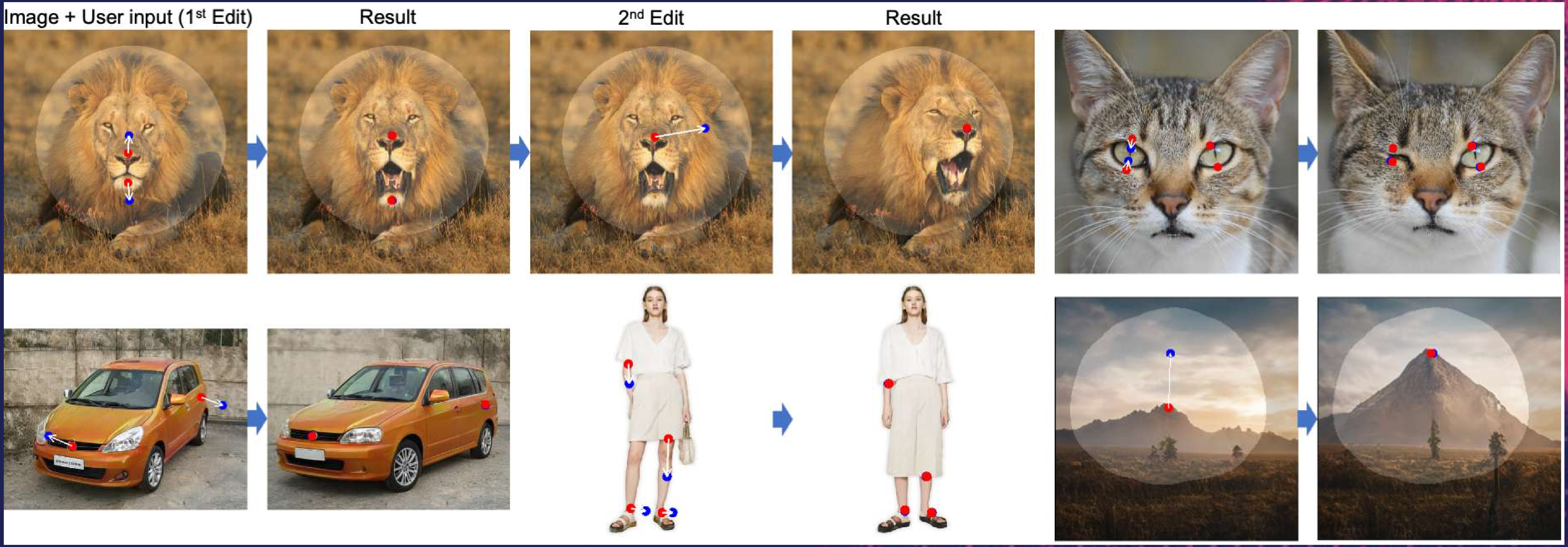
Epoch 2: Generative AI

A Gentle Introduction

CONTENTS

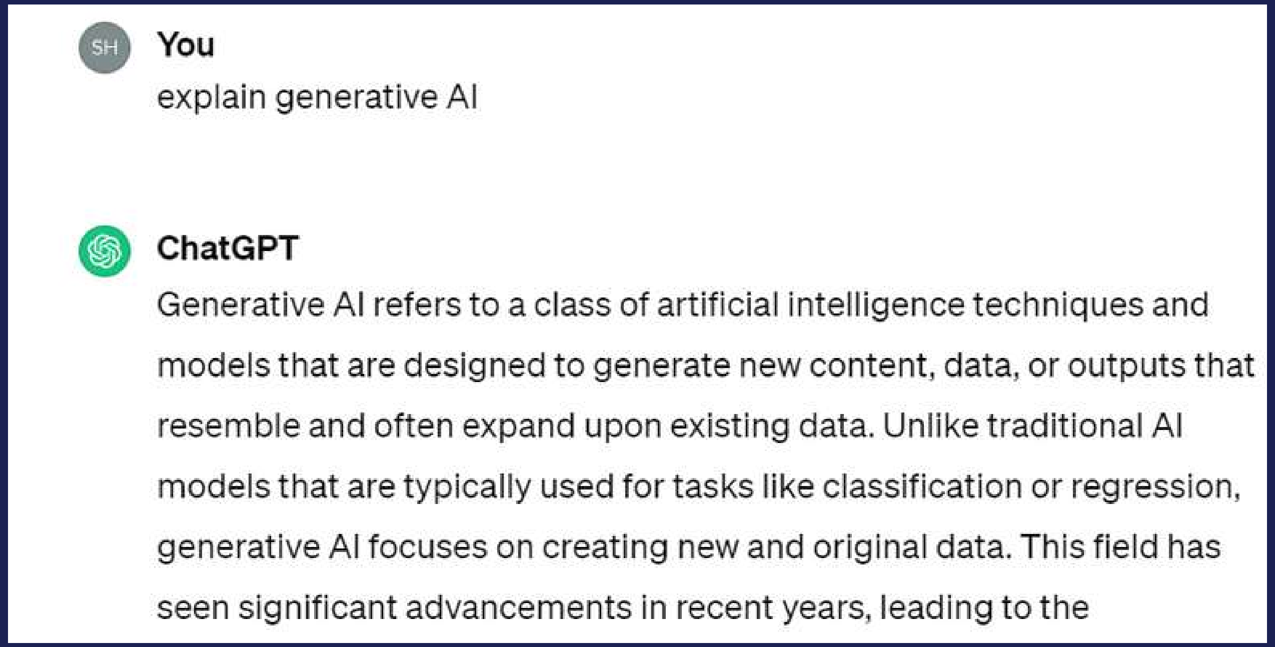
So what is Generative AI?

Generative AI refers to a category of artificial intelligence that creates new content/data that resemble or are similar to existing data.



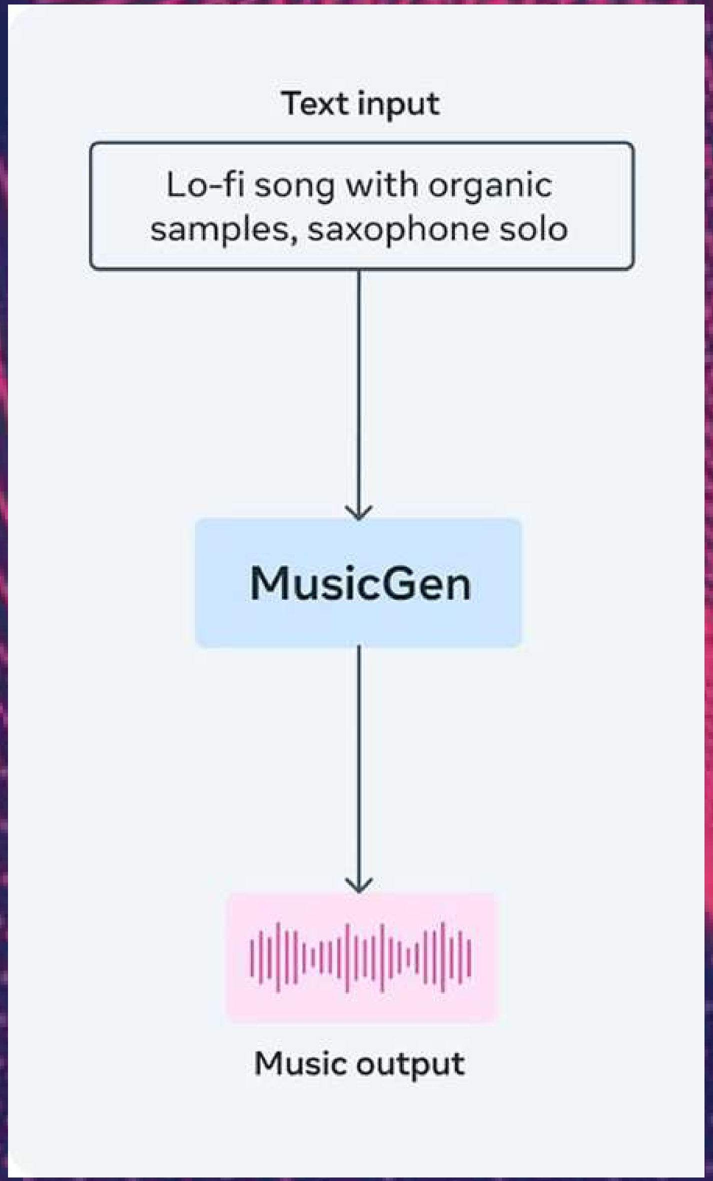
What are some applications of Generative AI?

Natural Language Processing


SH You
explain generative AI

ChatGPT
Generative AI refers to a class of artificial intelligence techniques and models that are designed to generate new content, data, or outputs that resemble and often expand upon existing data. Unlike traditional AI models that are typically used for tasks like classification or regression, generative AI focuses on creating new and original data. This field has seen significant advancements in recent years, leading to the

Music



Generative AI

Computer Vision

How is Generative AI used in images?

Image Synthesis

A head of broccoli
complaining
about the weather

Style Transfer

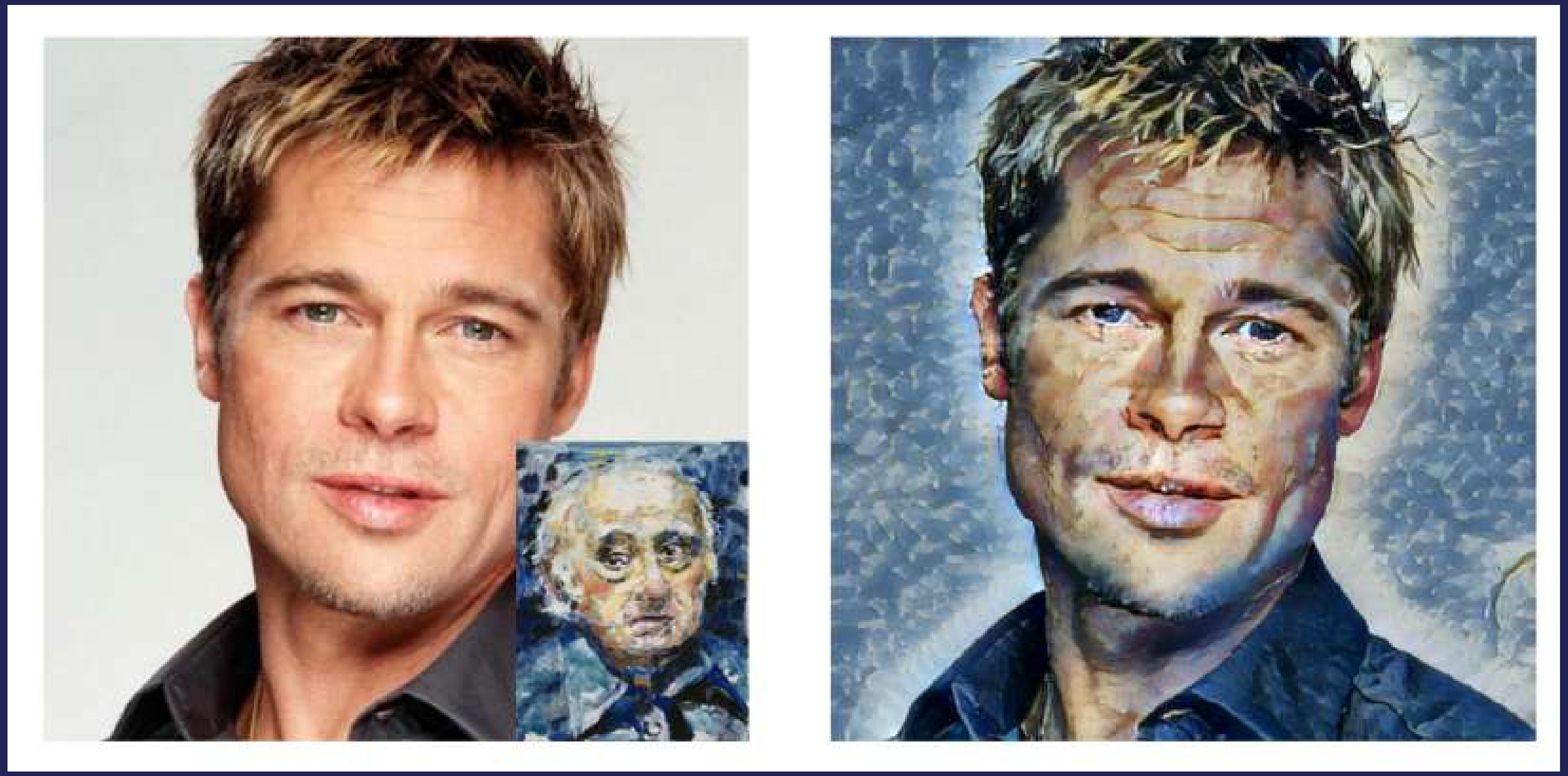
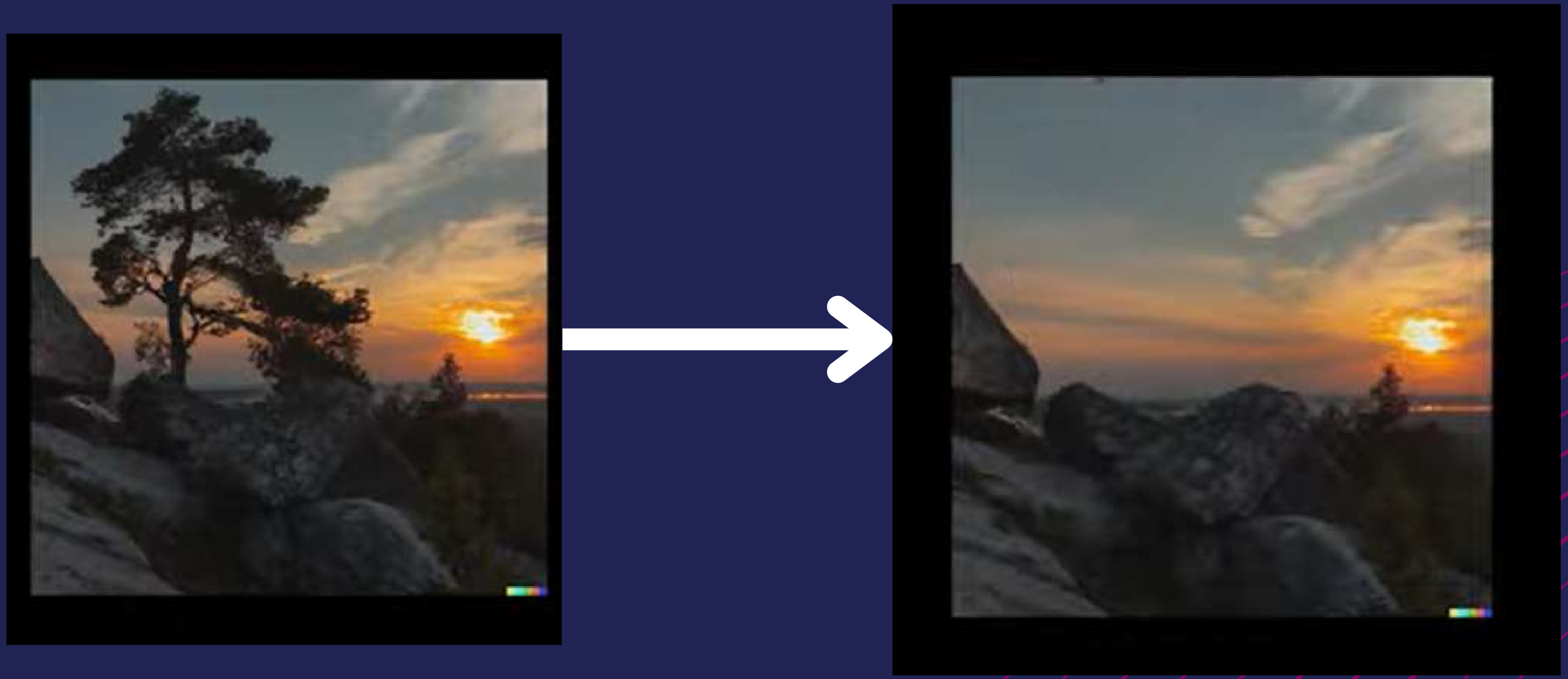


