Deep Learning Bootcamp with PyTorch

Noel Shin

Contents

Introduction

Goal of lecture

Getting used to

- **read a PyTorch code of interest,**
- **make a deep learning model using PyTorch.**

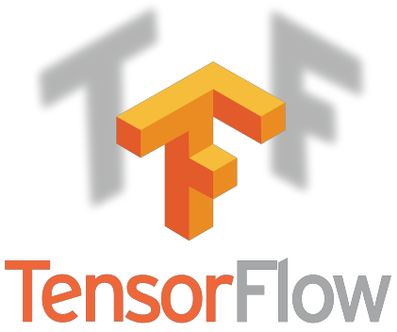
What is PyTorch?

PyTorch is an **open-source machine learning library** for Python, based on Torch.

- **Tensor computation with strong GPU acceleration***

- Deep neural networks built on a tape-based **autograd system**

Why PyTorch?



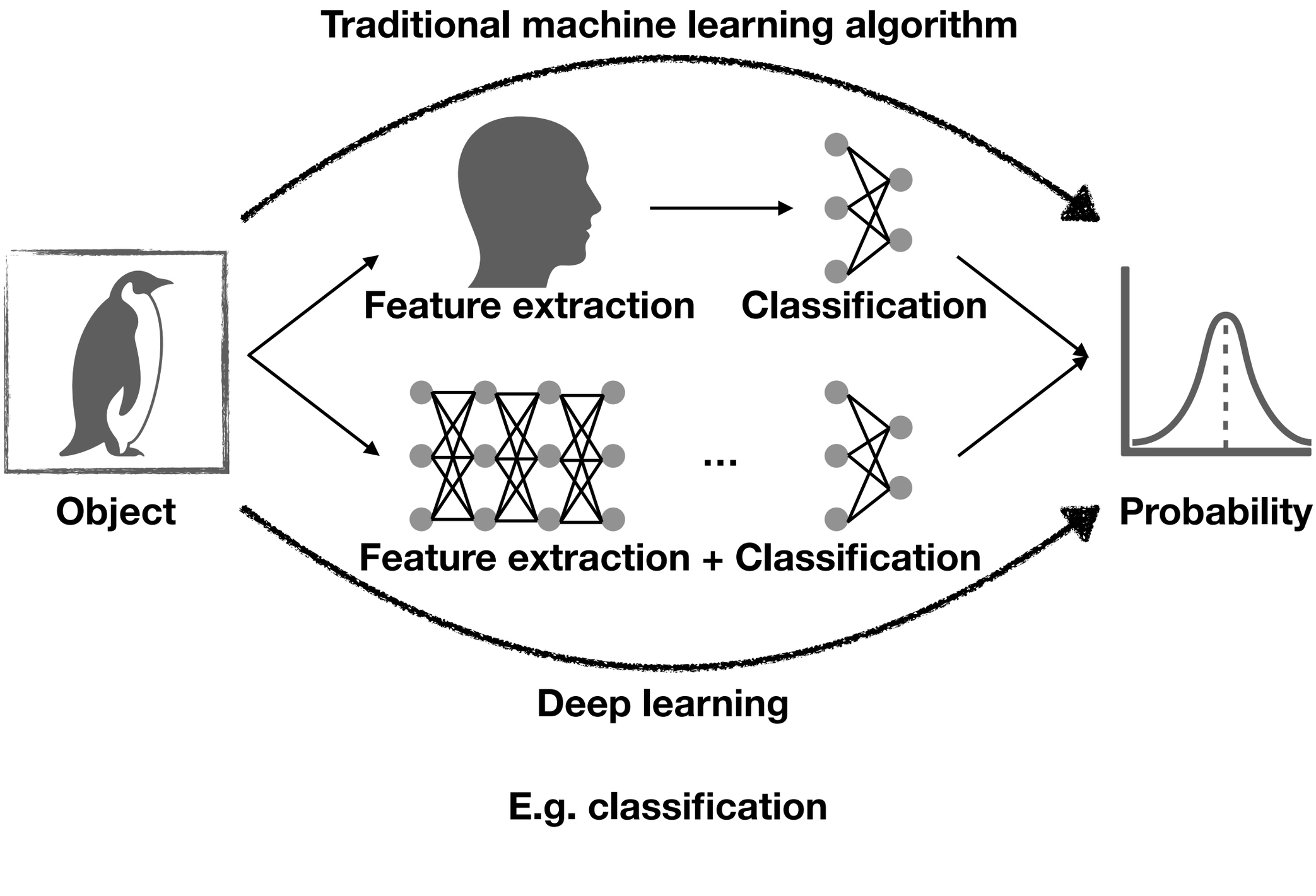
- A lot easier to learn than to learn TensorFlow.

- Available to customize a model. This is usually infeasible with Keras.

- Public codes for deep learning research papers are usually written with PyTorch.

Basics

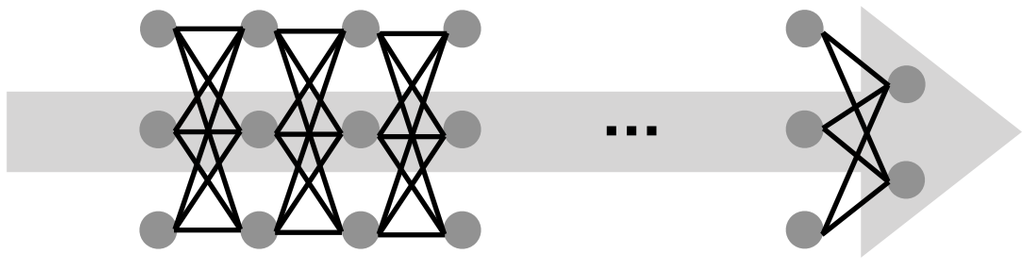
What is deep learning?



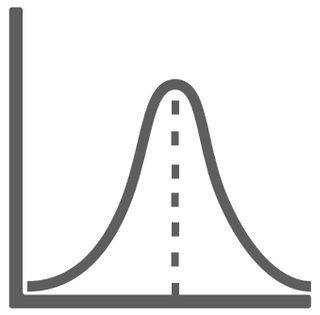
E.g. classification

Classification Model

How does it work?



