DATE : 11.11.2024

DT/NT : DT

LESSON : Deep Learning

SUBJECT : Optical Objects Recognition (OCR)

BATCH : 250

Data Science

TECHPRO EDUCATION

techproeducation.com

+1 (585) 304 29 59

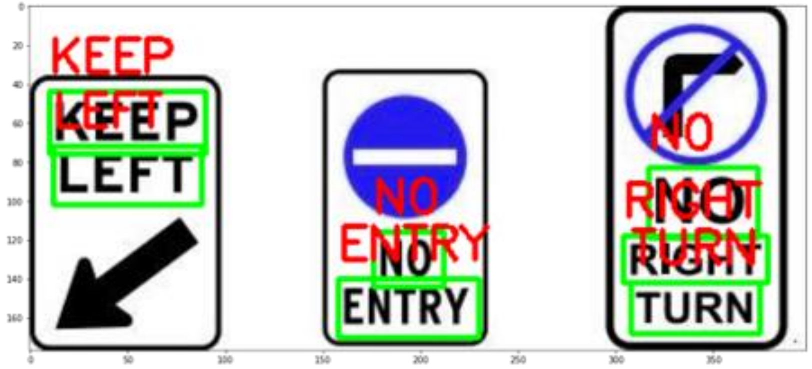
OPTICAL CHARACTER RECOGNITION

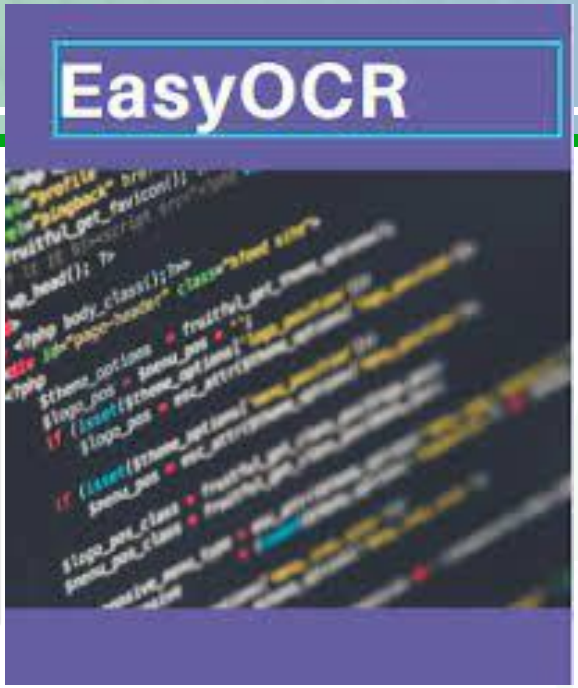
OCR

michele 1025 W. LINE
AVENUE APT . 2B
CHICAGO, IL 60610

Text Recognition

OCR





Tesseract OCR

OCR


```
In [40]: plt.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))

Out[40]: <matplotlib.image.AxesImage at 0x19b90513748>
```