Image Inpainting

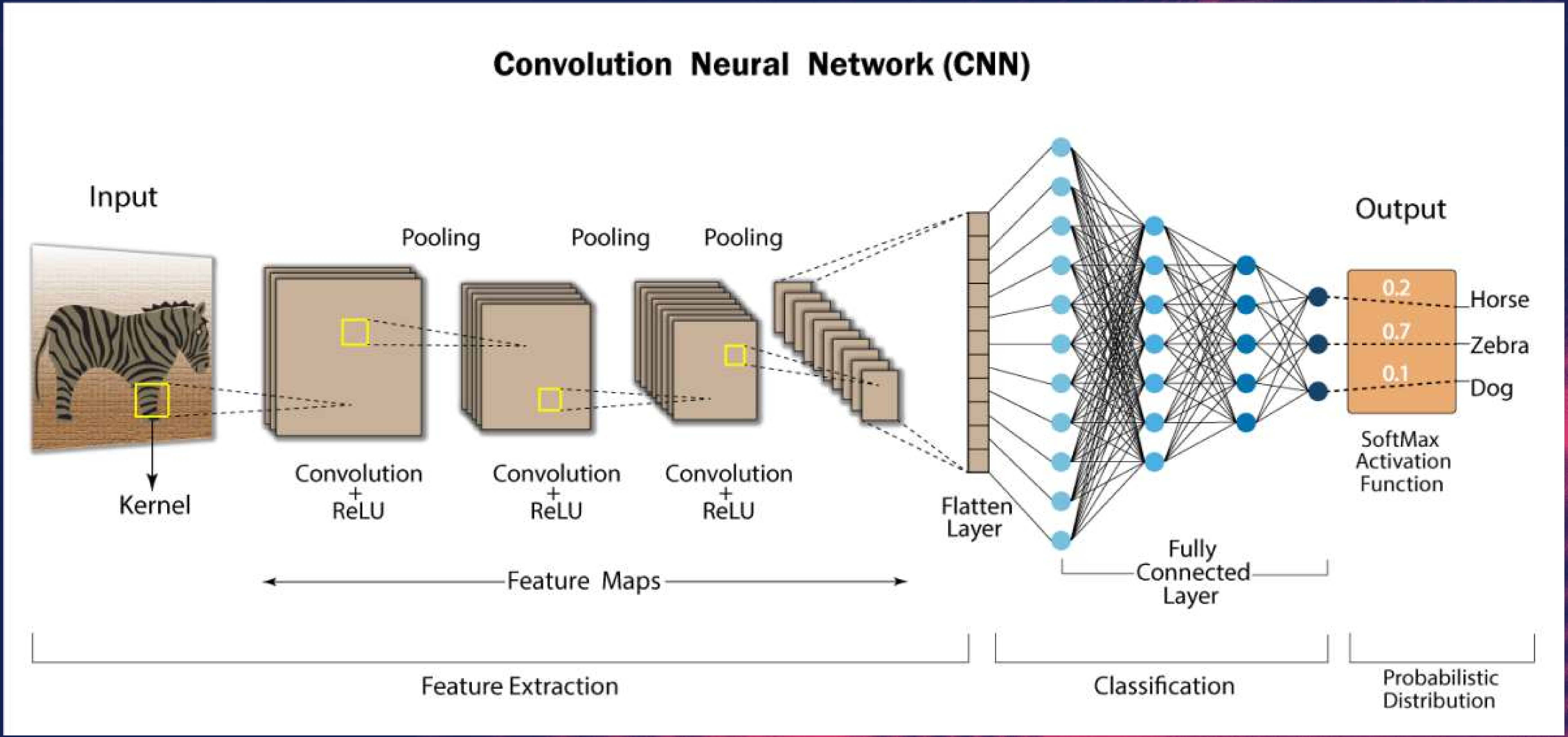


Since we are working on images, we need to first learn how to extract information from them