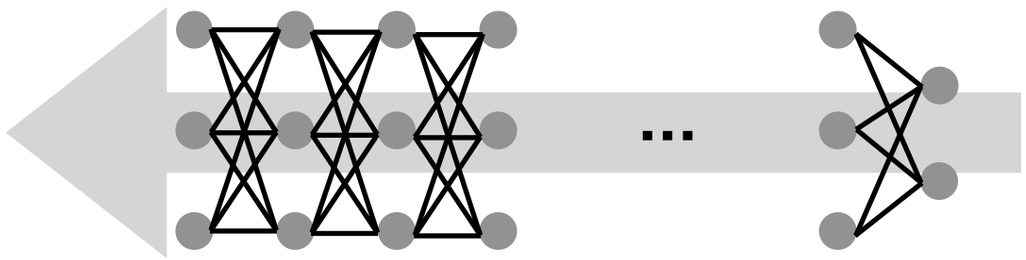
Forward path



Penguin: 0.01

Iteration

Difference measurement (Loss)

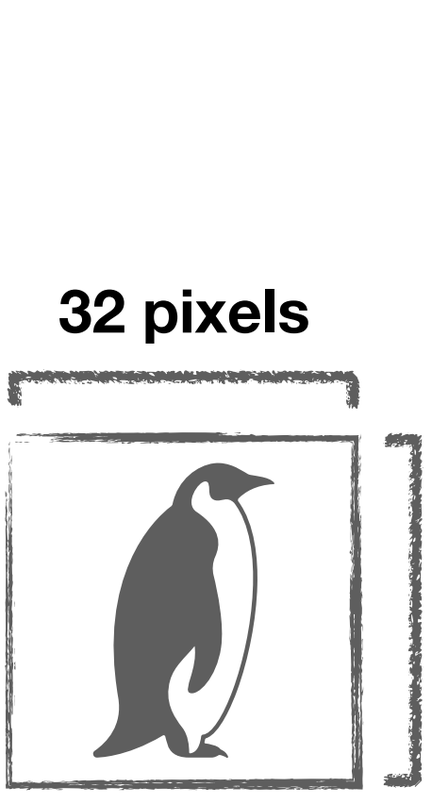


$\nabla_\theta L$

Penguin: 1.0

Backward path
(Gradient Back-propagation)

Forward path



Flatten

Bias

1

X_1

X_2

X_{1024}

$W_{0,0}$

$W_{1,0}$

$W_{2,0}$

$W_{1024,0}$

Y_0 Z_0

Y_1 Z_1

Y_2 Z_2

Y_m Z_m

$Y_0^{(L)}$ $Z_0^{(L)}$

$Y_1^{(L)}$ $Z_1^{(L)}$

$Y_2^{(L)}$ $Z_2^{(L)}$

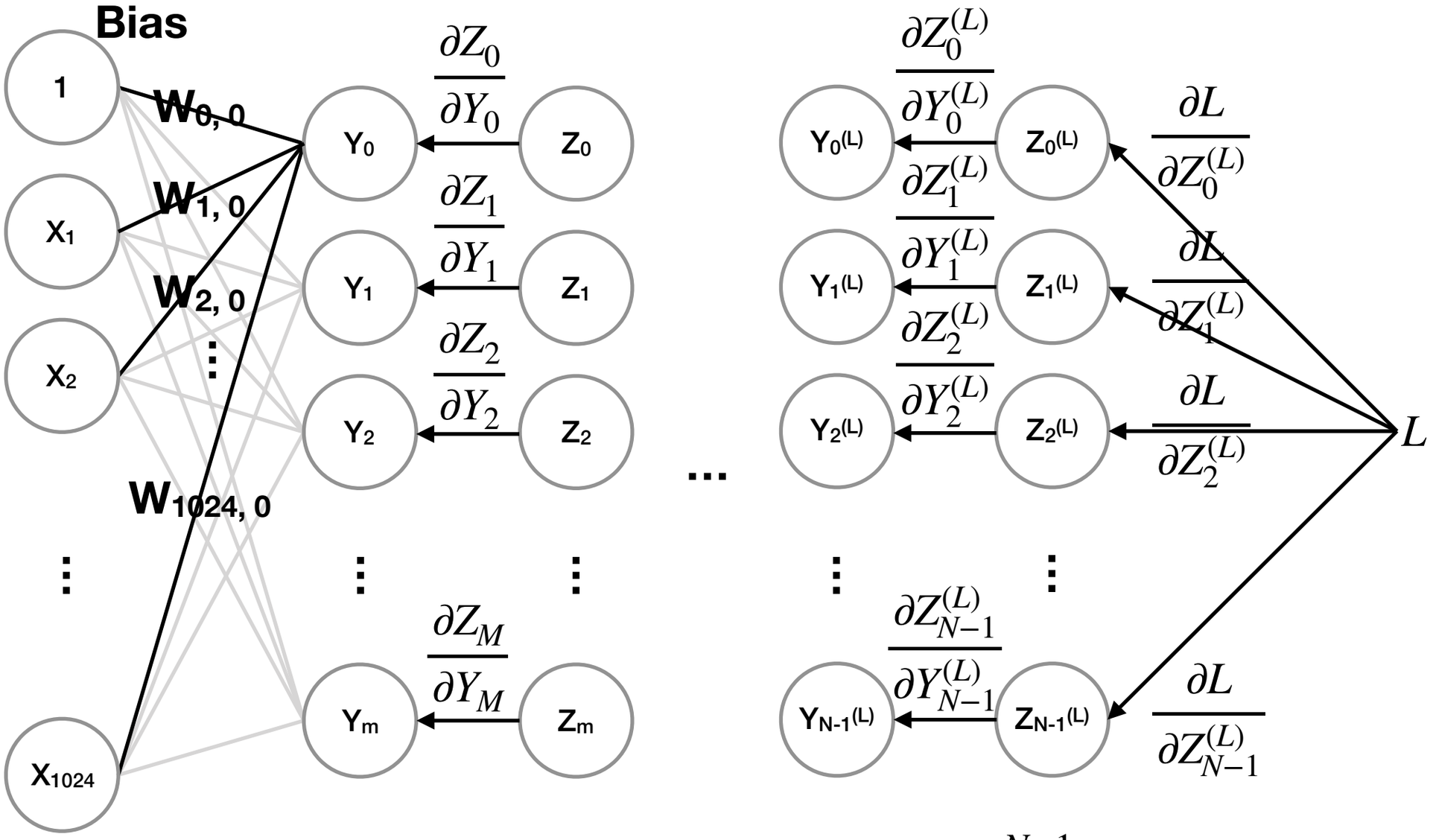
$Y_{N-1}^{(L)}$ $Z_{N-1}^{(L)}$

$$Y = WX \qquad Z = \sigma(Y) \qquad Z^{(L)} = softmax(Y^{(L)})$$

Input layer | 1st Hidden layer | 1st Activation layer | Lth Hidden layer | Output layer