4. Use Easy OCR To Read Text

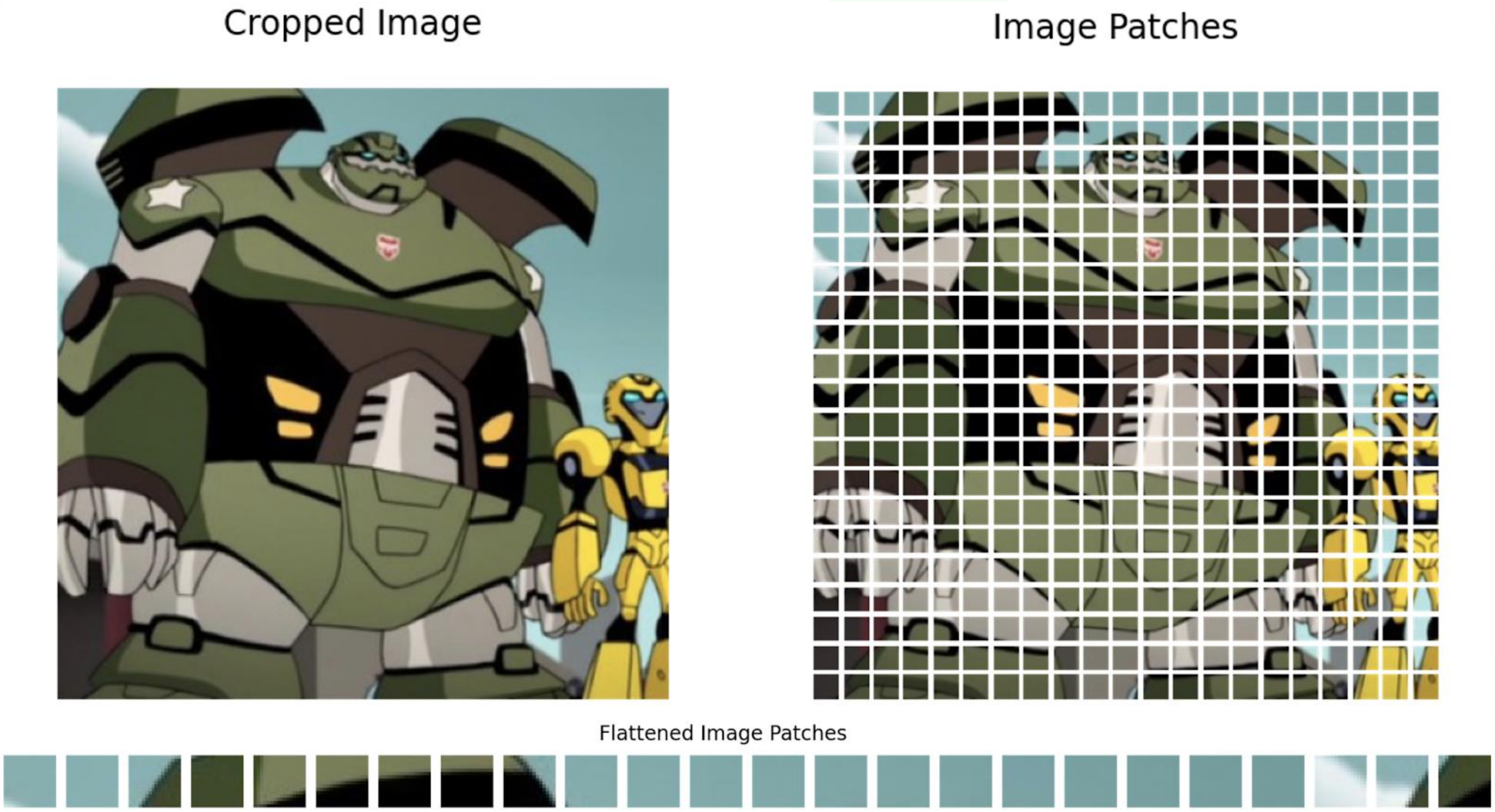
```
In [44]: reader = easyocr.Reader(['en'])
         result = reader.readtext(cropped_image)
         result

Out[44]: [([[0, 0], [100, 0], [100, 29], [0, 29]], ':COVID19', 0.47240057587623596)]
```

```
text = pytesseract.image_to_string(cropped,lang="eng")
print("detected text: ",text)
```