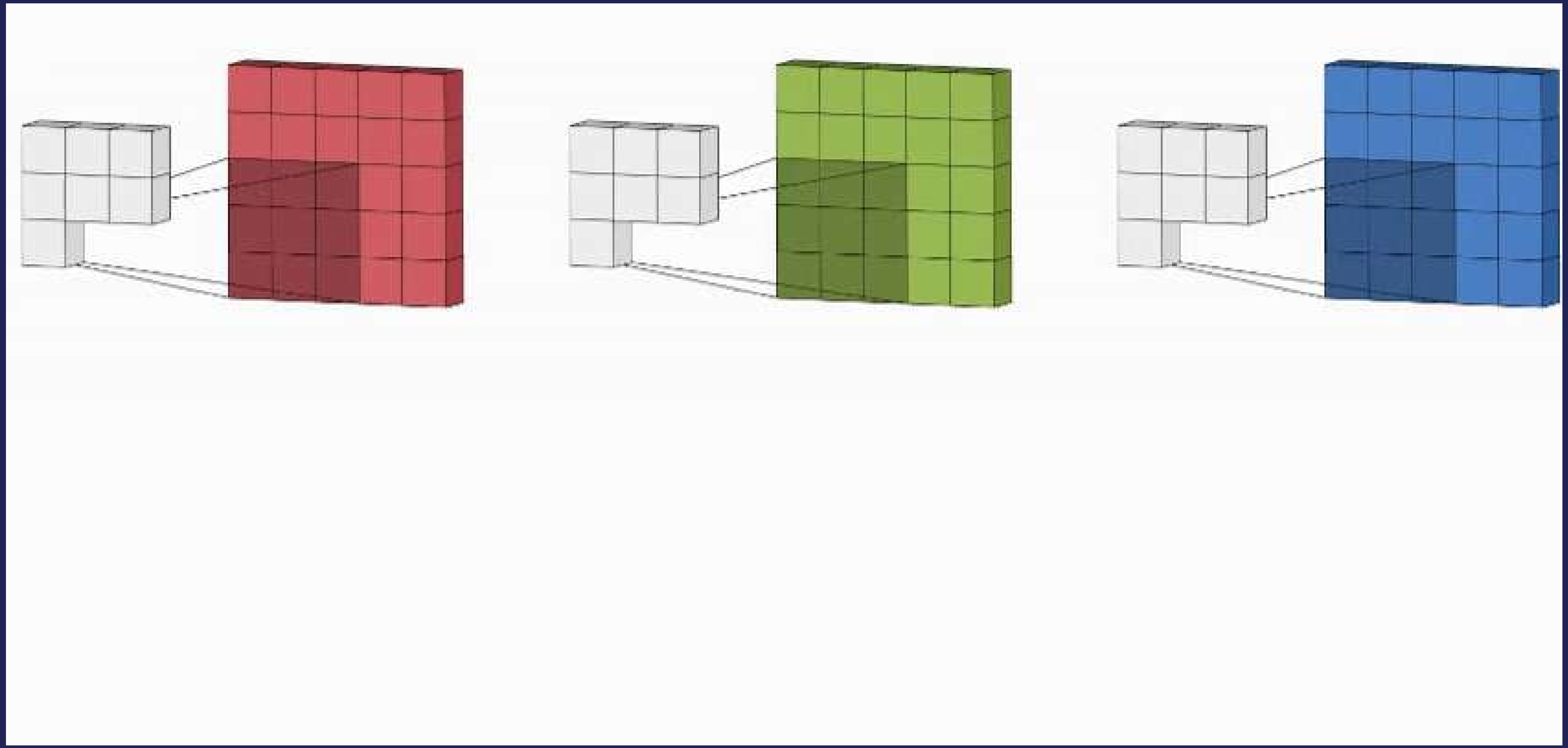
$$\downarrow$$

CNNs

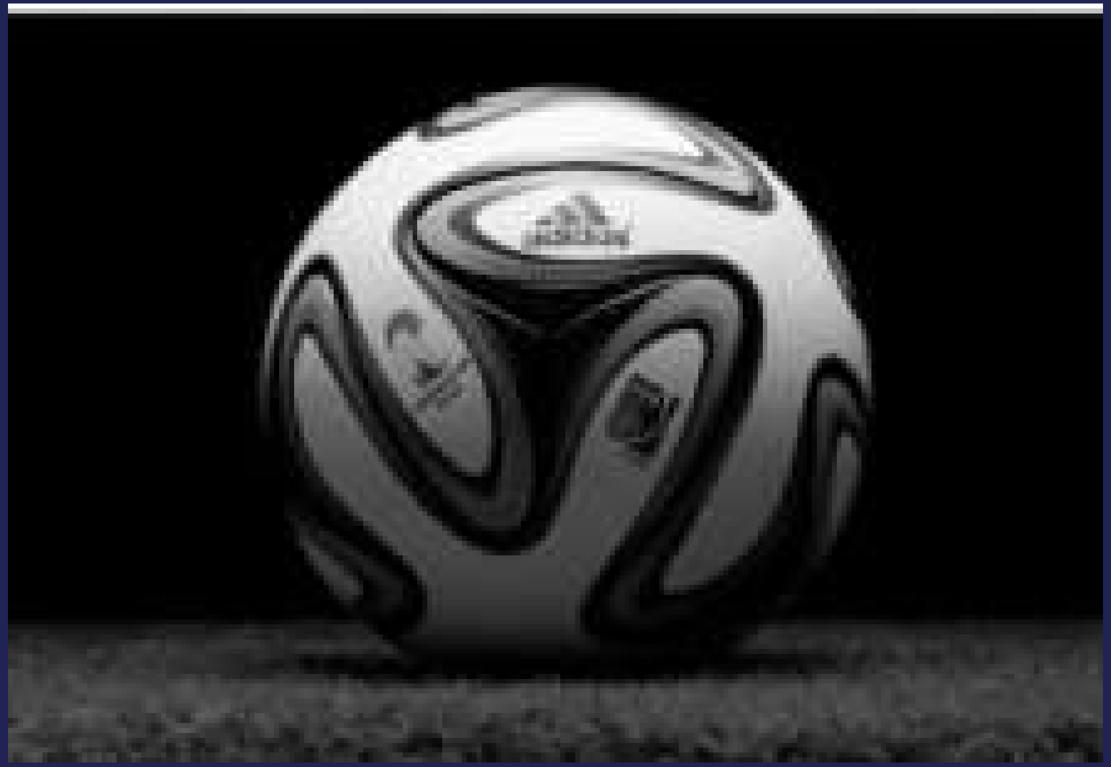
CNNs
Convolutional Neural Networks



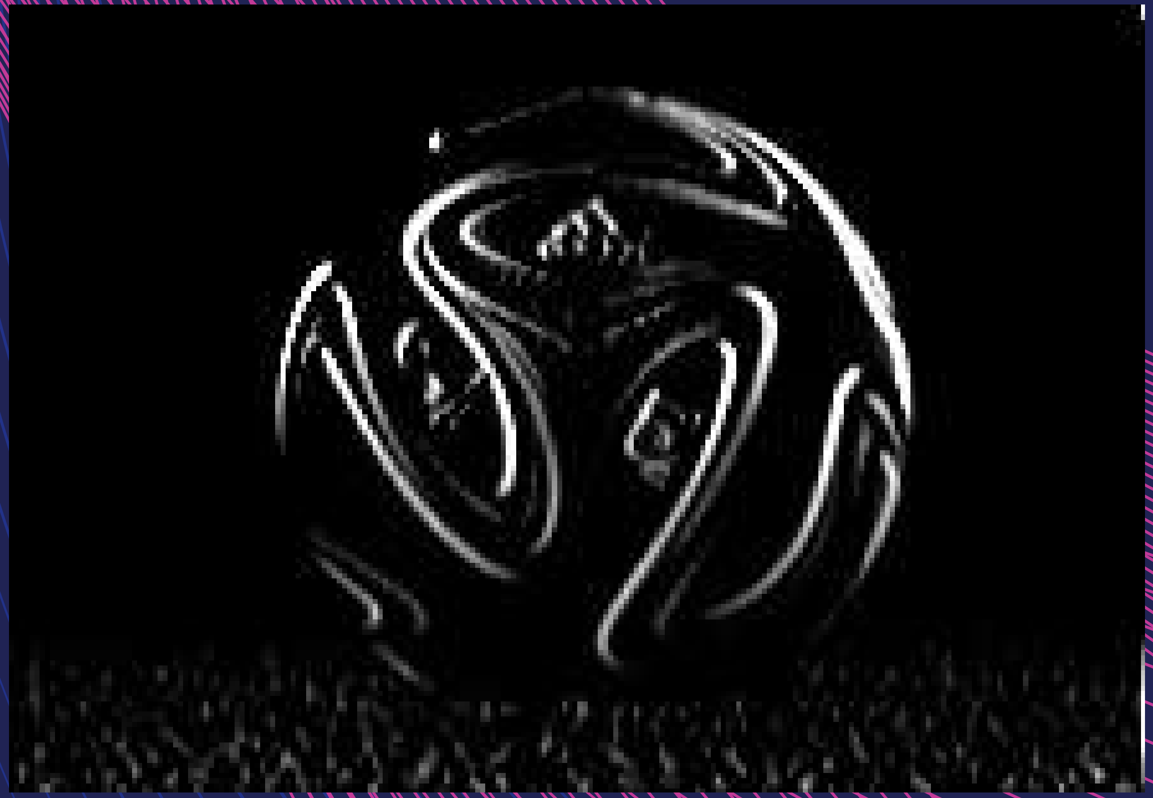
The Convolution Operation



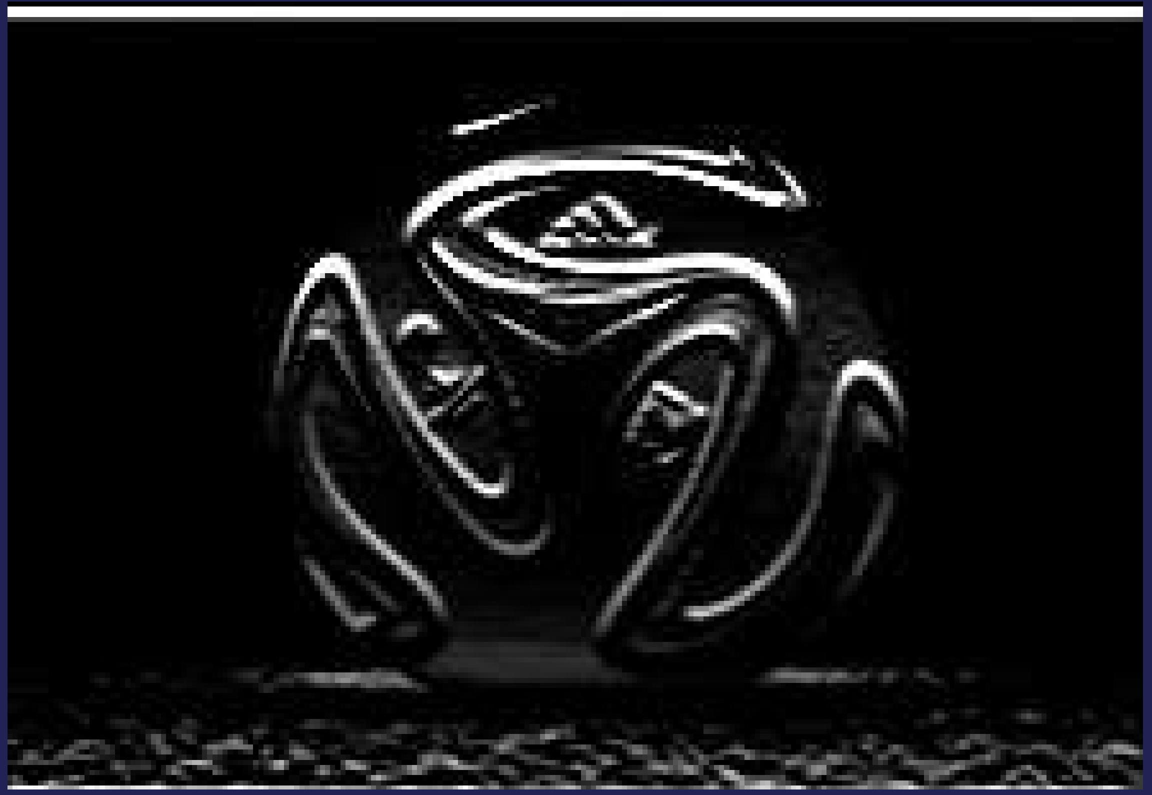
Original image



vertical filter

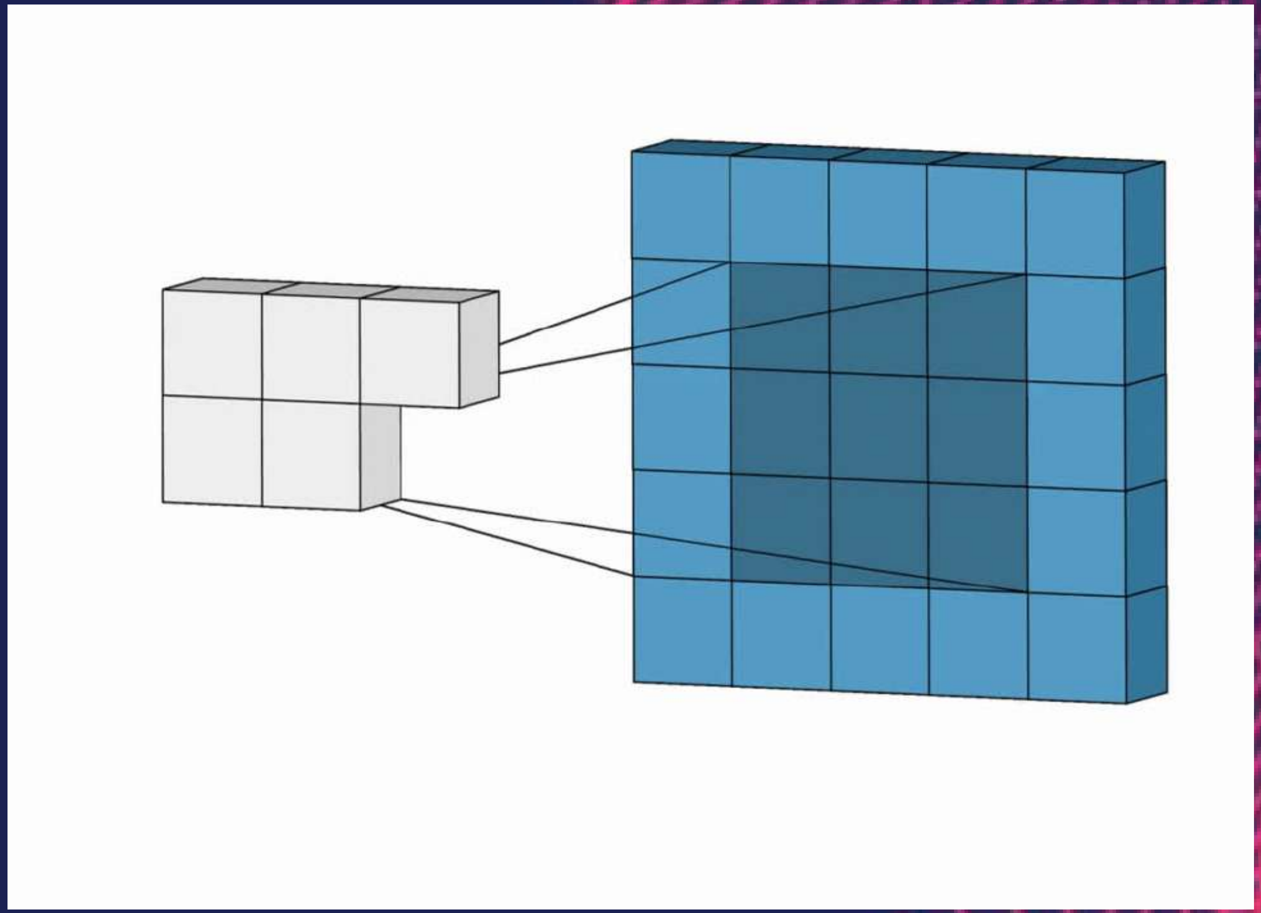


horizontal filter

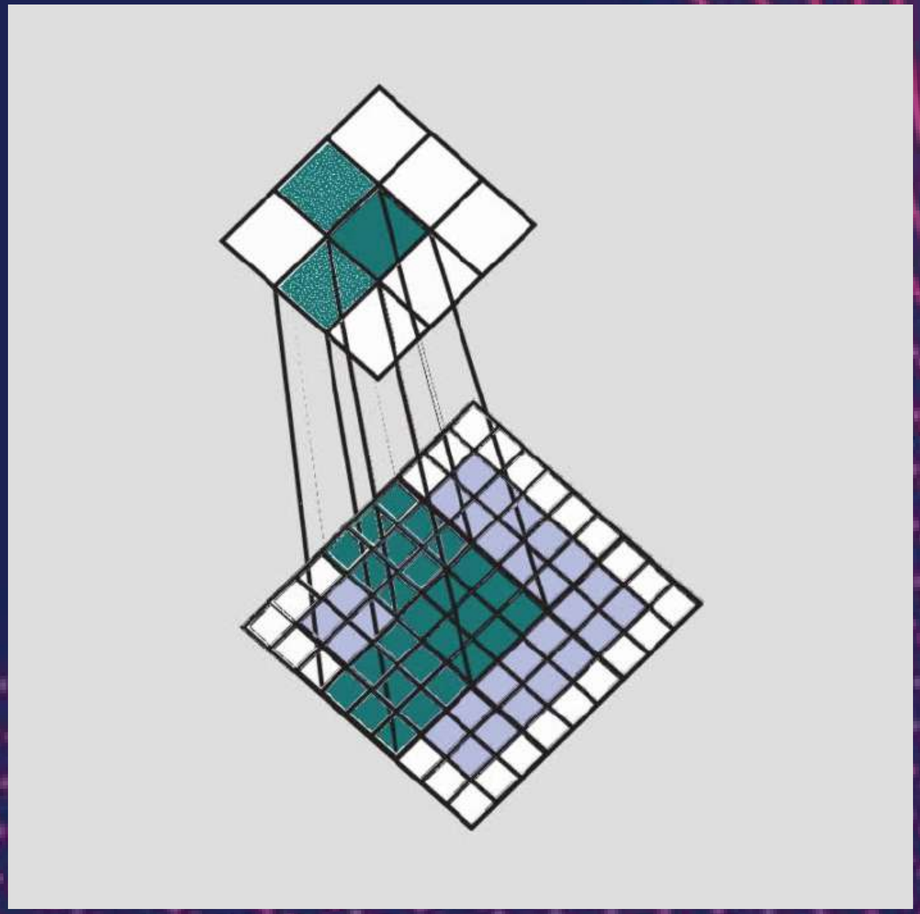


STRIDE

Stride is the no. of steps we move the kernel each time we convolve.