Densely connected (fully connected)

Backward path



$$\frac{\partial L}{\partial W0,\ 0} = \frac{\partial L}{\partial Y_0}\frac{\partial Y_0}{W_{0,\ 0}}$$

$$W_{0,\ 0} = W_{0,\ 0} - \rho\frac{\partial L}{\partial W_{0,\ 0}}$$

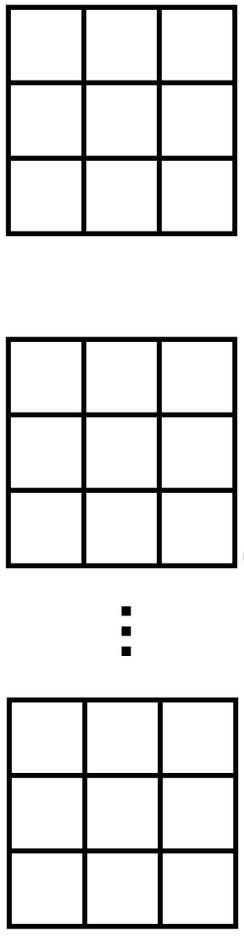
$$L = -\sum_{k=0}^{N-1} t_k log Z_k = -\ log Z_i$$

Cross-entropy loss

Convolution*

ICxHxW
1x32x32

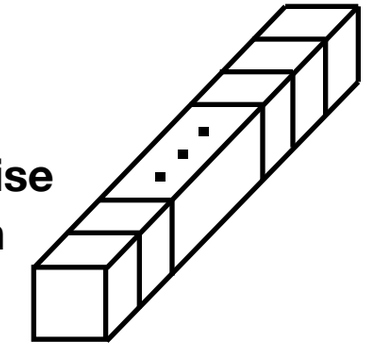
⊛
Convolution



OCxICxKHxKW
Nx1x3x3

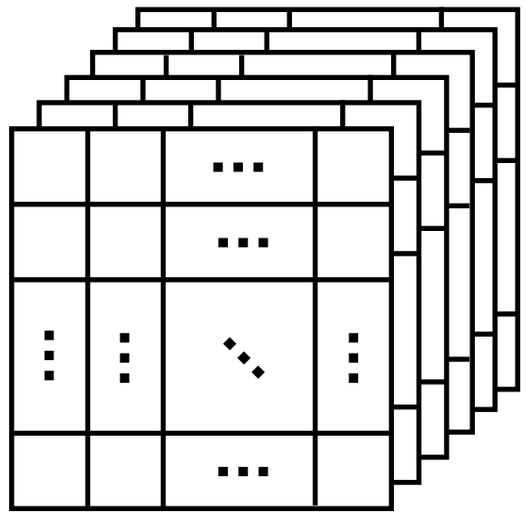
⊕
Channelwise Addition

Bias



OCx1
Nx1

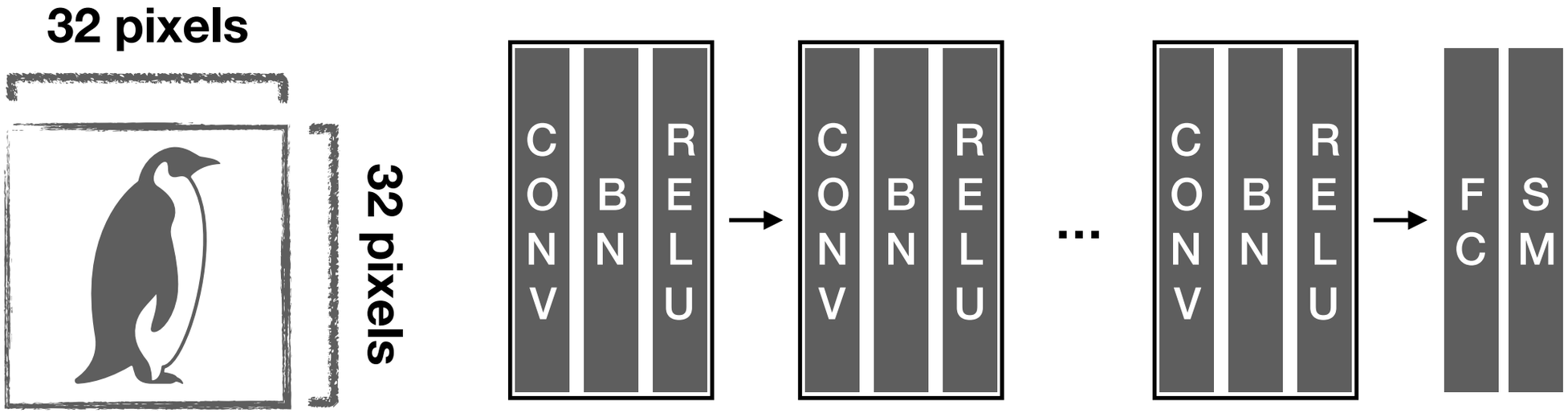
=



OCxHxW
Nx29x29
(No padding)

H: Height
W: Width
OC: Output Channel
IC: Input Channel
KH: Kernel Height
KW: Kernel Width