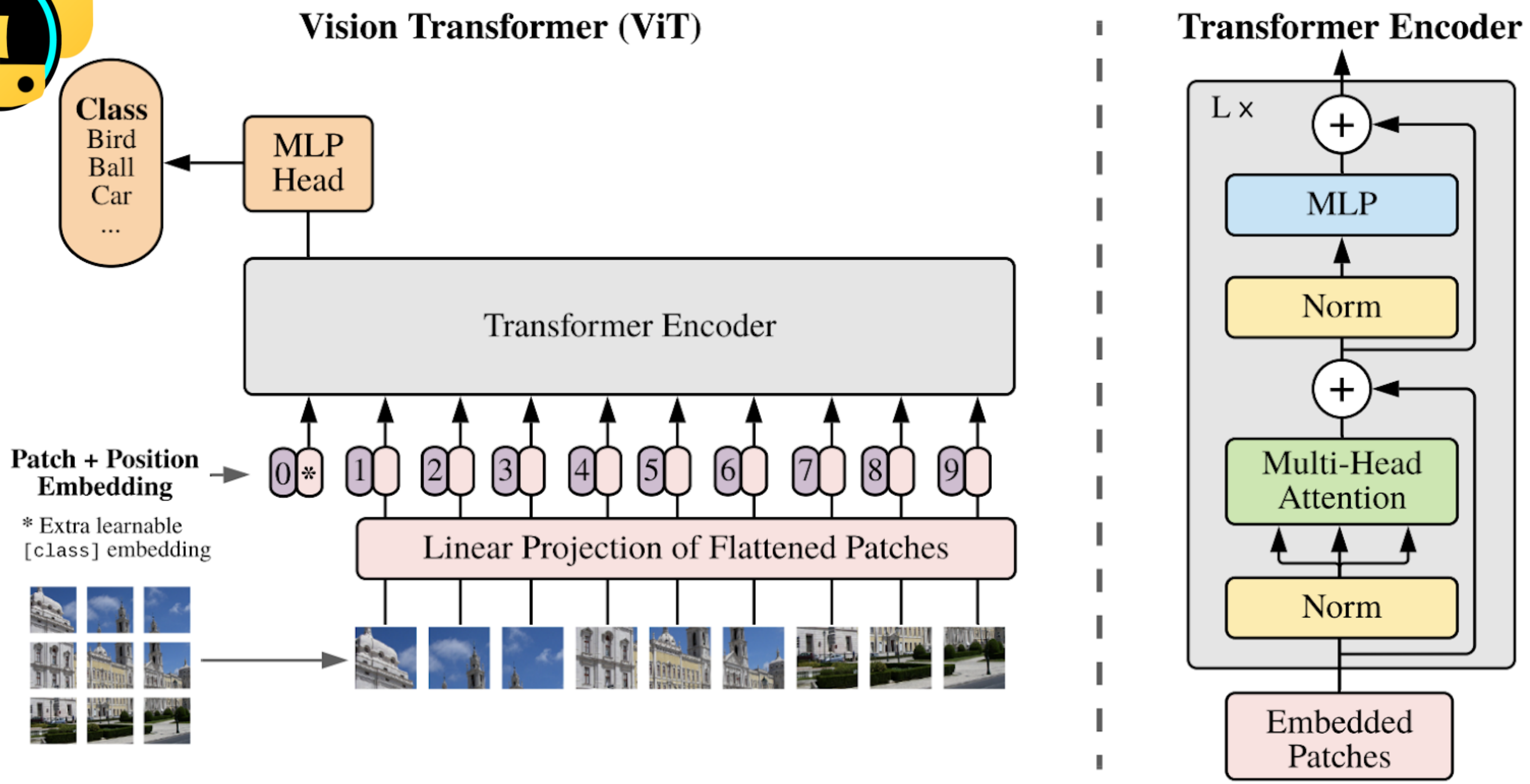
VISION TRANSFORMER (ViT)