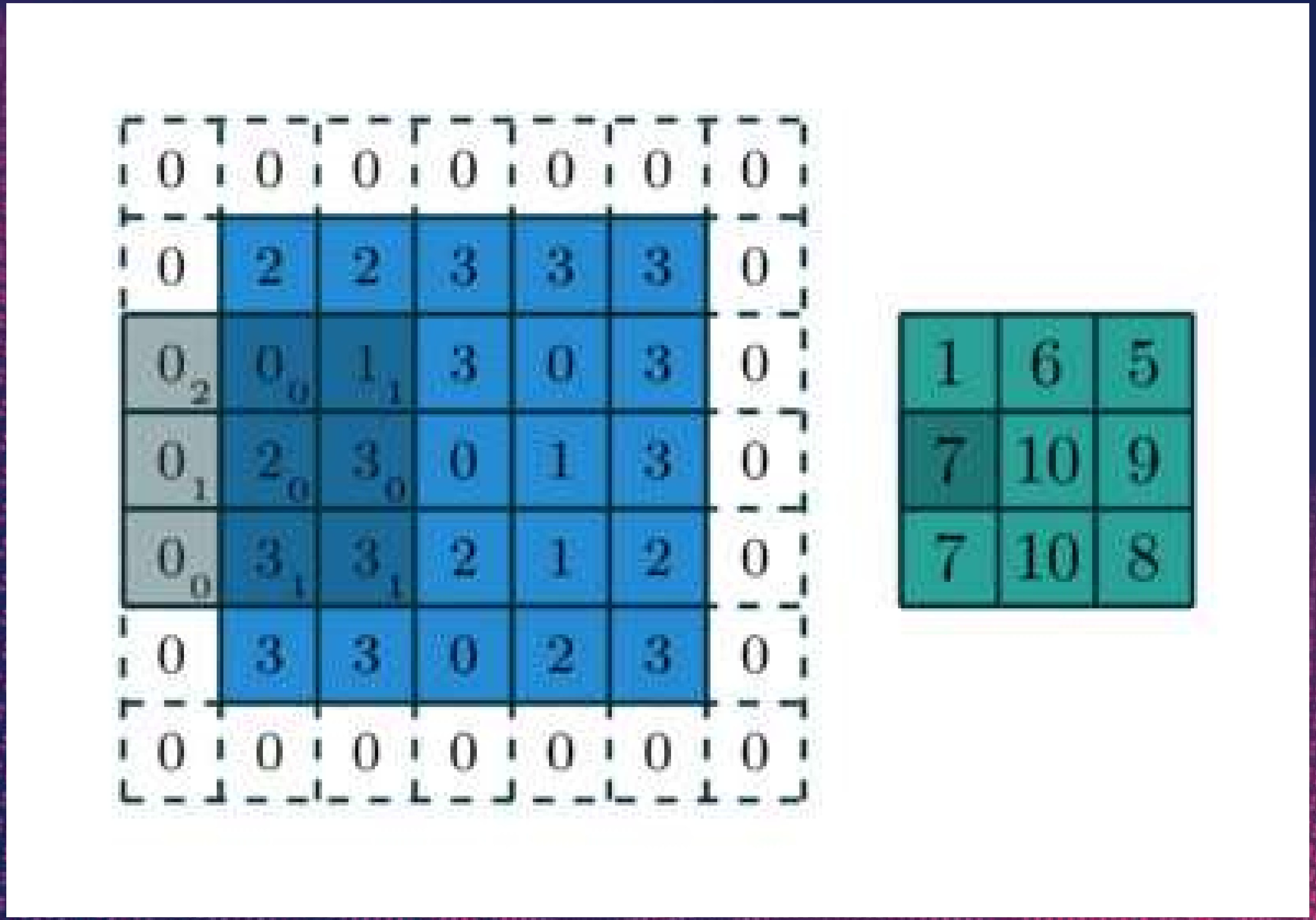
Normally we use a stride of 1, so every possible section of the image is taken.



Padding

During the convolution operation we extract the information of each pixel by applying a kernel over it such the required pixel is at the middle of the kernel. However, we end up losing the data on the edges of the image because we are unable to apply the kernel over those pixels.

So, we do padding!



Pooling Layer

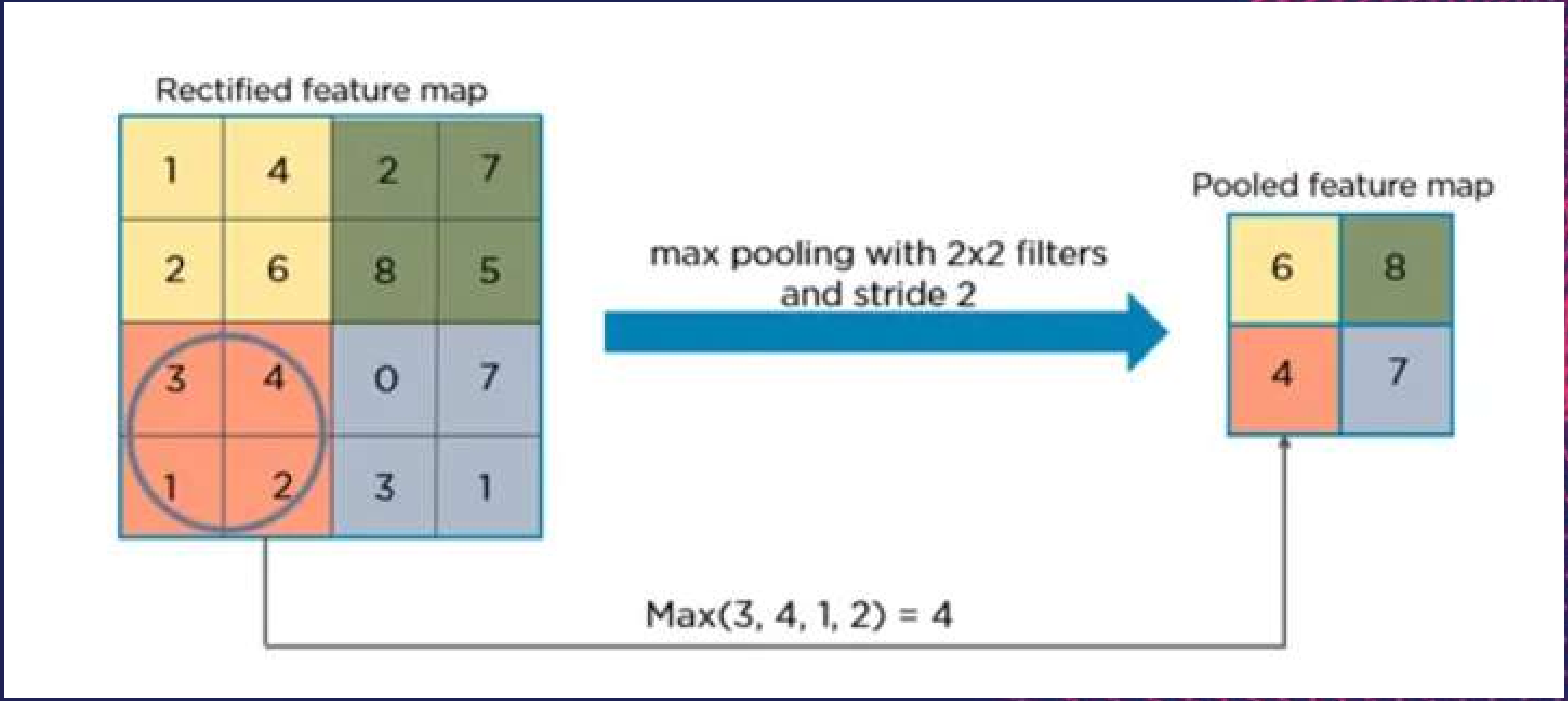
We reduce the dimensionality of the convolved image by pooling it. Normally, a stride of the size of the block is used (so there's no overlap)

This greatly reduces the size of the image, and also removes unnecessary detail (preventing overfitting)

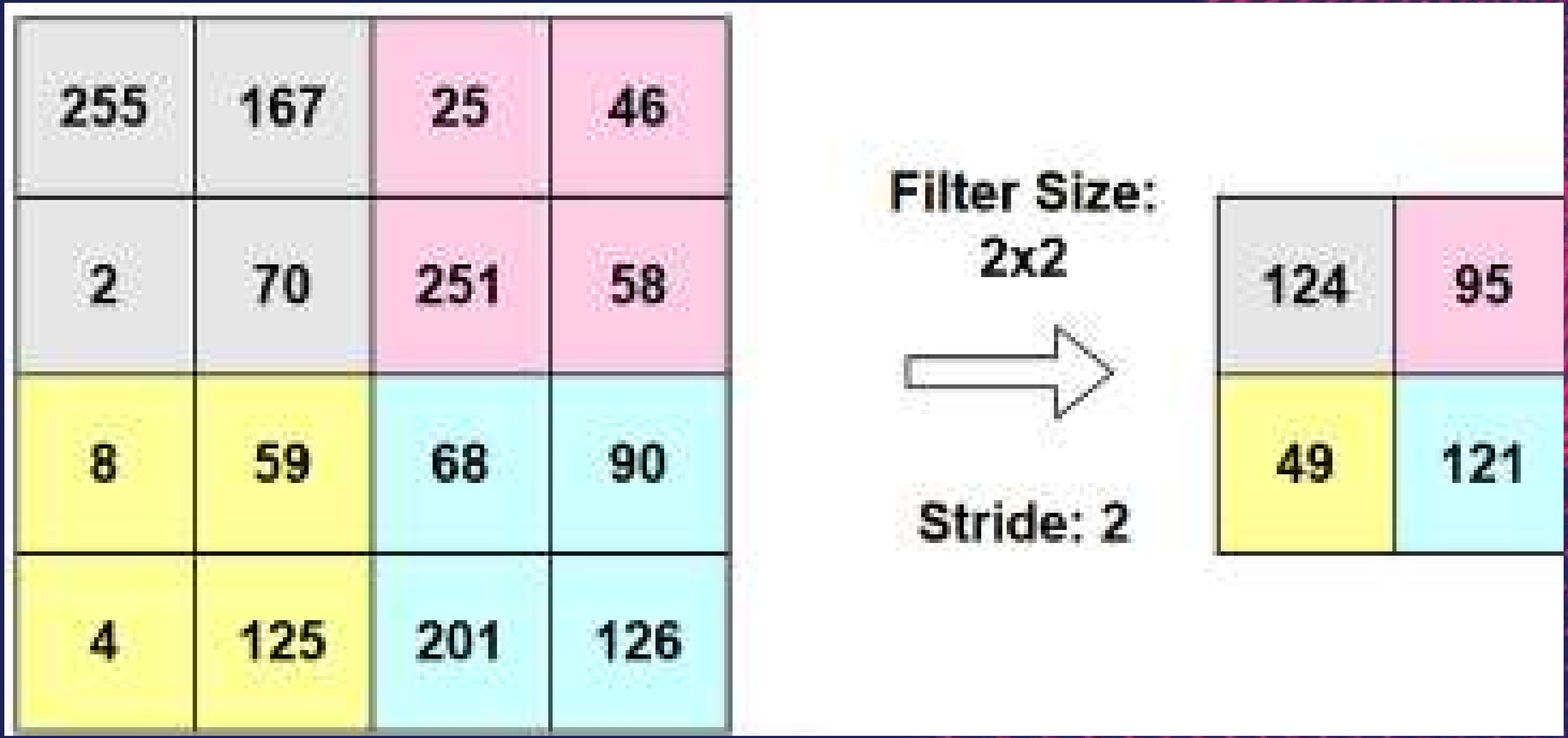
Max Pooling

Average Pooling

Max Pooling


Rectified feature map
1
4
2
7
2
6
8
5
3
4
0
7
1
2
3
1
max pooling with 2x2 filters and stride 2
Pooled feature map
6
8
4
7
Max(3, 4, 1, 2) = 4

Average Pooling



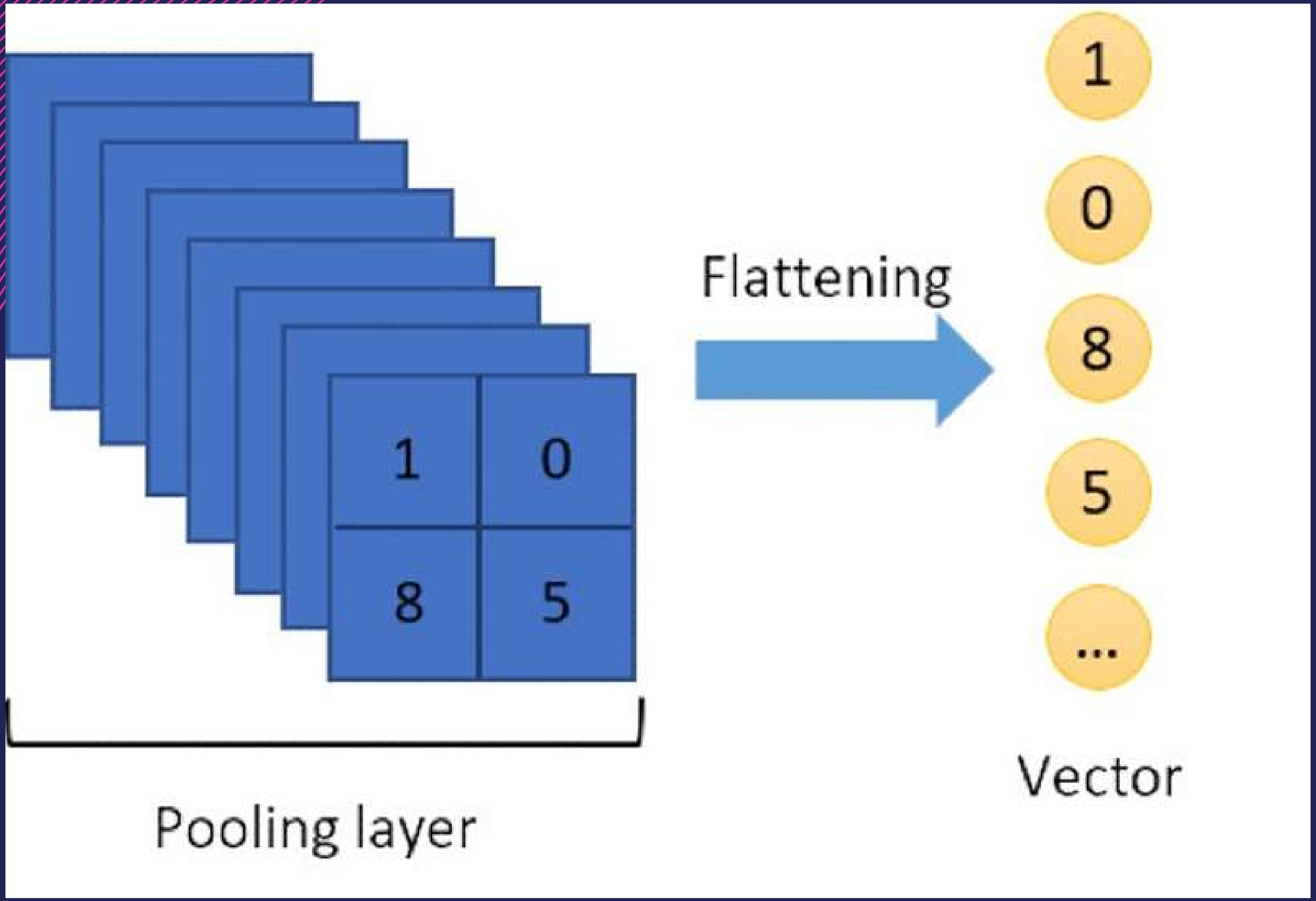
255	167	25	46
2	70	251	58
8	59	68	90
4	125	201	126