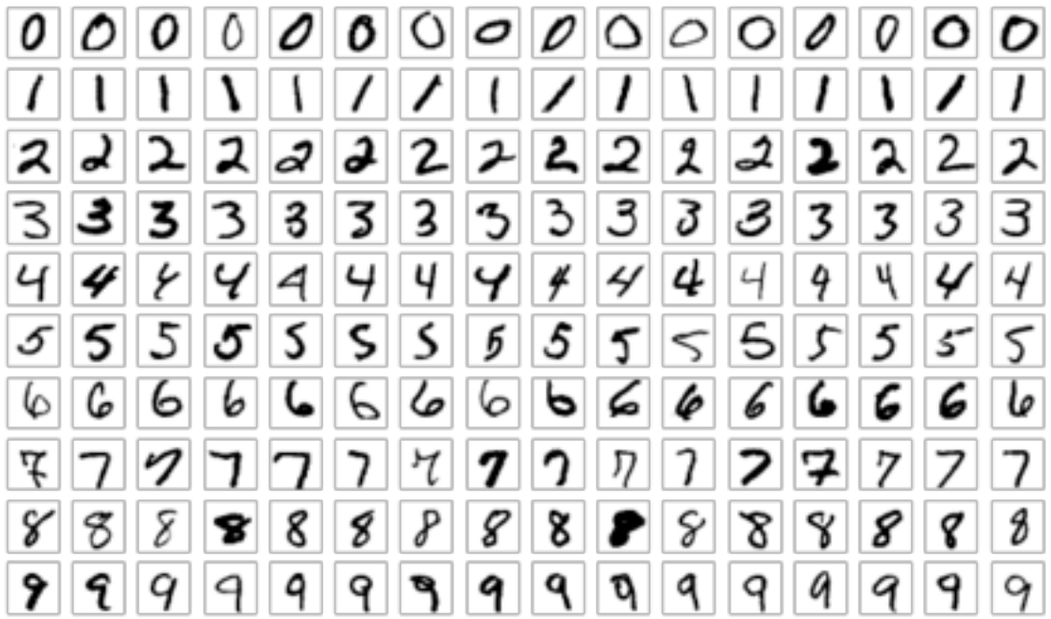
Convolutional neural network



CONV: Convolution
BN: Batch Normalization
RELU: REctified Linear Unit
FC: Fully Connected layer
SM: SoftMax activation layer

Practice

MNIST database



Training set: 60,000 images and labels
Test set: 10,000 images and labels

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems.