ViT



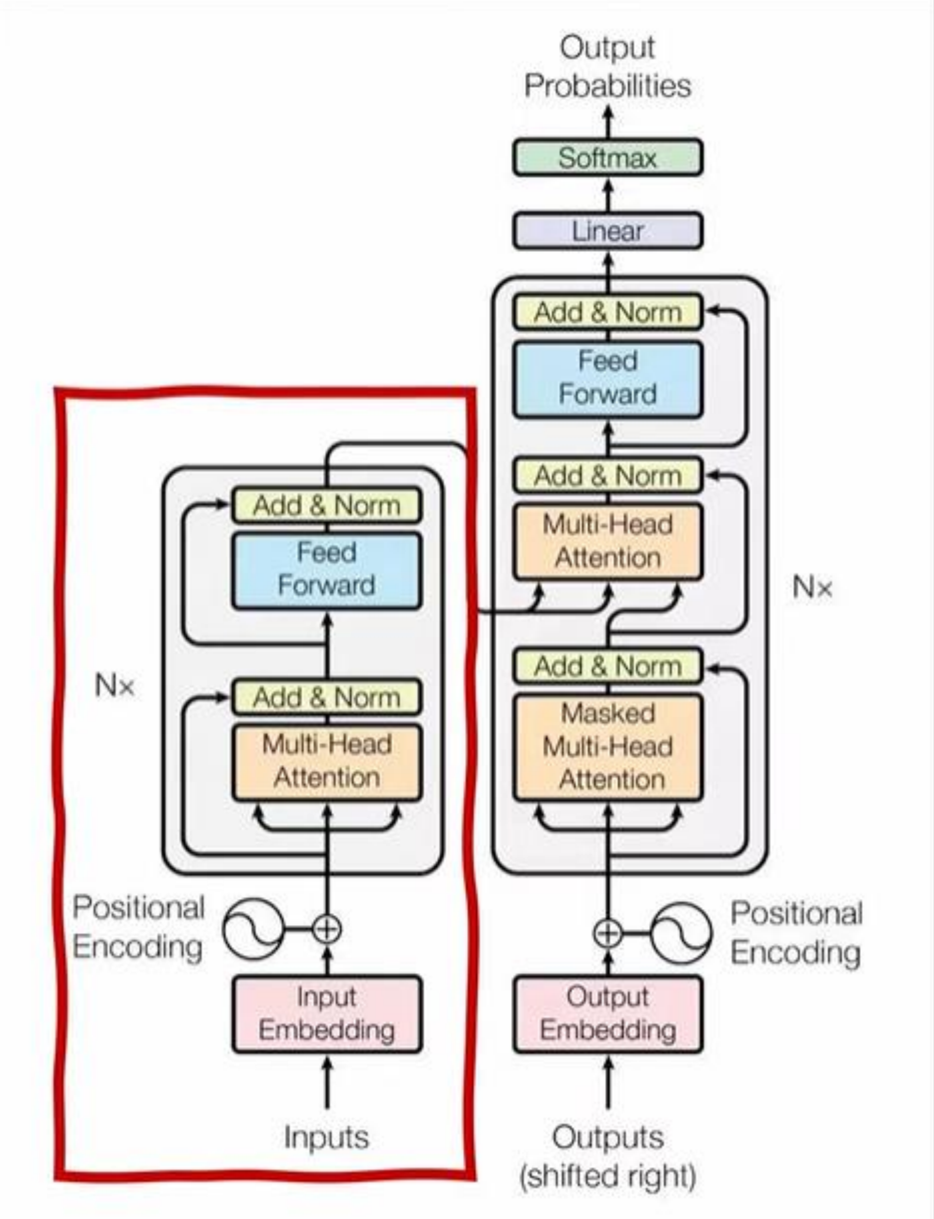
ViT

ViT



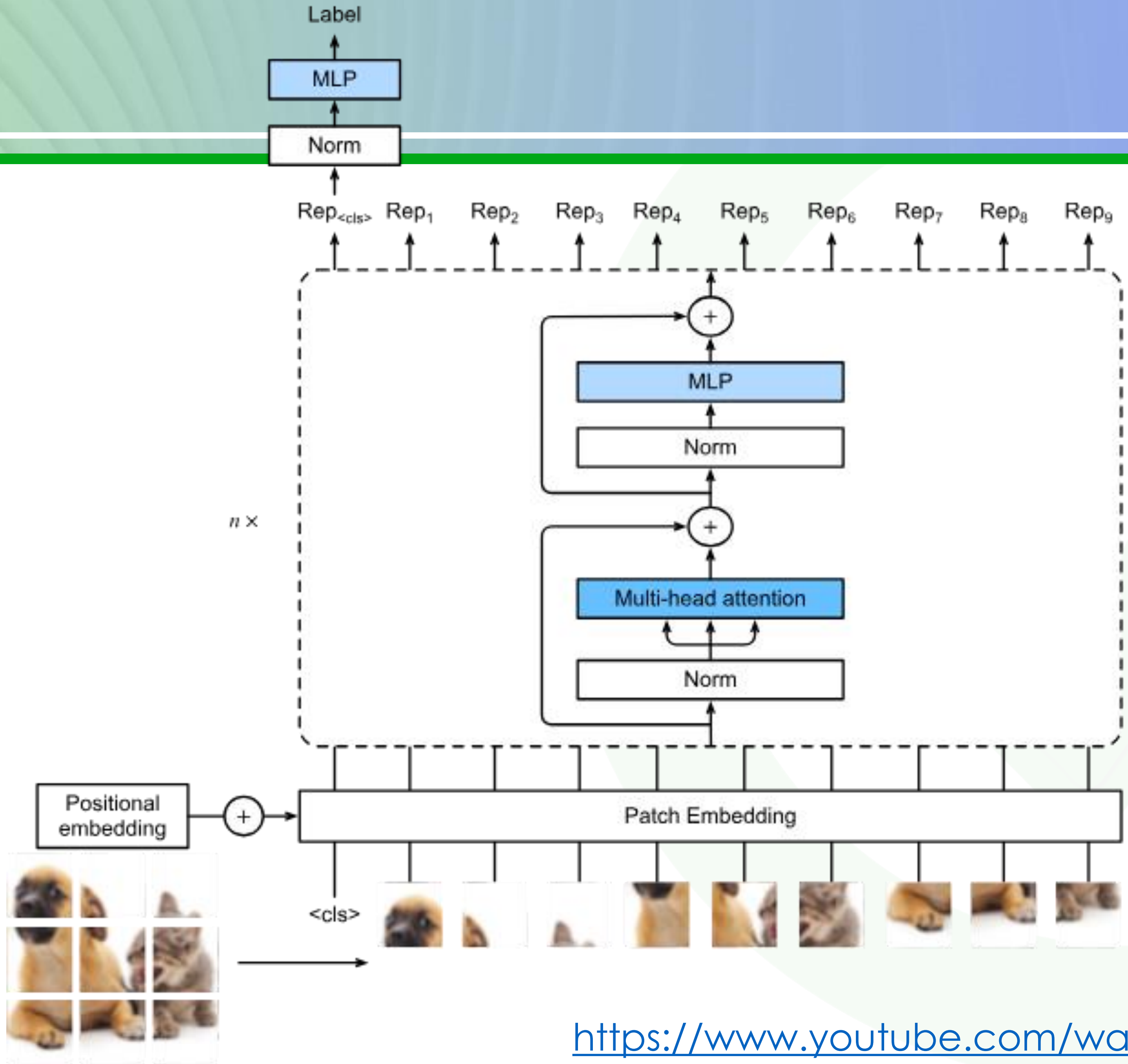
Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0
*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
Transformer Encoder
L x
MLP
Norm
Multi-Head Attention
Norm
Embedded Patches

