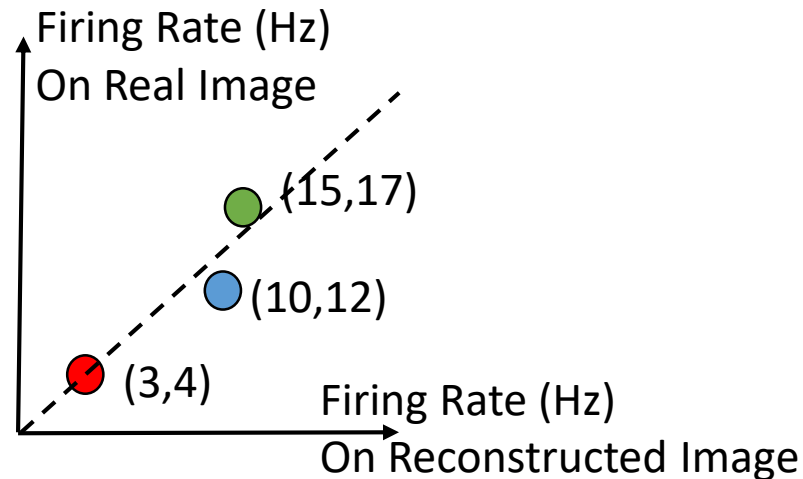
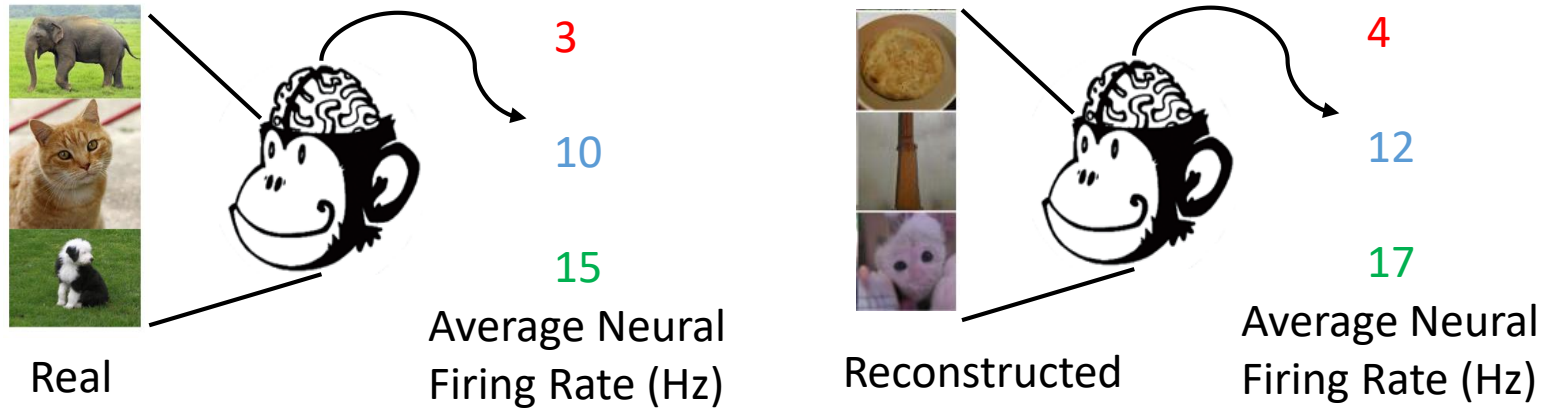
Example trial on Amazon Mechanical Turk



More brain scores

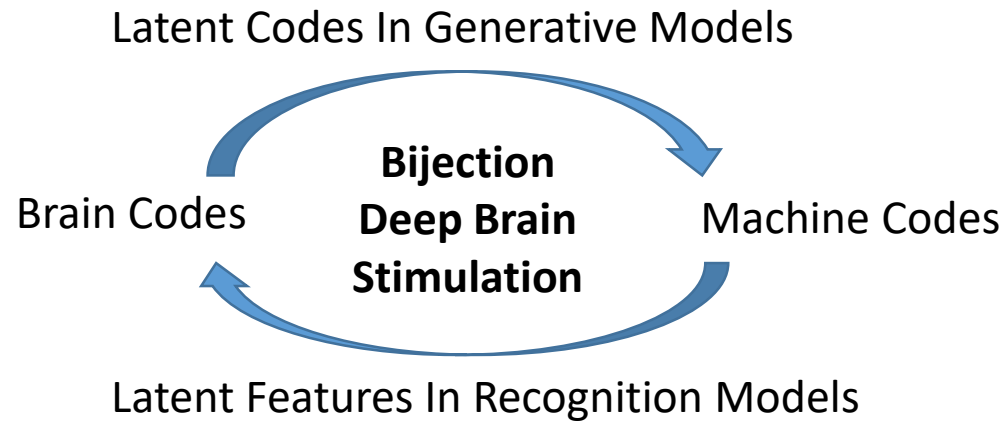
*Control ability of brain signals:

Linear correlation between predicted brain signals and real brain signal

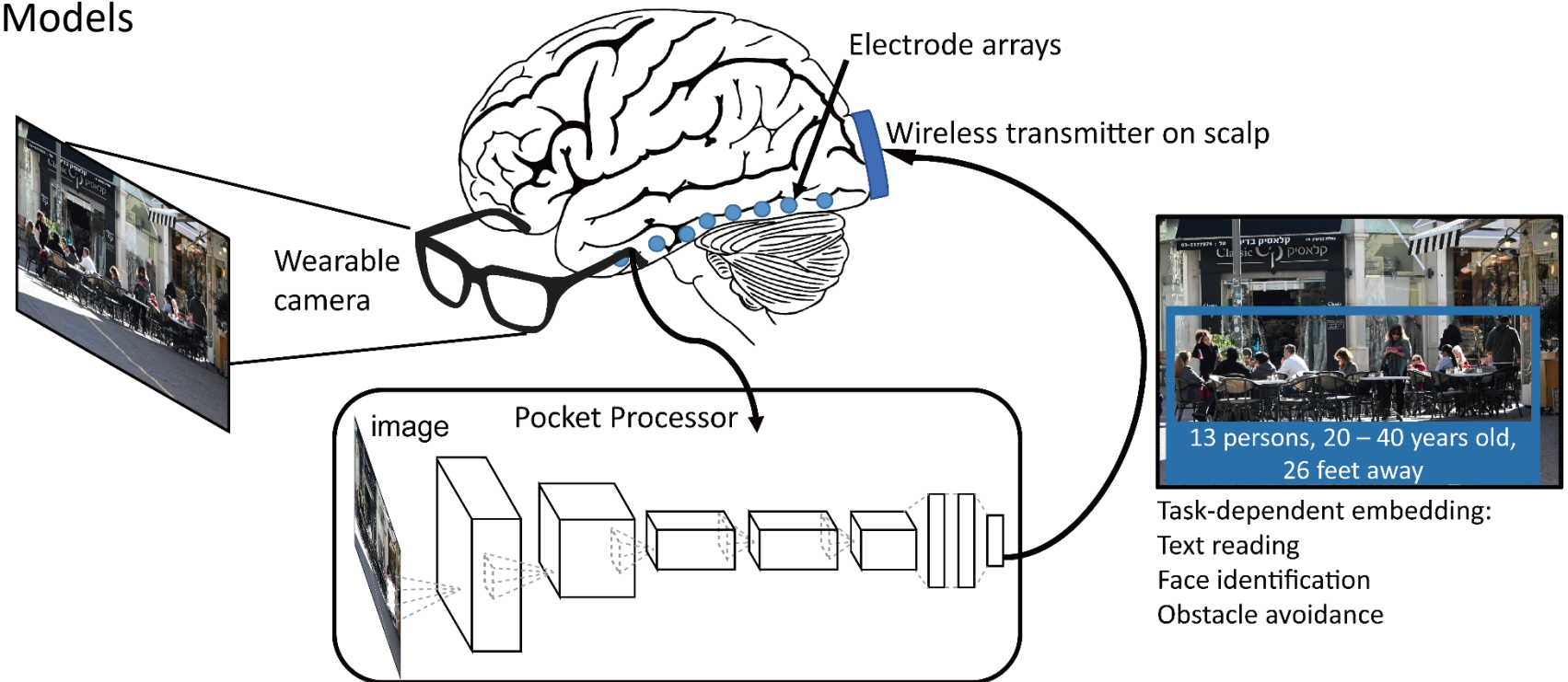


**Any other brain
signals ...**

Towards Brain Computer Interface

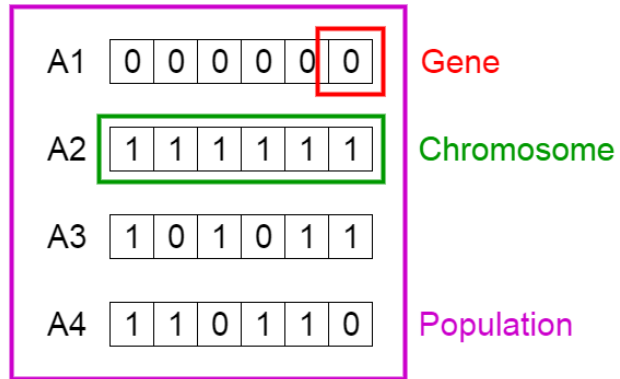


Brain Machine Interface (BCI) for the visually impaired



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

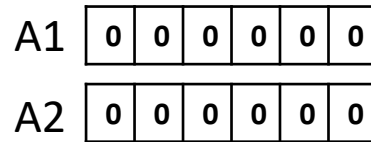


Population, Chromosomes and Genes

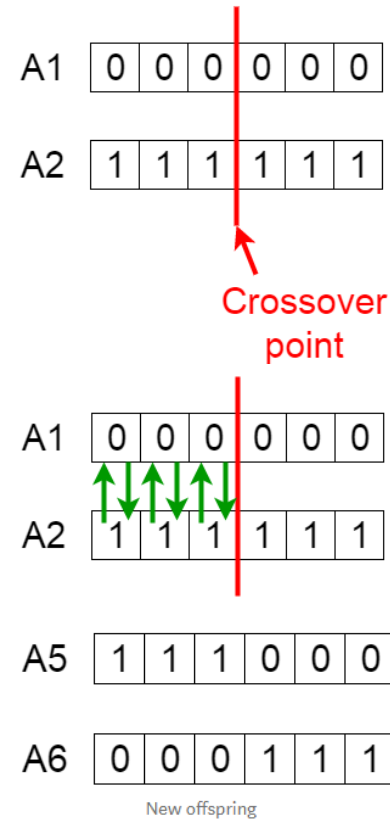
Fitness score
(Average Firing Rates)

A1	0	0	0	0	0	0	20
A2	1	1	1	1	1	1	15
A3	1	0	1	0	1	1	4
A4	1	1	0	1	1	0	2

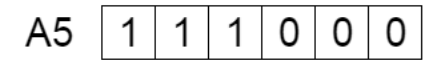
Selection