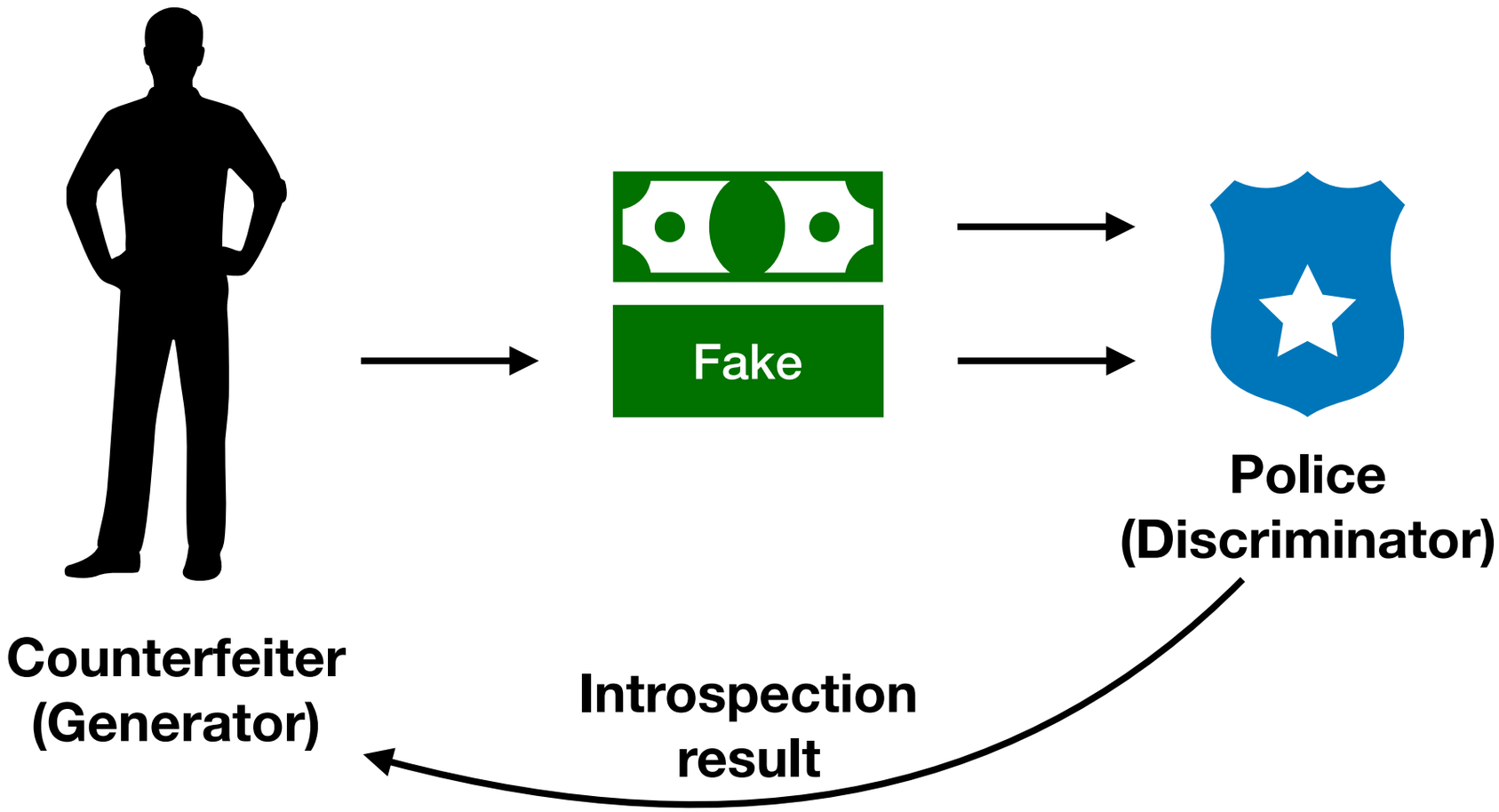
Generative Model

Various generative models

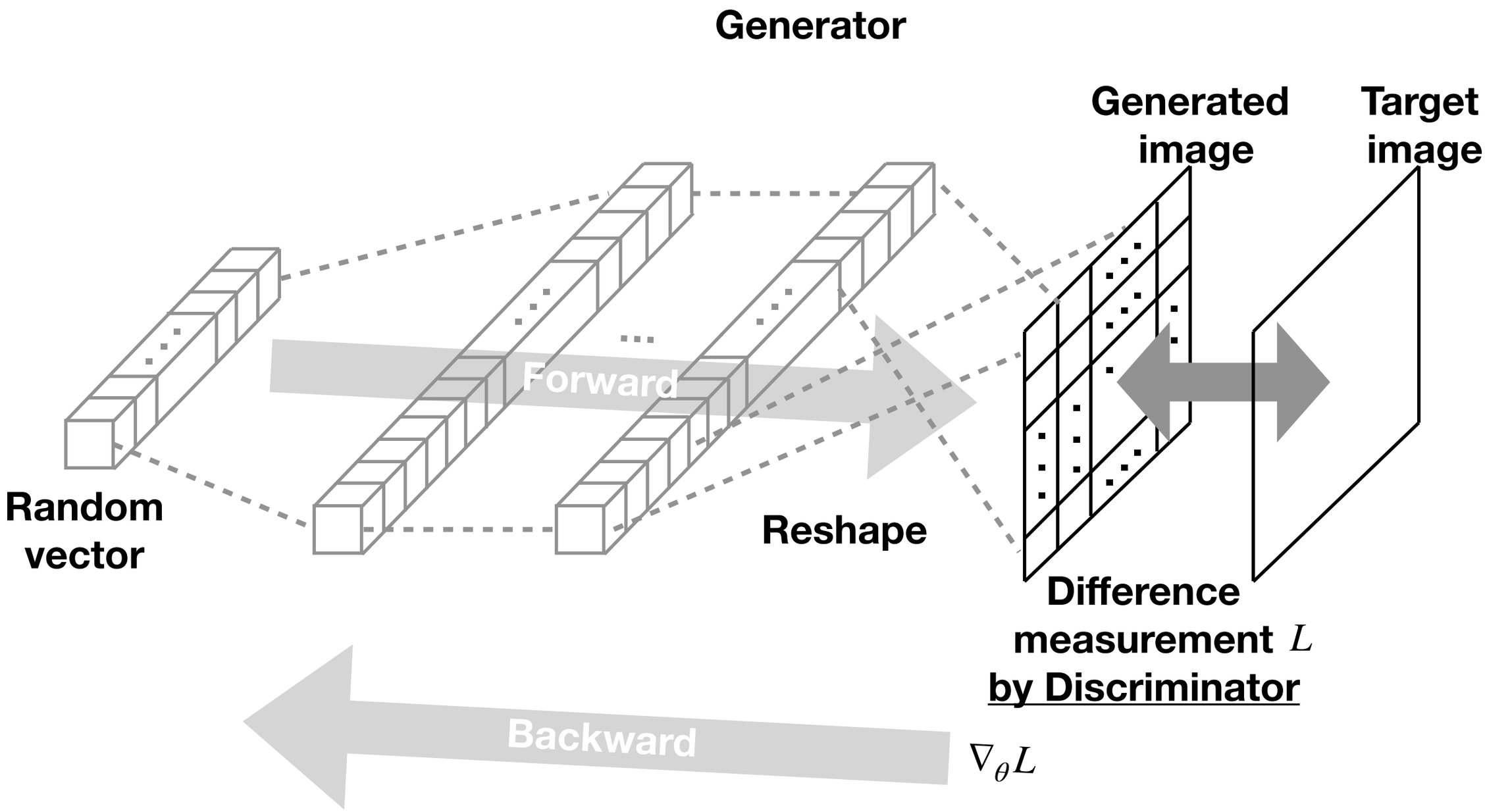
- Hidden Markov Model (HMM)
- Restricted Boltzmann Machine (RBM)
- Variational Auto-Encoder (VAE)
- Recurrent Neural Network (RNN)
- **Generative Adversarial Network (GAN)**

GAN

What is GAN?