ViT



Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs
Positional Encoding
Output Embedding
Outputs (shifted right)

ViT



https://www.youtube.com/watch?v=j3VNqtJUoz0

GENERATIVE ADVERSARIAL NETWORKS (GAN)

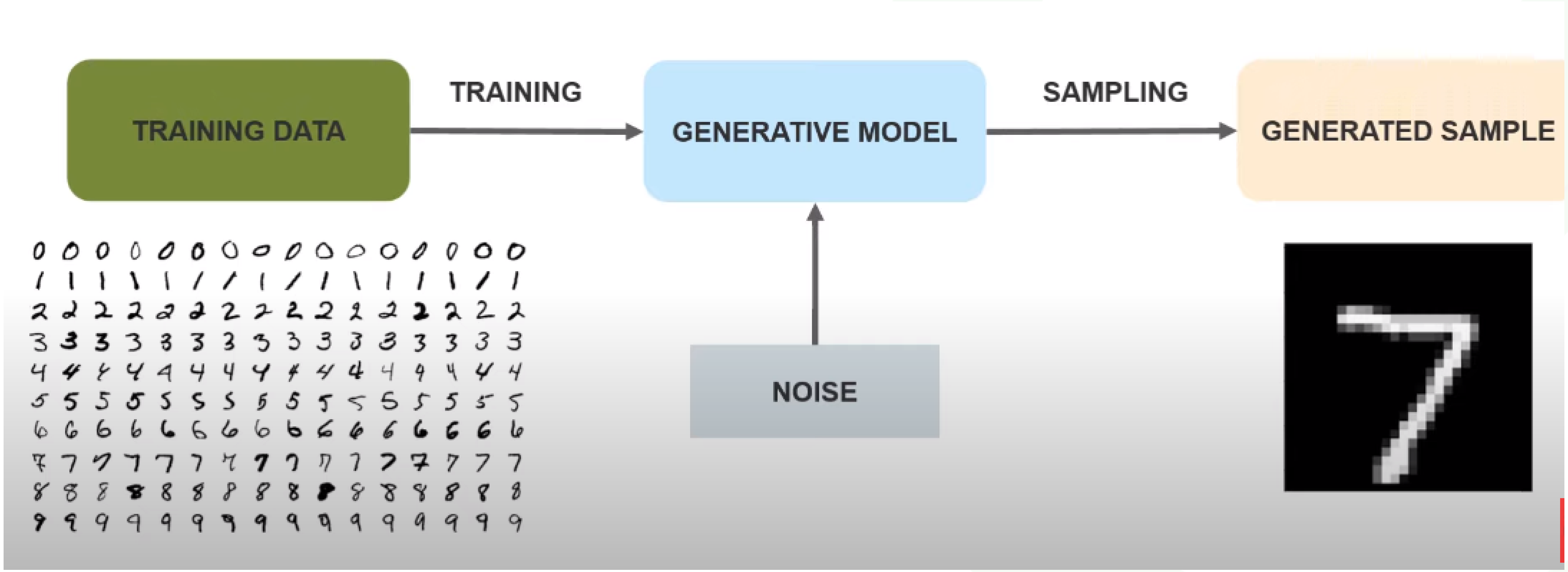
GAN

GAN

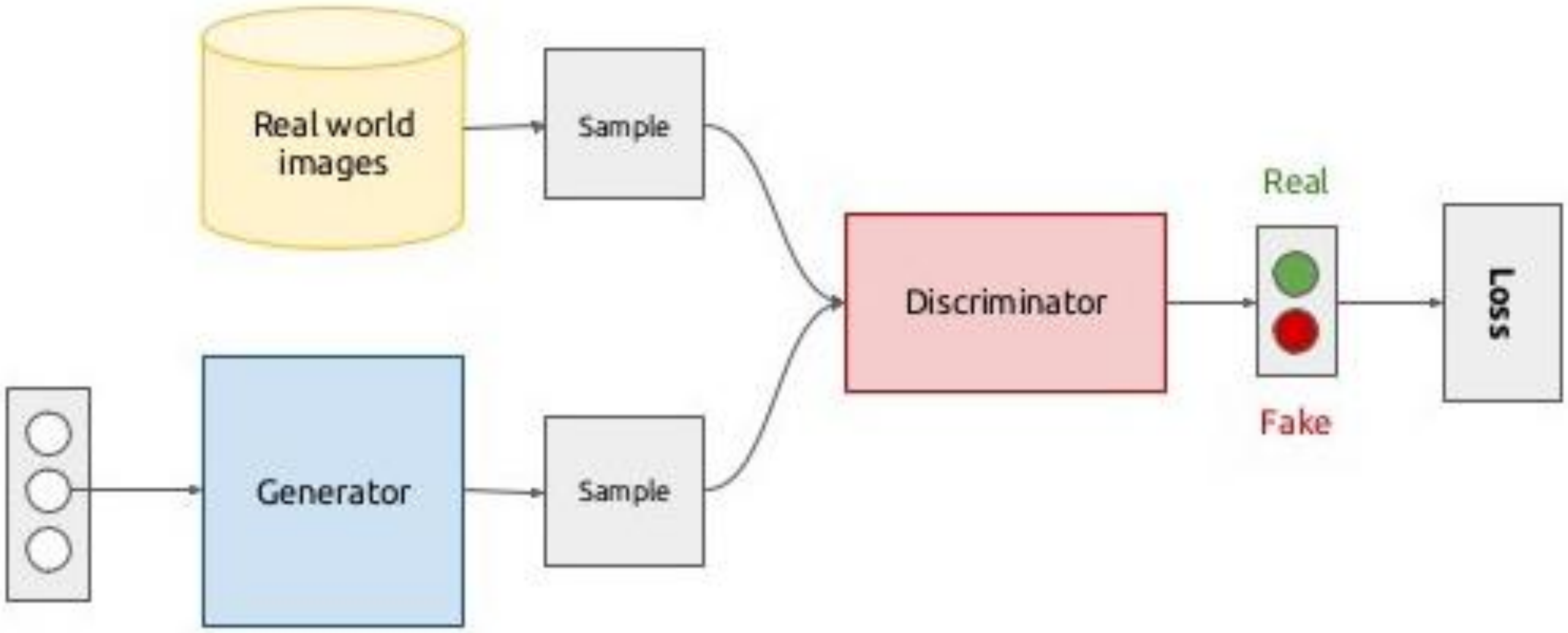

deeplearning.ai presents
Heroes of Deep Learning
Ian Goodfellow
Research Scientist at Google Brain