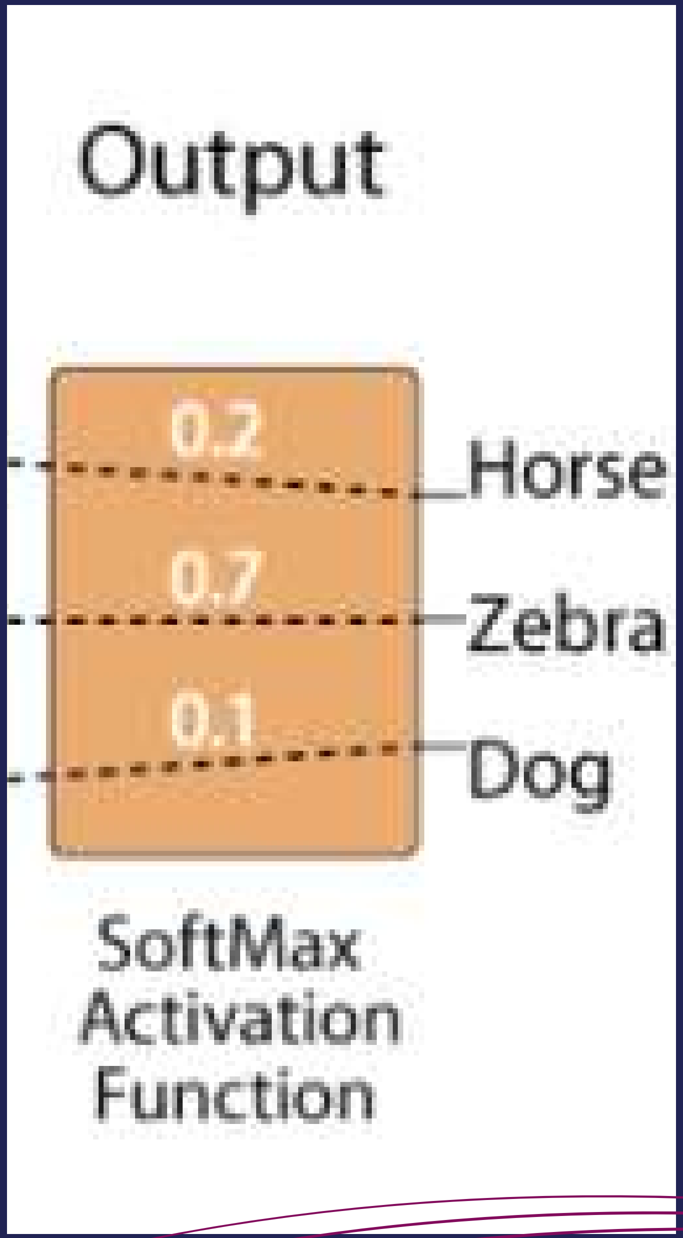
Filter Size: 2x2

Stride: 2

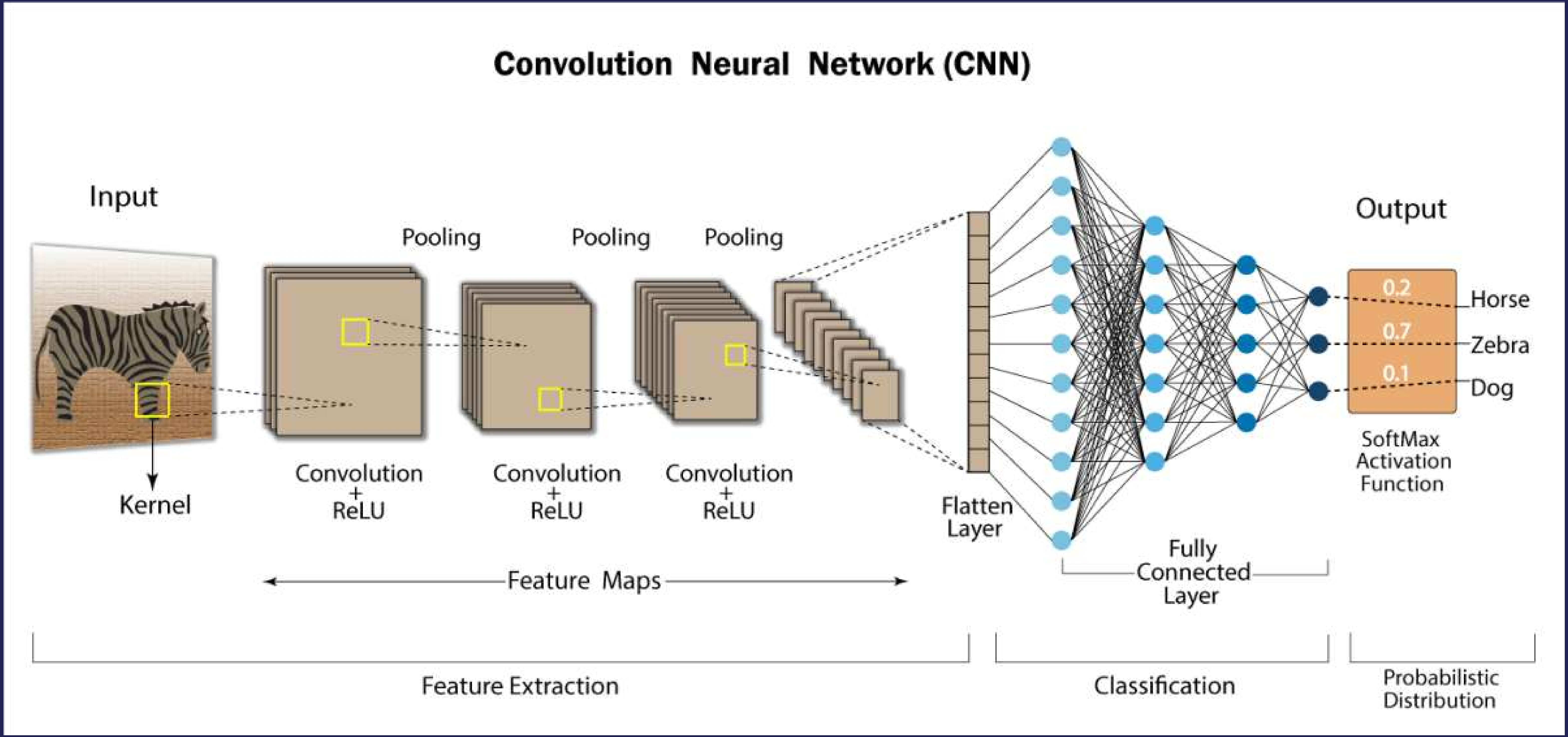
124	95
49	121

Flattening Layer+SoftMax





The full architecture



Why CNNs?
Why not just use vanilla neural networks?

ADVANTAGES OF CNNS

Reduced Memory Requirements

Due to parameter sharing and local connectivity, CNNs often require less memory compared to fully connected networks. This is especially important when processing high-dimensional inputs like images.

Translation Invariance

Convolutional layers provide translation invariance, meaning that the model can recognize patterns regardless of their position in the input space

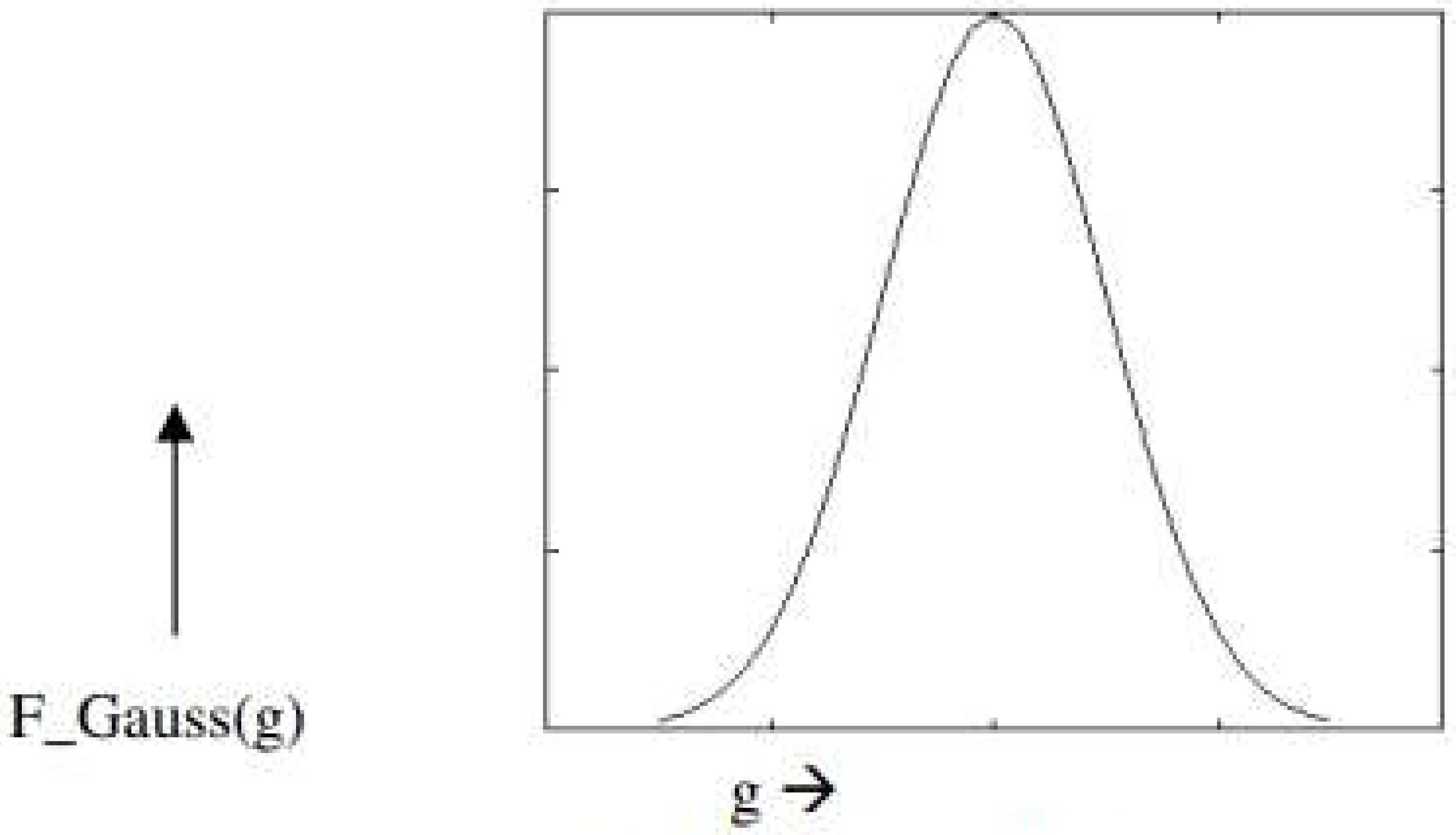
ADVANTAGES OF CNNS

Parameter Sharing:

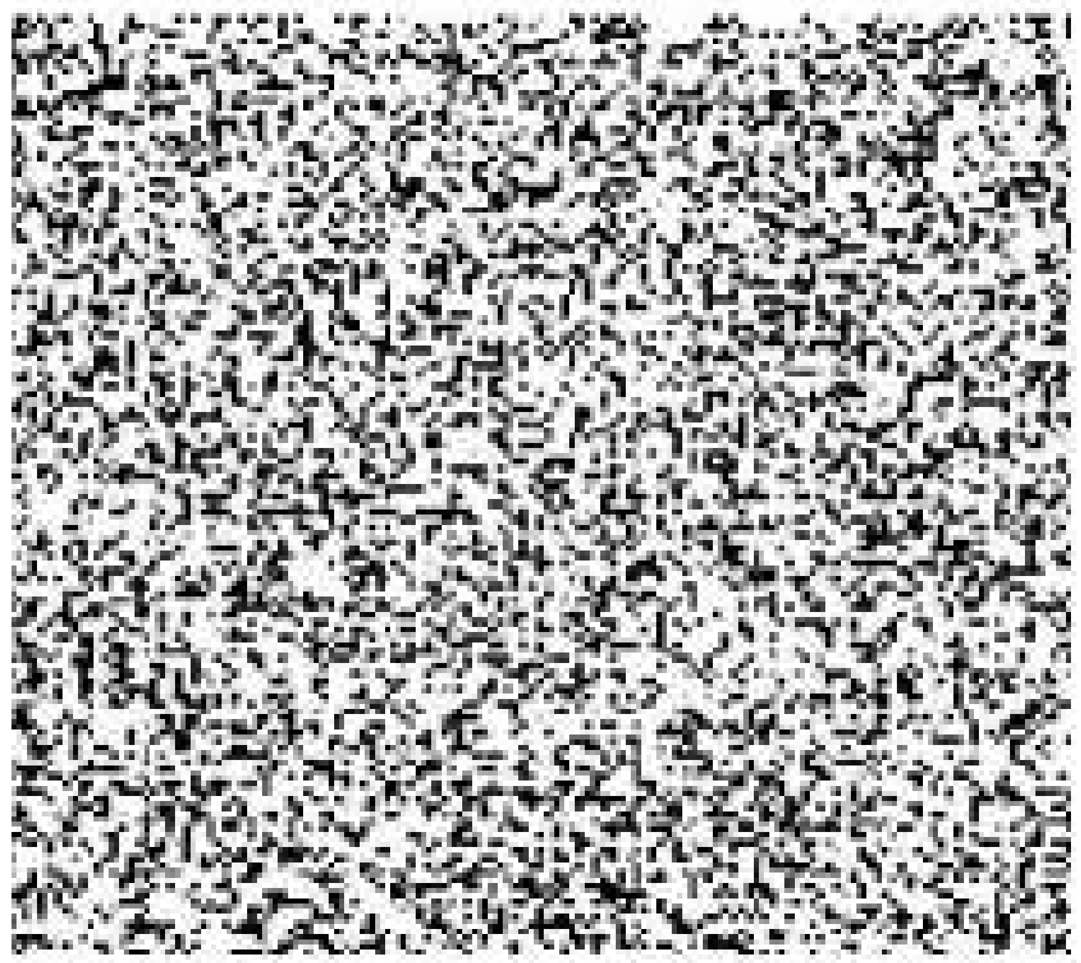
In convolutional layers, parameters (weights and biases) are shared across different regions of the input. This sharing reduces the total number of parameters, making the model more efficient and reducing the risk of overfitting.

A Gentle Introduction to GENERATIVE ADVERSERIAL NETWORKS

What is noise



Gaussian Distribution function



Gaussian noise

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad X \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

NORMAL/GAUSSIAN

Random variables hold values derived from the outcomes of random experiments. For example, random variable X holds the number of heads in flipping a coin 100 times.

If we normalize *n* independent random variables and repeat the experiment many times, the collected results tend toward a normal distribution when the sample size (*n*) gets larger

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

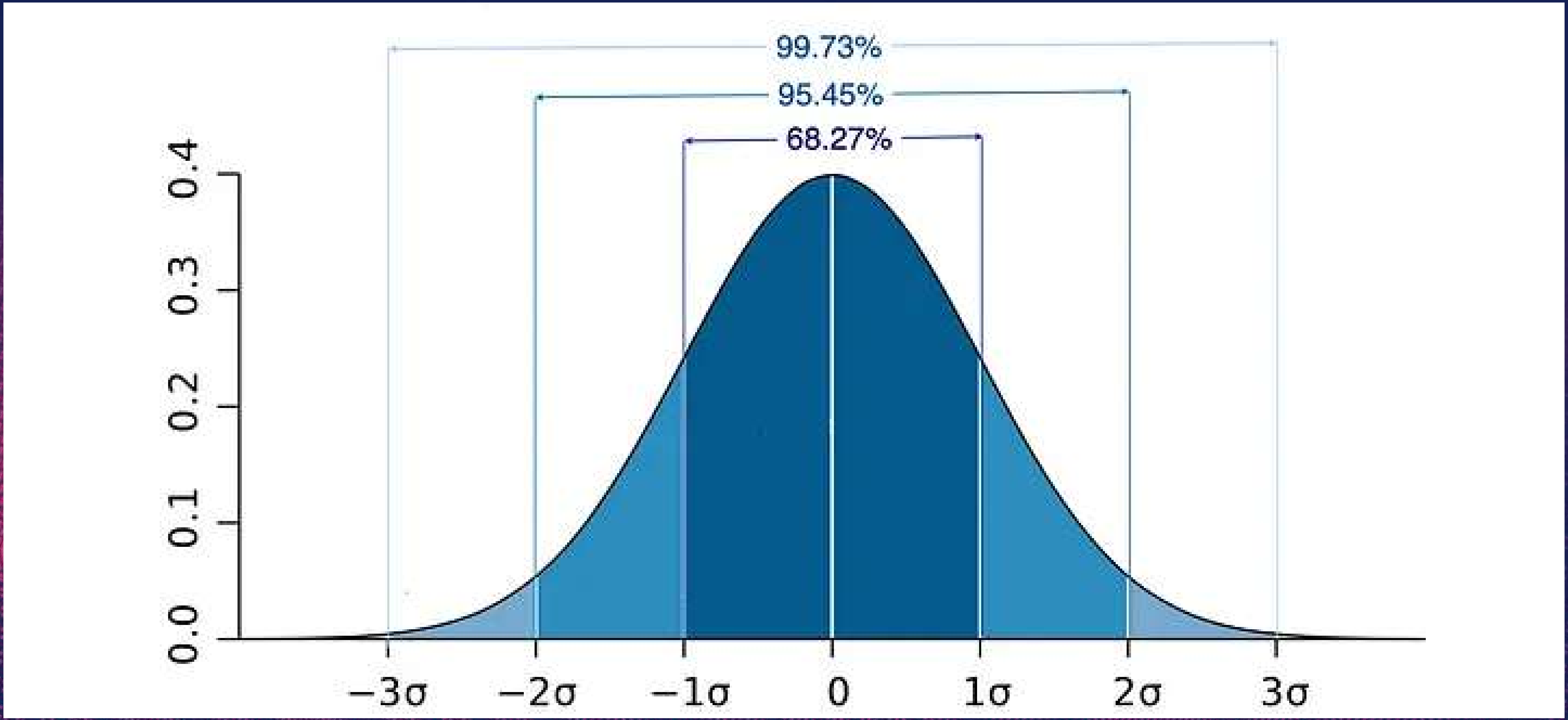
univariate normal distribution

$$p(x|\mu, \sigma^2)$$

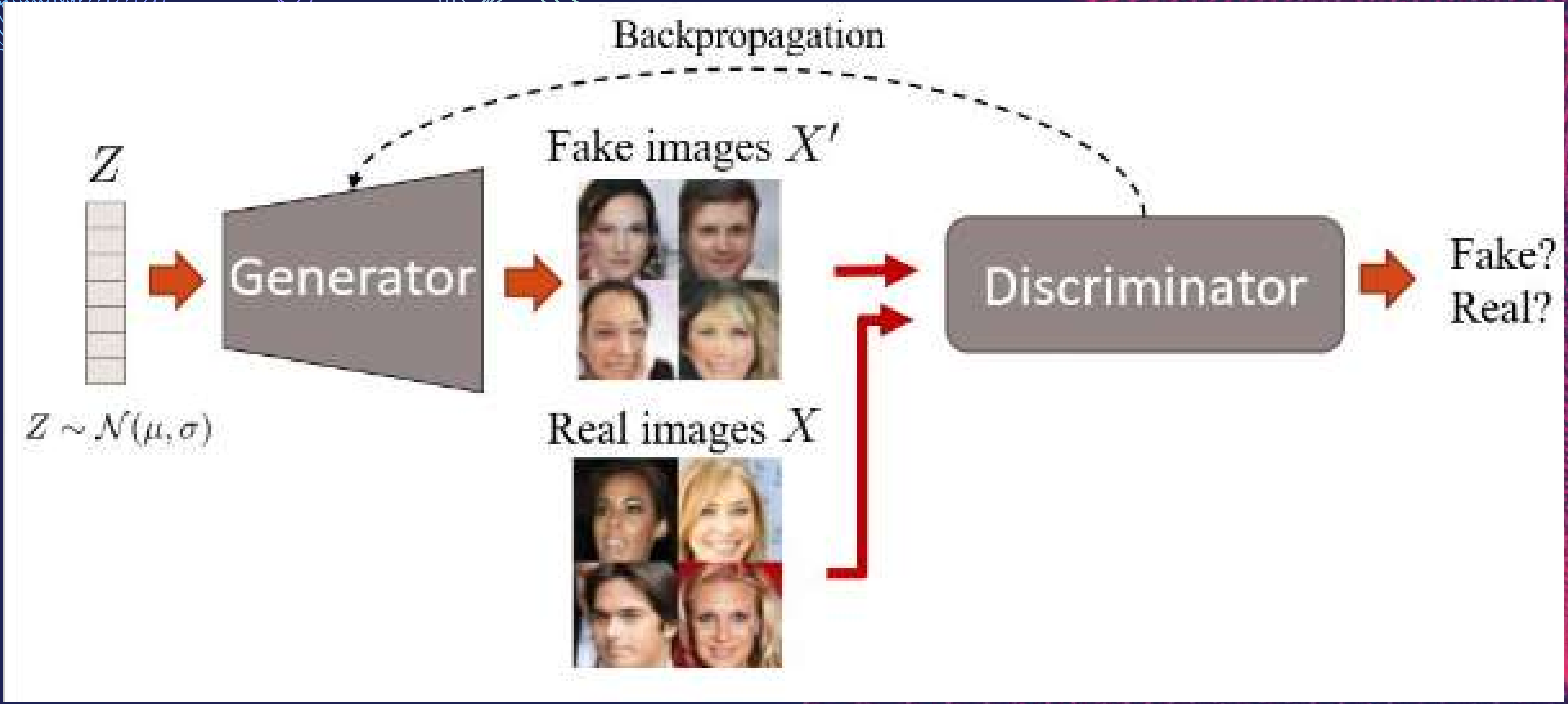
$$f_{\mathbf{X}}(x_1, \ldots, x_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}}$$

multivariate normal distribution

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$


99.73%
95.45%
68.27%
0.4
0.3
0.2
0.1
0.0
−3σ
−2σ
−1σ
0
1σ
2σ
3σ

The main architecture



THE MIN MAX LOSS FUNCTION

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

That looks extemely complicated yet it is devilishly simple. Let’s break it down.

Know what this loss function is?

$$L(y, \hat{y}) = -(y.log\hat{y} + (1 - y).\log(1 - \hat{y}))$$

The Discriminator Loss

$$L(1, \hat{y}) = -(1.\log(D(x)) + (1-1).\log(1-D(x)) = -\log(D(x))$$

Where, $\hat{y} = D(x)$ → output of the discriminator for a real input image

$$L(0, \hat{y}) = -(0.\log(D(G(z))) + (1-0).\log(1-D(G(z))) = -\log(1-D(G(z)))$$

Where, $\hat{y} = D(G(z))$
z is the input noise vector provided to the generator
$G(z)$ is the output of the generator
$D(G(z))$ is the output of the discriminator with generated data as the input

$$L(Discriminator) = -[\log(D(x)) + \log\left(1 - D(G(z))\right)]$$

The Discriminator Loss

$$L(Discriminator\ ) = \min[-[\log\big(D(x)\big) + \log\Big(1 - D\big(G(z)\big)\Big)]]$$

$$L(Discriminator\ ) = \max[\log\big(D(x)\big) + \log\Big(1 - D\big(G(z)\big)\Big)]$$

The Generator Loss

$$L(0, \hat{y}) = -(0. \log(D(G(z))) + (1-0). \log(1 - D(G(z))) = -\log(1 - D(G(z)))$$

Where, $\hat{y} = D(G(z))$
z is the input noise vector provided to the generator
$G(z)$ is the output of the generator
$D(G(z))$ is the output of the discriminator with generated data as the input

$$L(Generator\ ) = min[\log(1 - D\big(G(z)\big))]$$

Effective Loss

$$L(GAN) = GminDmax\,[\log(D(x)) + \log\left(1 - D(G(z))\right)]$$

$$L(GAN) = GminDmax\left[E_{x(real)}\log(D(x)) + E_{z(fake)}\log\left(1 - D(G(z))\right)\right]$$