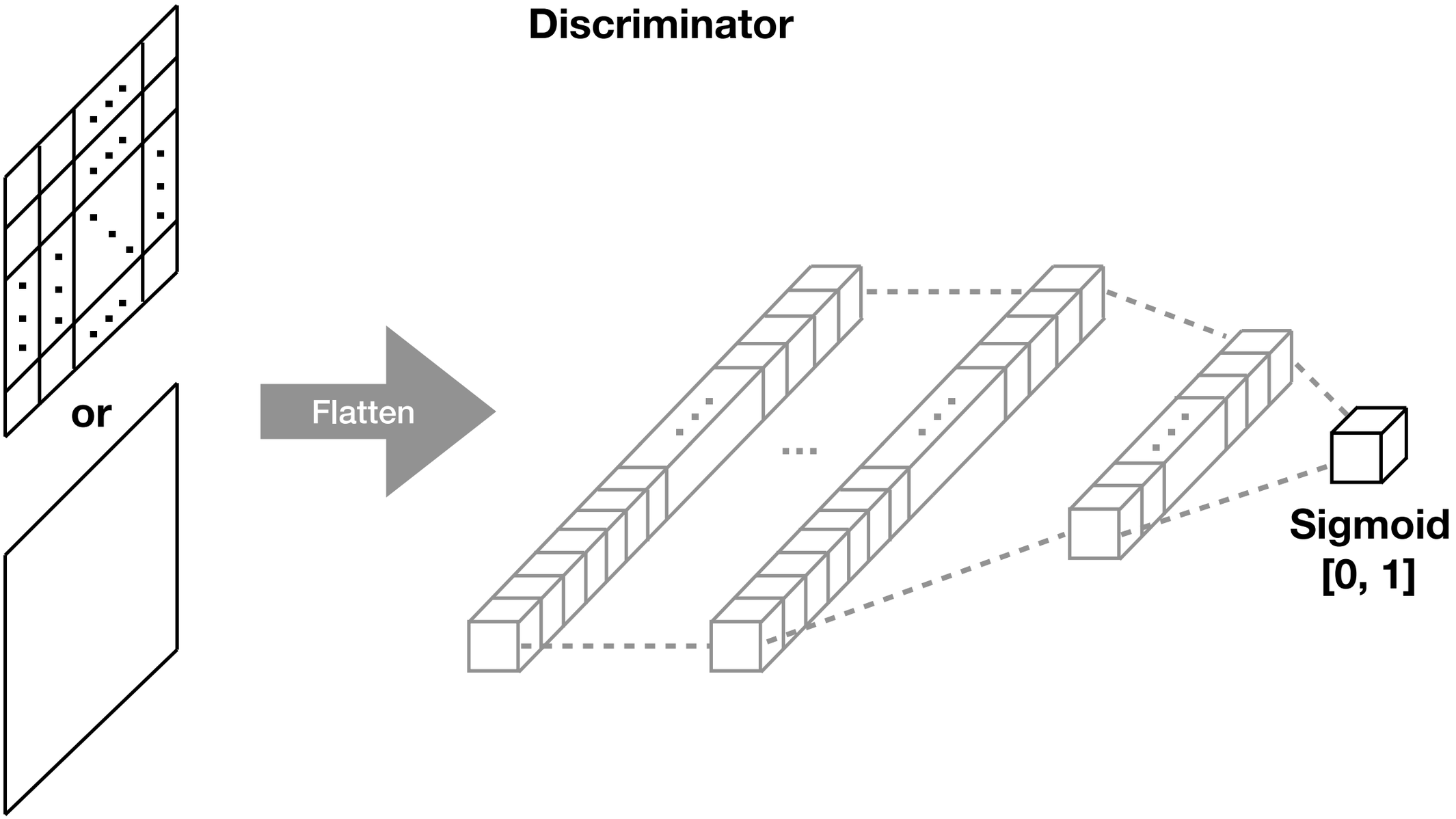

Fake
Police
(Discriminator)
Counterfeiter
(Generator)
Introspection
result

How does GAN work?



E.g. image generation model

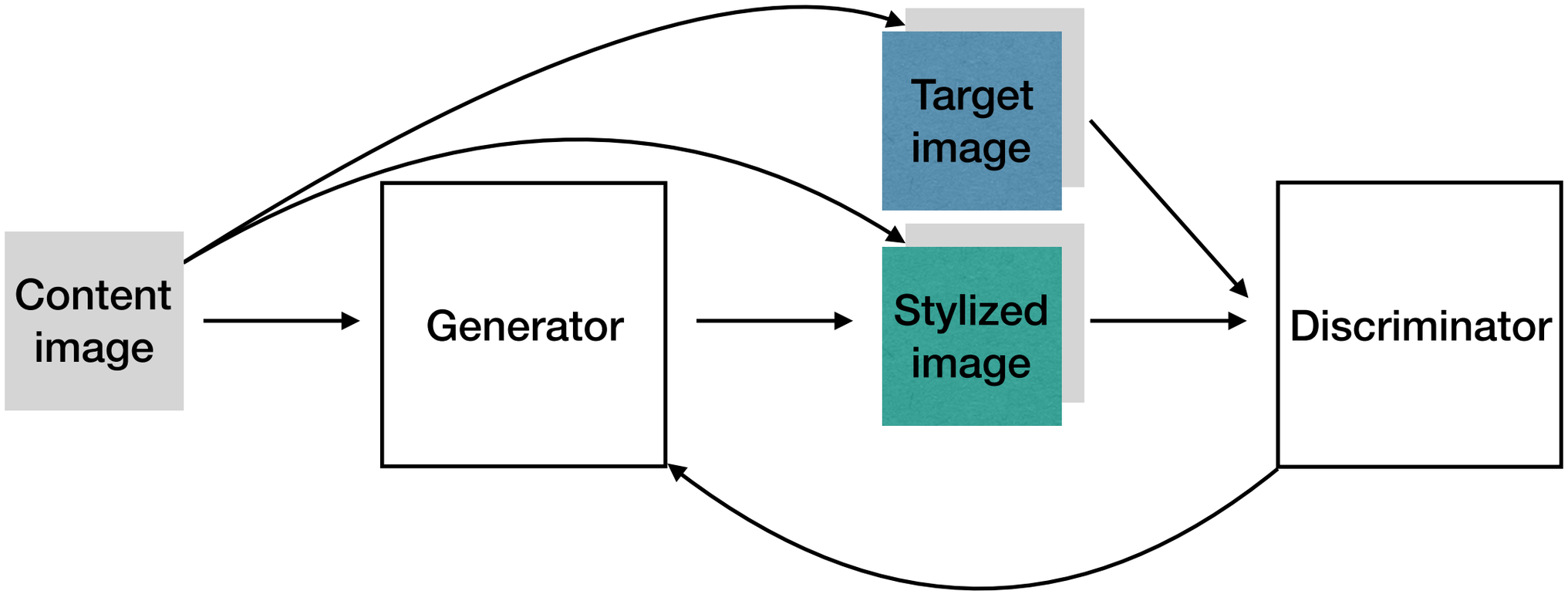
How does GAN work?



Practice

Conditional GAN

What is cGAN?