GAN

GAN

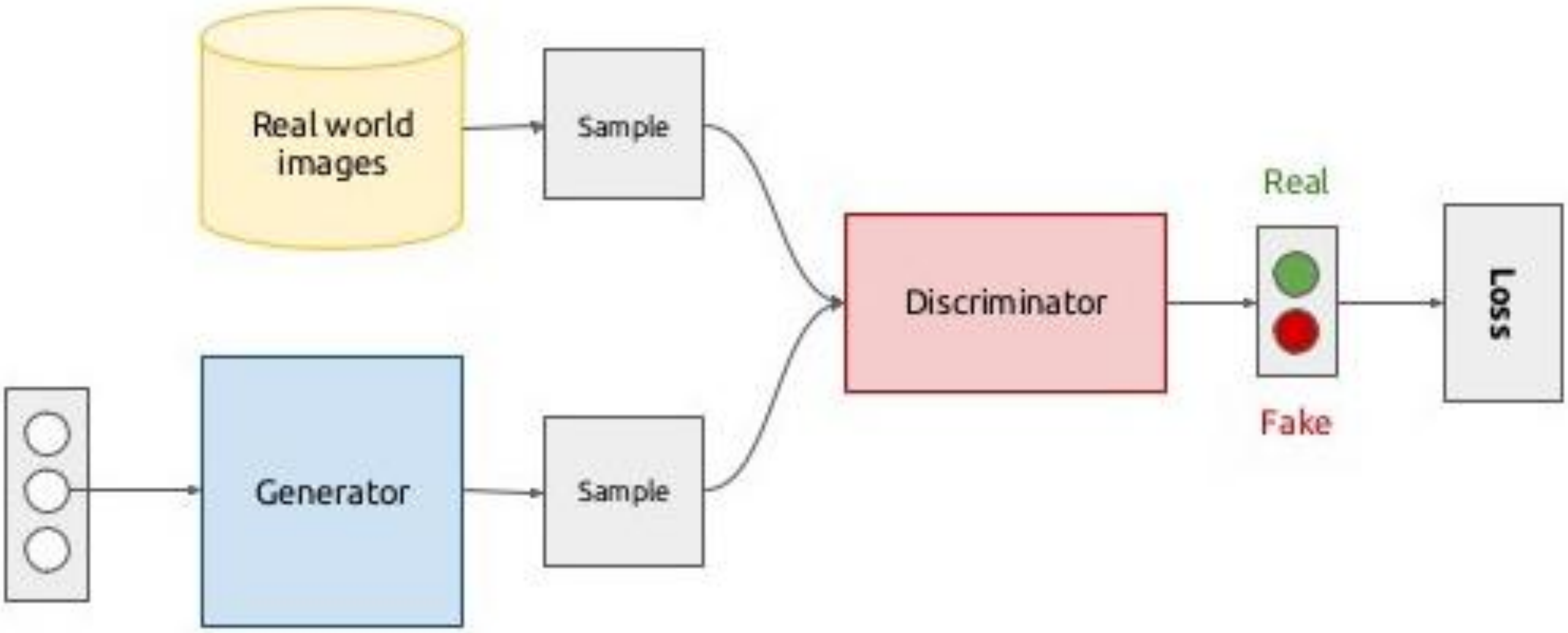

TRAINING DATA
TRAINING
GENERATIVE MODEL
SAMPLING
GENERATED SAMPLE
NOISE

GAN



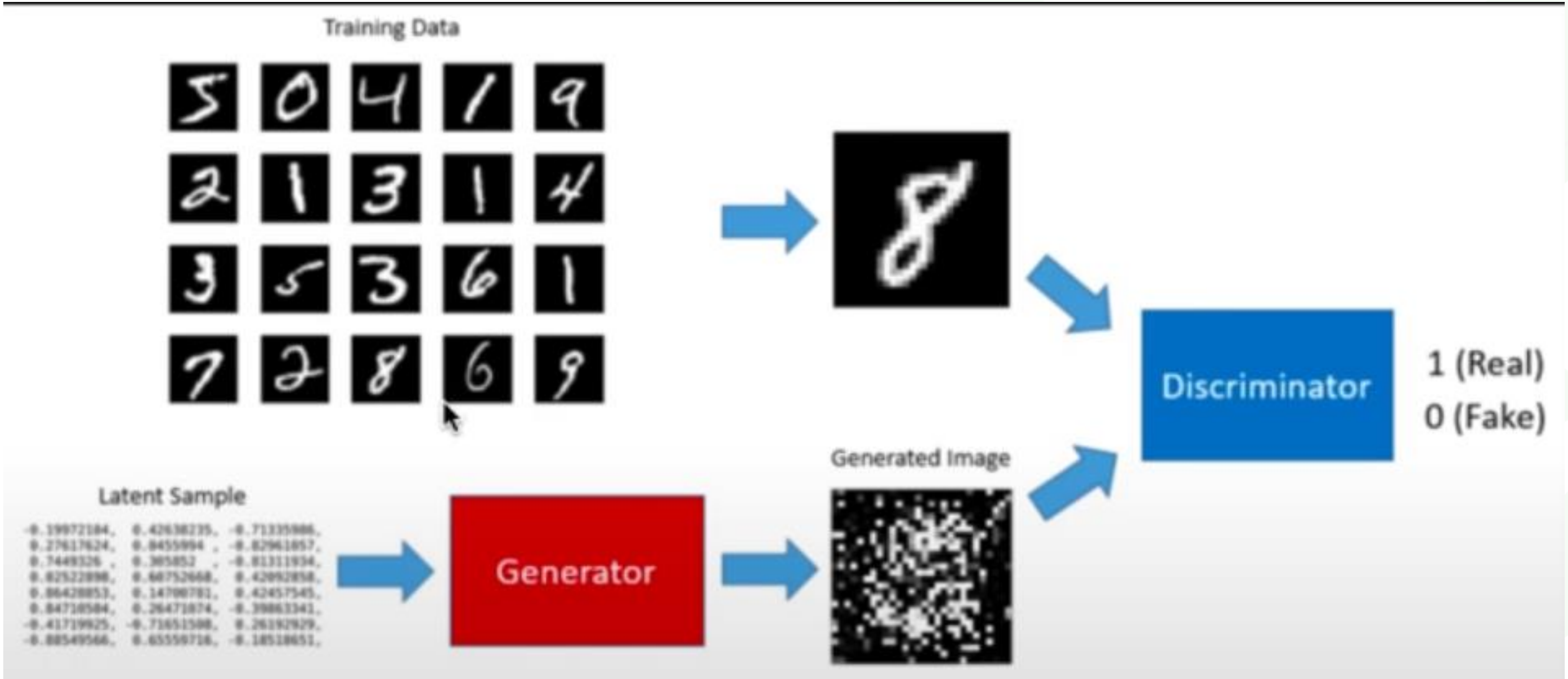
Real world images
Sample
Generator
Sample
Discriminator
Real
Fake
Loss