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

DCGAN

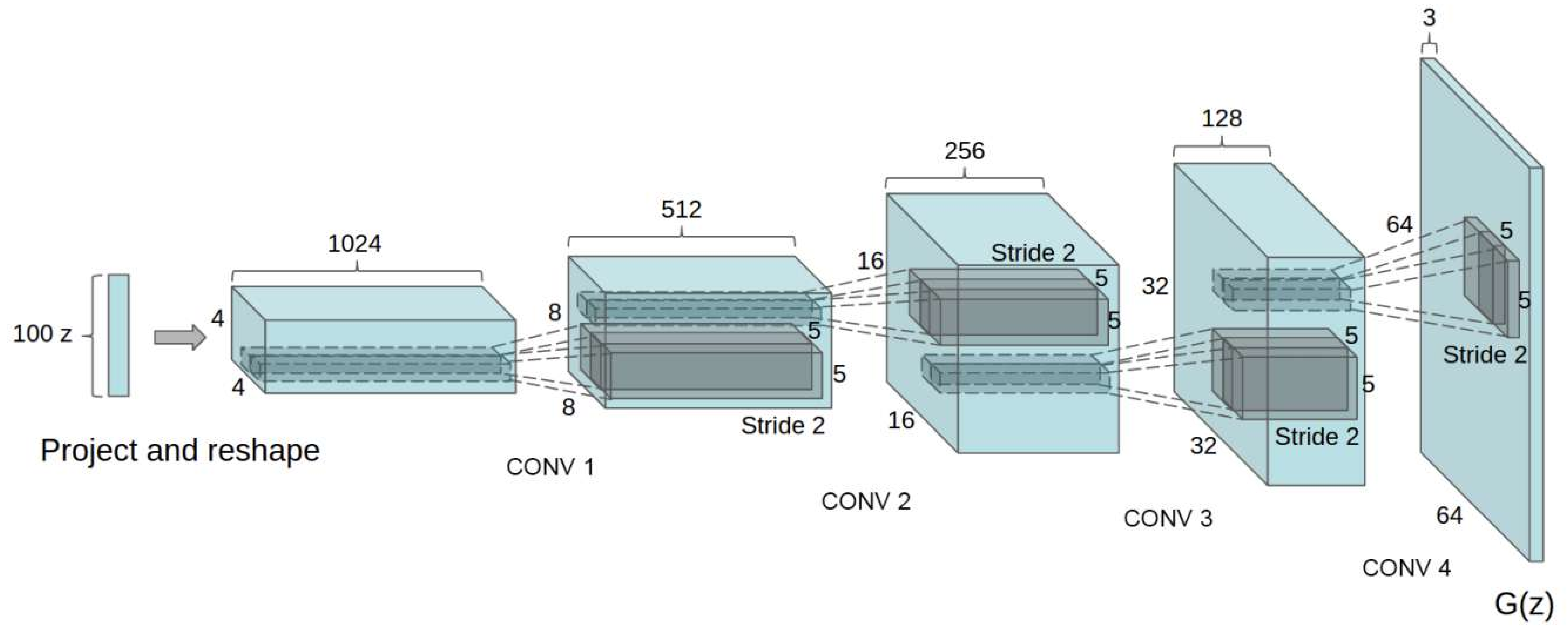
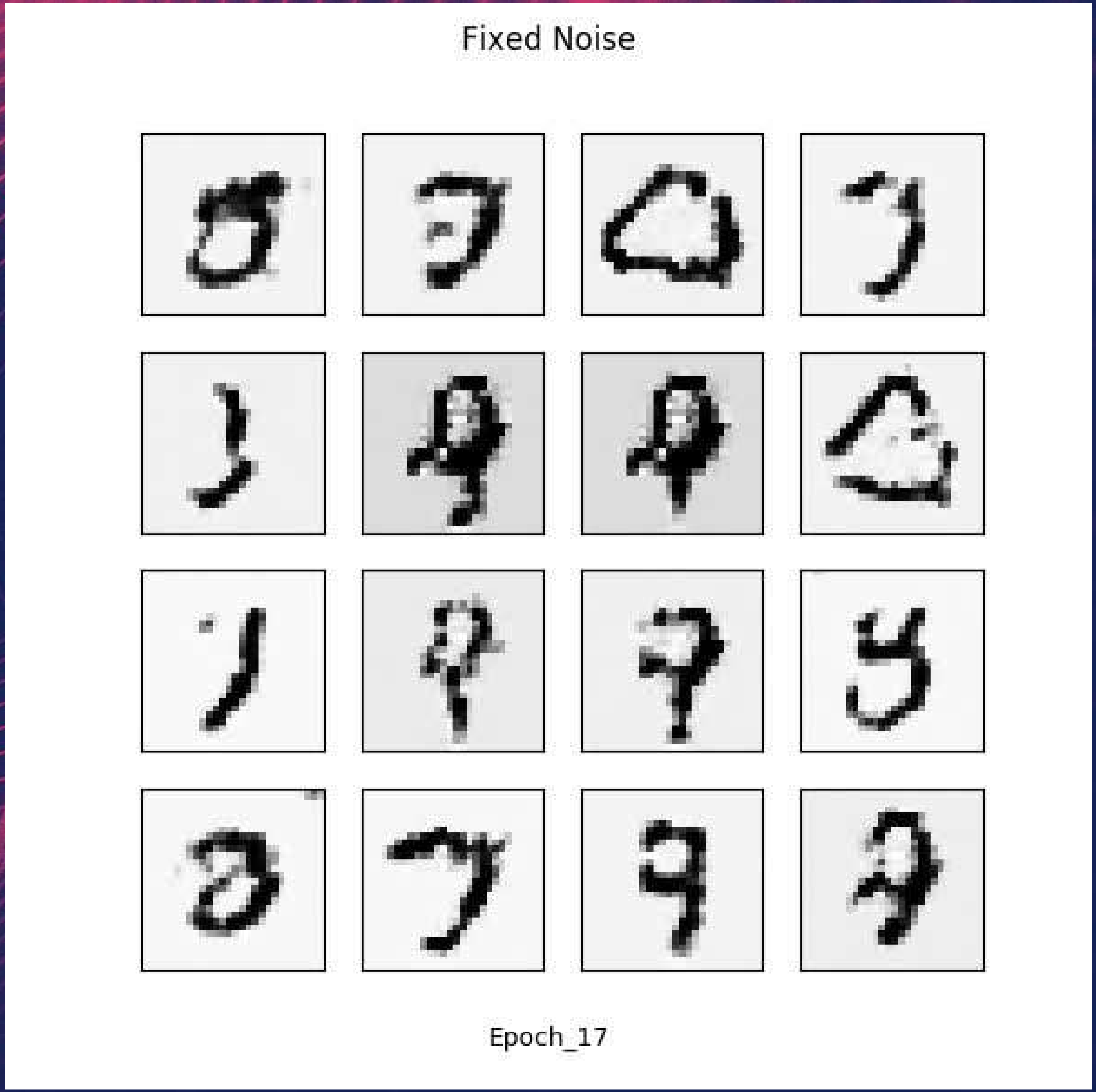
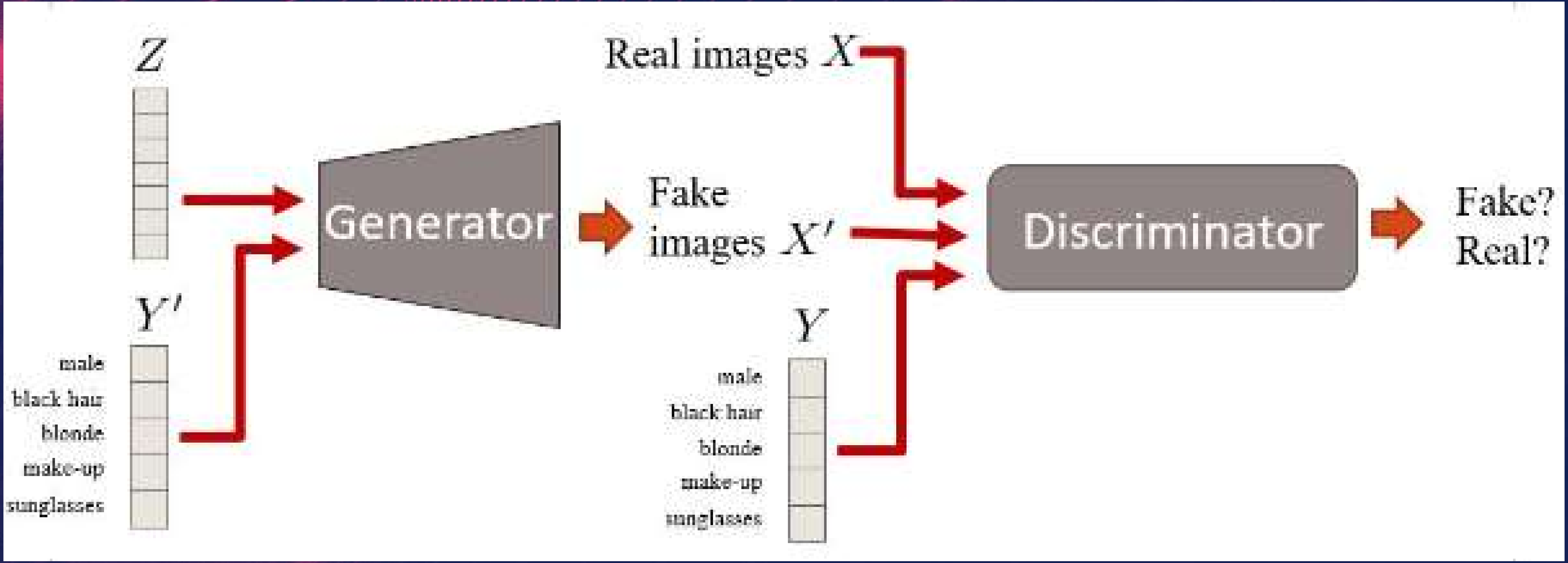
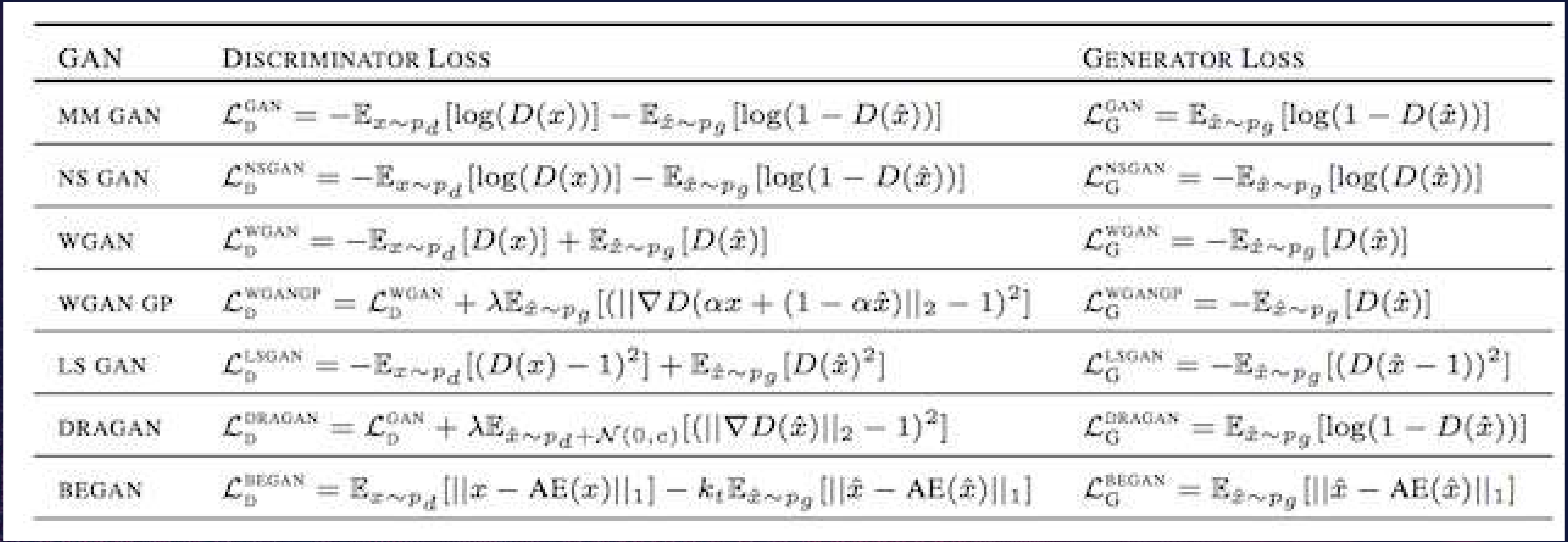


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.