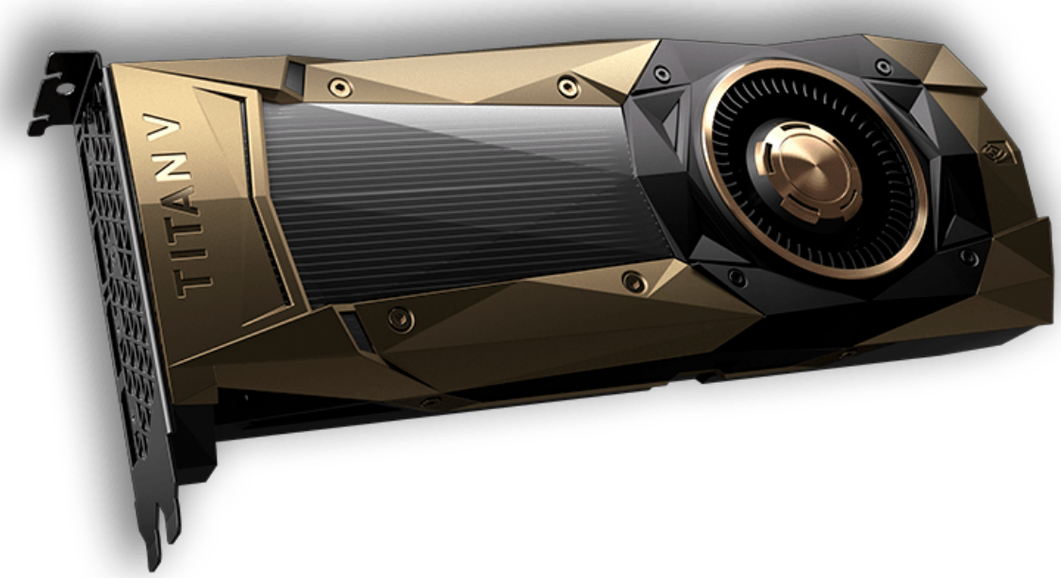

Target image
Content image
Generator
Stylized image
Discriminator

Practice

Appendix

CPU vs. GPU

credit. hothardware.com

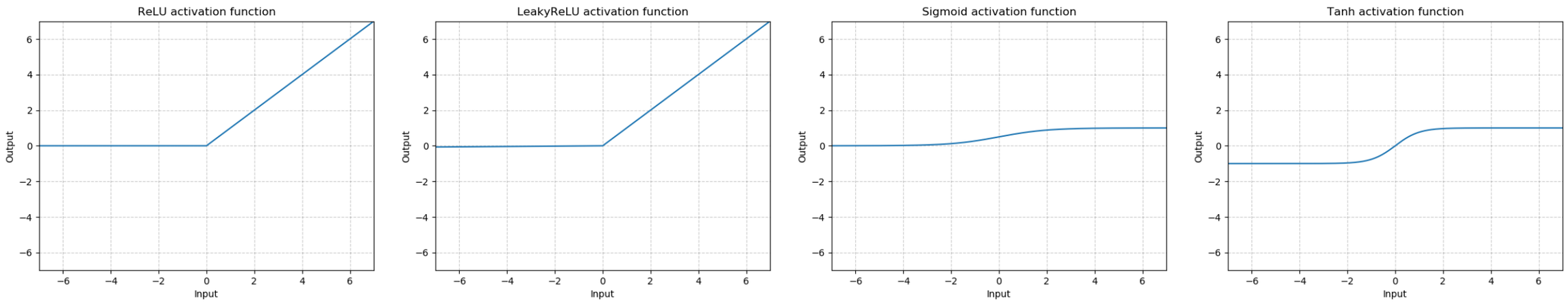


credit. NVIDIA

A central processing unit (CPU), also called a central processor or main processor, is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logic, controlling, and input/output operations specified by the instructions.

A graphics processing unit (GPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device.

Activation functions

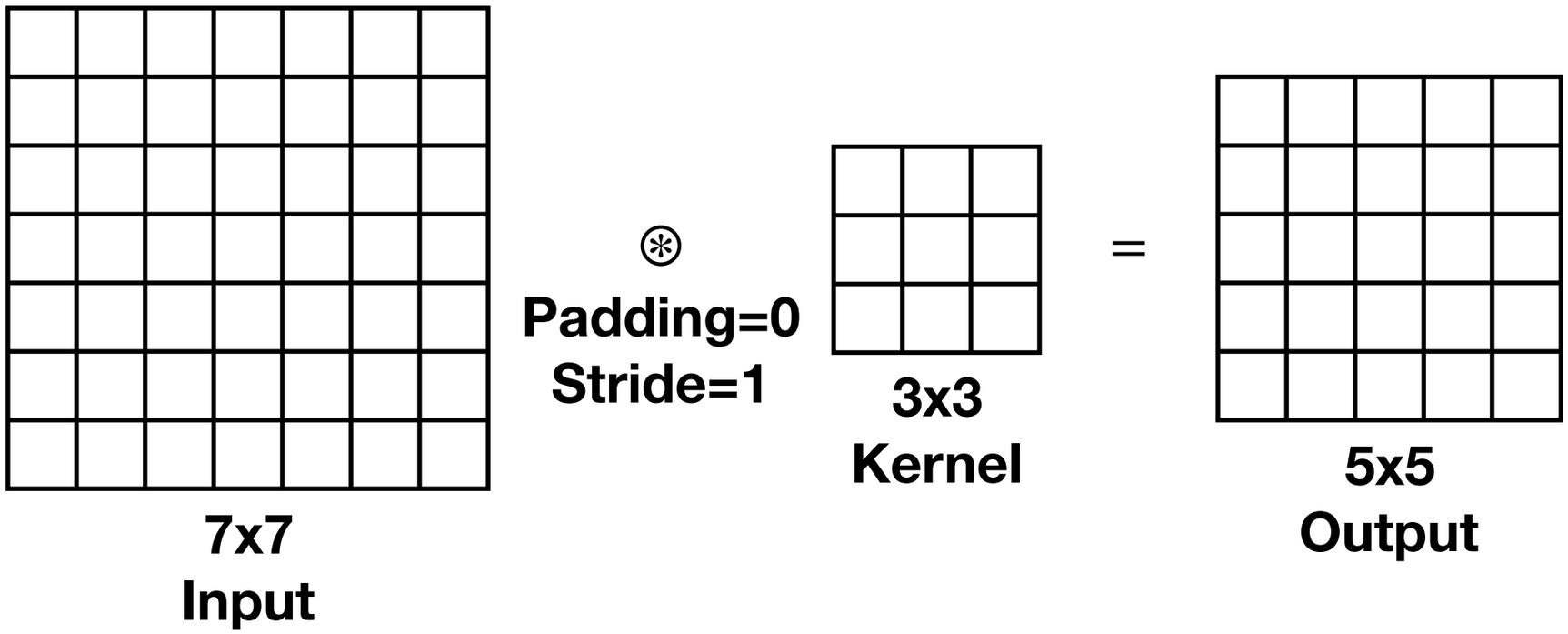


- ReLU (Rectified linear unit) : $max(0,\ x)$
- Leaky ReLU : $max(0,\ x)\ +\ negative\ slope \times min(0,\ x)$
- Hyperbolic tangent (tanh): $\frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Logistic sigmoid : $\frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} = \frac{1}{2} + \frac{1}{2}tanh(\frac{x}{2})$