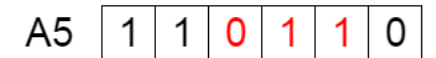
Fittest parents



Before Mutation



After Mutation



Mutation: Before and After

```

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

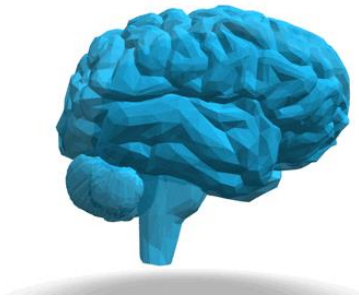
Preliminaries on fMRI data collection



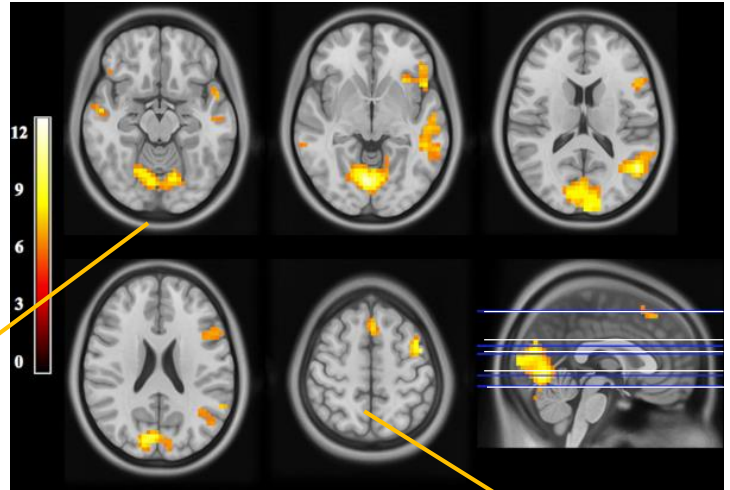
MRI machine

Image presentation

Controller



3D brain scans
slice by slice



5 slices



Number of voxels per slice

Num of voxels - dimension vector for fMRI scans

Image credit: https://www.researchgate.net/figure/fMRI-maps-group-analysis-Statistical-maps-axial-slices-oriented-in-radiological_fig1_299516048;
<http://aucanlab.com/fmri/>