Fixed Noise
Epoch_17

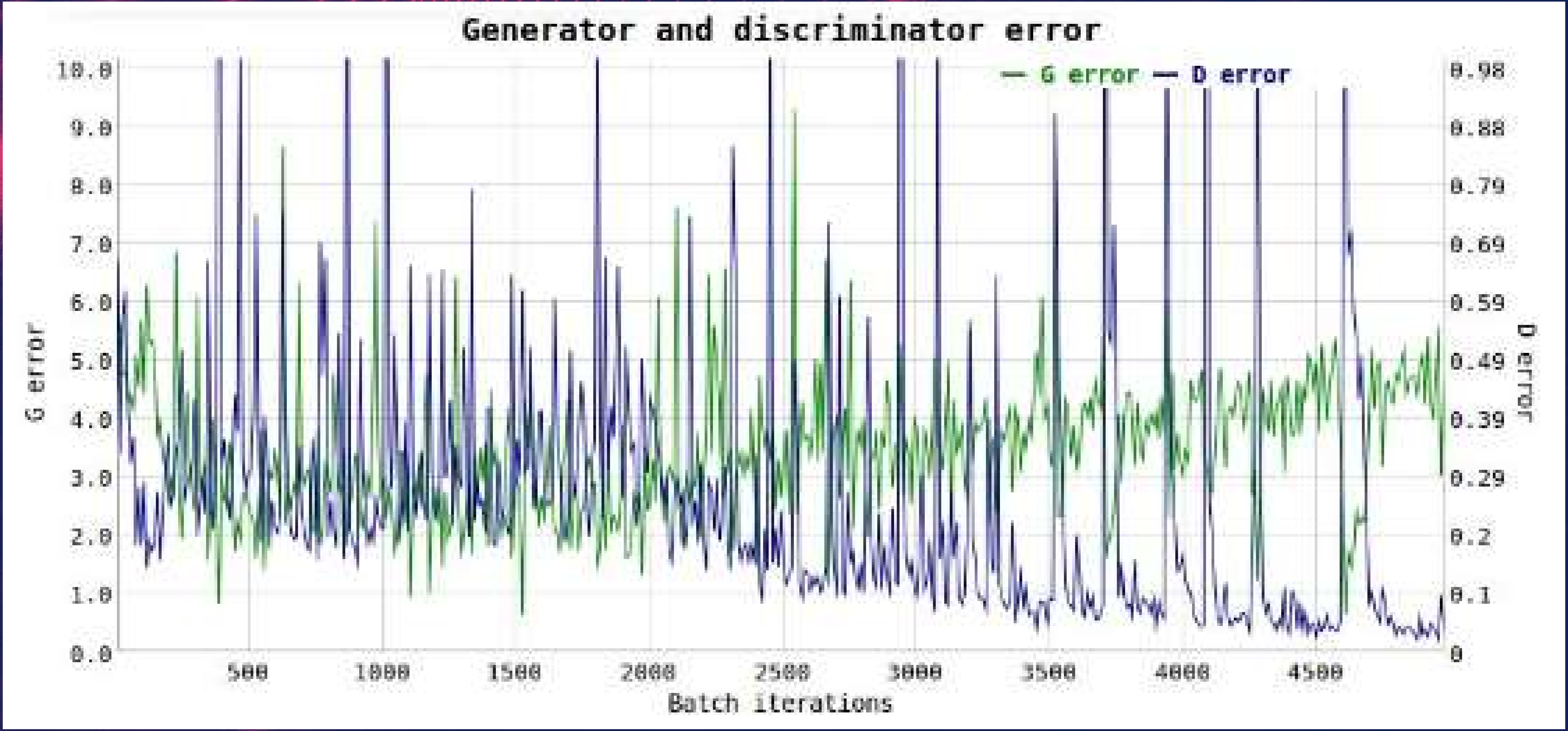
CGAN


Z
Y'
male
black hair
blonde
make-up
sunglasses
Generator
Real images X
Fake images X'
Y
male
black hair
blonde
make-up
sunglasses
Discriminator
Fake?
Real?

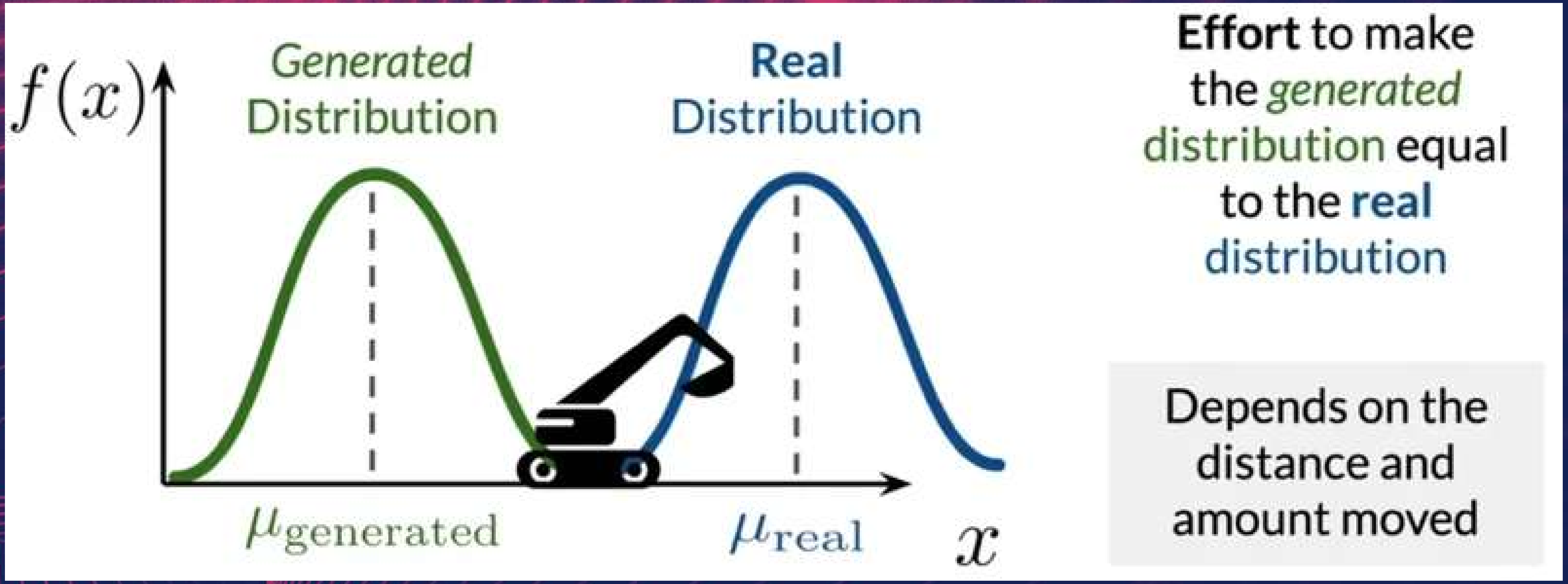


GAN	Discriminator Loss	Generator Loss
MM GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{GAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$
NS GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{NSGAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{NSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} = -\mathbb{E}_{x\sim p_d}[D(x)] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
WGAN GP	$\mathcal{L}_{\mathrm{D}}^{\mathrm{WGANGP}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_g}[(\lvert\lvert\nabla D(\alpha x + (1 - \alpha\hat{x})\rvert\rvert_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{WGANGP}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{LSGAN}} = -\mathbb{E}_{x\sim p_d}[(D(x) - 1)^2] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{LSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[(D(\hat{x} - 1))^2]$
DRAGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{DRAGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_d + \mathcal{N}(0,c)}[(\lvert\lvert\nabla D(\hat{x})\rvert\rvert_2 - 1)^2]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{DRAGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$
BEGAN	$\mathcal{L}_{\mathrm{D}}^{\mathrm{BEGAN}} = \mathbb{E}_{x\sim p_d}[\lvert\lvert x - \mathrm{AE}(x)\rvert\rvert_1] - k_t\mathbb{E}_{\hat{x}\sim p_g}[\lvert\lvert\hat{x} - \mathrm{AE}(\hat{x})\rvert\rvert_1]$	$\mathcal{L}_{\mathrm{G}}^{\mathrm{BEGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\lvert\lvert\hat{x} - \mathrm{AE}(\hat{x})\rvert\rvert_1]$

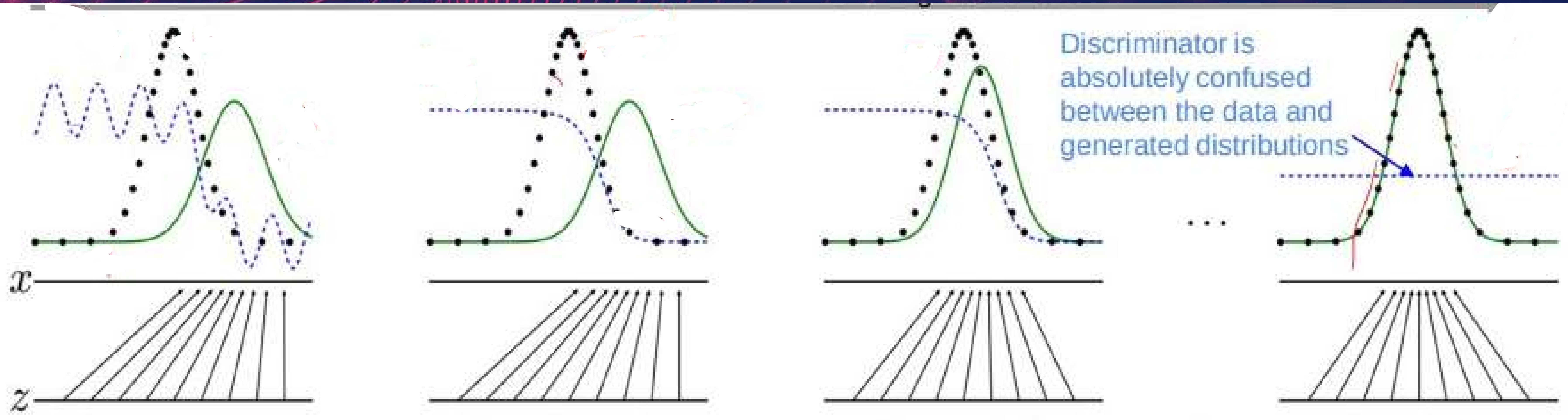
Do you notice any issue?

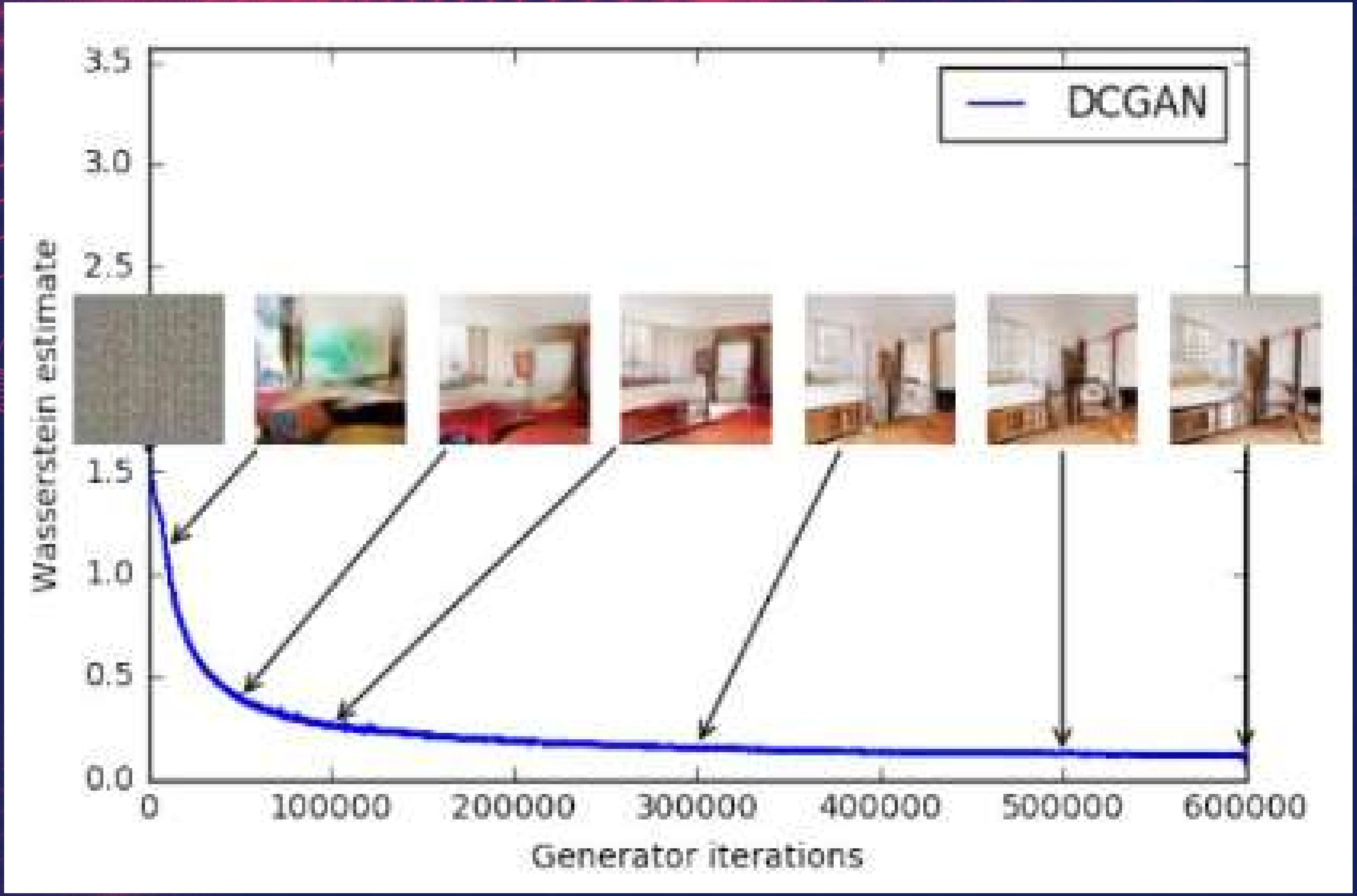


WGAN to the Rescue


$f(x)$
Generated Distribution
Real Distribution
Effort to make the generated distribution equal to the real distribution
Depends on the distance and amount moved
$\mu_{\text{generated}}$
μ_{real}
x

The Wasserstein distance is defined as the amount of work needed to be done to change one distribution to another. In terms of the above curve, it is defined as the area of the shaded region on the right.The Wasserstein distance is defined as the amount of work needed to be done to change one distribution to another. In terms of the above curve, it is defined as the area of the shaded region on the right.


Discriminator is absolutely confused between the data and generated distributions
x
z
. . .


DCGAN
Wasserstein estimate
Generator iterations
3.5
3.0
2.5
1.5
1.0
0.5
0.0
0
100000
200000
300000
400000
500000
600000