Loss functions

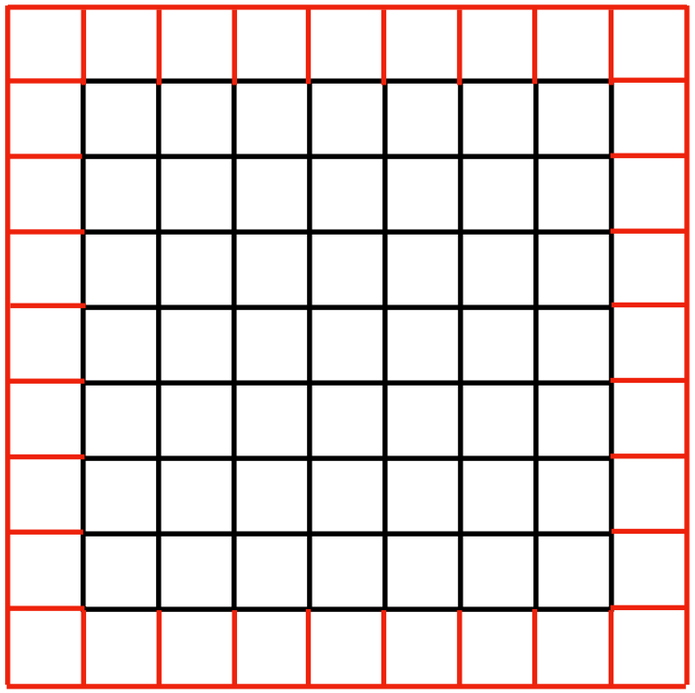
- Binary cross entropy loss (BCE) : $-\log \hat{p}(y_c \mid x) - \sum_{k=1,\ k \neq c}^{C} \log[1 - \hat{p}(y_k \mid x)]$
- Cross entropy loss (CE) : $\mathbb{E}_{p(y|x)}[-\log \hat{p}(y \mid x)] = -\sum_{c=1}^{C} p(y_c \mid x) \log \hat{p}(y_c \mid x) = -\log \hat{p}(y_c \mid x)$
- Mean squared error loss (MSE) : $\frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (x_{i,\ j} - \hat{x}_{i,\ j})^2$
- Mean absolute error loss (MAE) : $\frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |x_{i,\ j} - \hat{x}_{i,\ j}|$

Convolution



$$o = i - k + 1$$

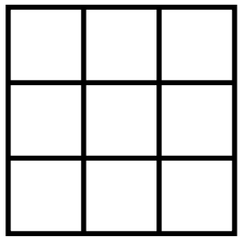
Convolution



(7 + 2 x p)x(7 + 2 x p)
Input

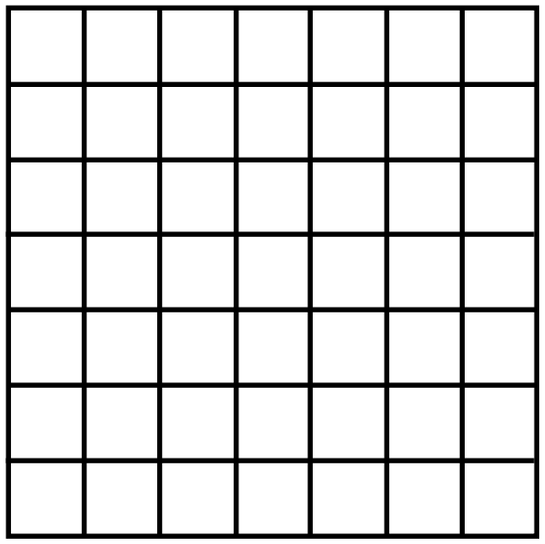
⊛

Padding=1
Stride=1



3x3
Kernel

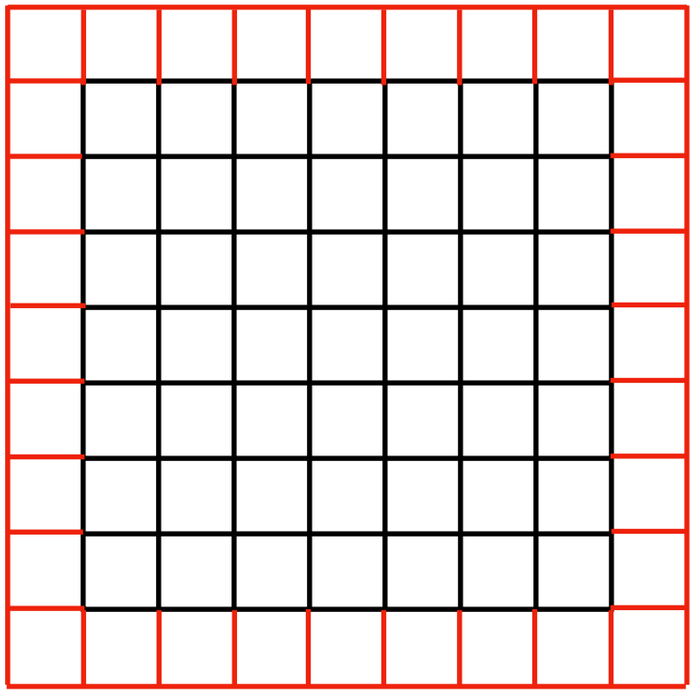
=



7x7
Output

$$o = i - k + 2p + 1$$

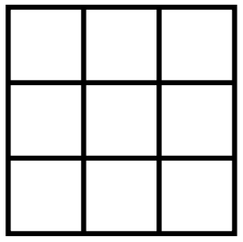
Convolution



(7 + 2 x p)x(7 + 2 x p)
Input

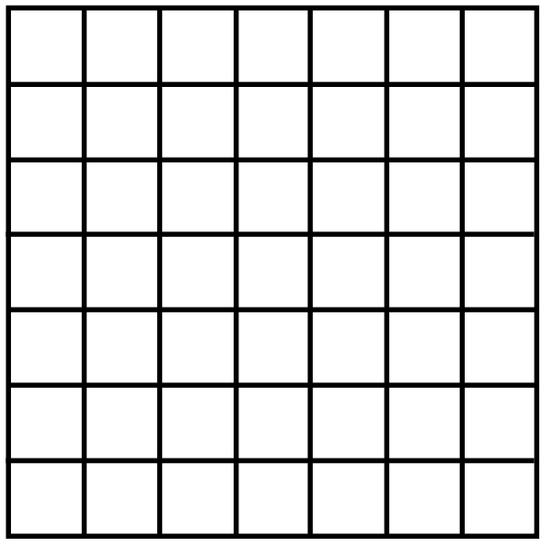
⊛

Padding=1
Stride=2



3x3
Kernel

=



4x4
Output

$$o = \lfloor \frac{i - k + 2p}{s} \rfloor + 1$$