GAN



Real world images
Sample
Generator
Sample
Discriminator
Real
Fake
Loss

GAN


Training Data
Discriminator
1 (Real)
0 (Fake)
Generated Image
Latent Sample
Generator

GAN

```
class GAN():
    def __init__(self):
        self.img_rows = 28
        self.img_cols = 28
        self.channels = 1
        self.img_shape = (self.img_rows, self.img_cols, self.channels)
        self.latent_dim = 100

        optimizer = Adam(0.0002, 0.5)

        # Build and compile the discriminator
        self.discriminator = self.build_discriminator()
        self.discriminator.compile(loss='binary_crossentropy',
            optimizer=optimizer,
            metrics=['accuracy'])

        # Build the generator
        self.generator = self.build_generator()

        # The generator takes noise as input and generates imgs
        z = Input(shape=(self.latent_dim,))
        img = self.generator(z)

        # For the combined model we will only train the generator
        self.discriminator.trainable = False

        # The discriminator takes generated images as input and determines validity
        validity = self.discriminator(img)

        # The combined model  (stacked generator and discriminator)
        # Trains the generator to fool the discriminator
        self.combined = Model(z, validity)
        self.combined.compile(loss='binary_crossentropy', optimizer=optimizer)
```

GAN

```python
def build_generator(self):

    model = Sequential()

    model.add(Dense(256, input_dim=self.latent_dim))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(1024))
    model.add(LeakyReLU(alpha=0.2))
    model.add(BatchNormalization(momentum=0.8))
    model.add(Dense(np.prod(self.img_shape), activation='tanh'))
    model.add(Reshape(self.img_shape))

    model.summary()

    noise = Input(shape=(self.latent_dim,))
    img = model(noise)

    return Model(noise, img)
```

```python
def build_discriminator(self):

    model = Sequential()

    model.add(Flatten(input_shape=self.img_shape))
    model.add(Dense(512))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(256))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dense(1, activation='sigmoid'))
    model.summary()

    img = Input(shape=self.img_shape)
    validity = model(img)

    return Model(img, validity)
```

GAN

Generative Adversarial Text to Image Synthesis

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran REEDSCOT[1], AKATA[2], XCYAN[1], LLAJAN[1]
Bernt Schiele, Honglak Lee SCHIELE[2], HONGLAK[1]
[1] University of Michigan, Ann Arbor, MI, USA (UMICH.EDU)
[2] Max Planck Institute for Informatics, Saarbrücken, Germany (MPI-INF.MPG.DE)

Abstract

Automatic synthesis of realistic images from text would be interesting and useful, but current AI systems are still far from this goal. However, in recent years generic and powerful recurrent neural network architectures have been developed to learn discriminative text feature representations. Meanwhile, deep convolutional generative adversarial networks (GANs) have begun to generate highly compelling images of specific categories, such as faces, album covers, and room interiors. In this work, we develop a novel deep architecture and GAN formulation to effectively bridge these advances in text and image modeling, translating visual concepts from characters to pixels. We demonstrate the capability of our model to generate plausible images of birds and flowers from detailed text descriptions.

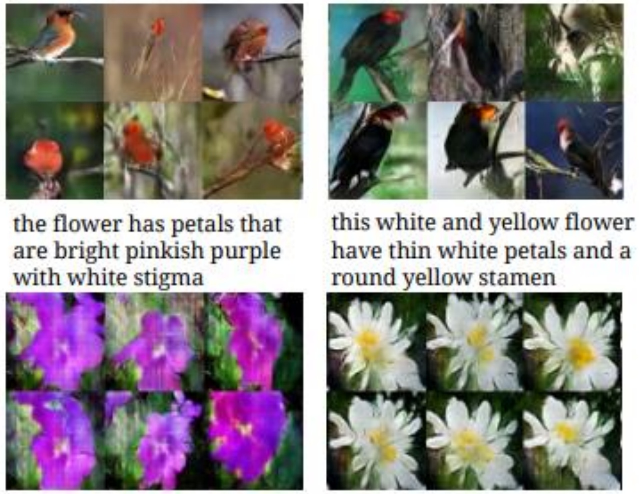


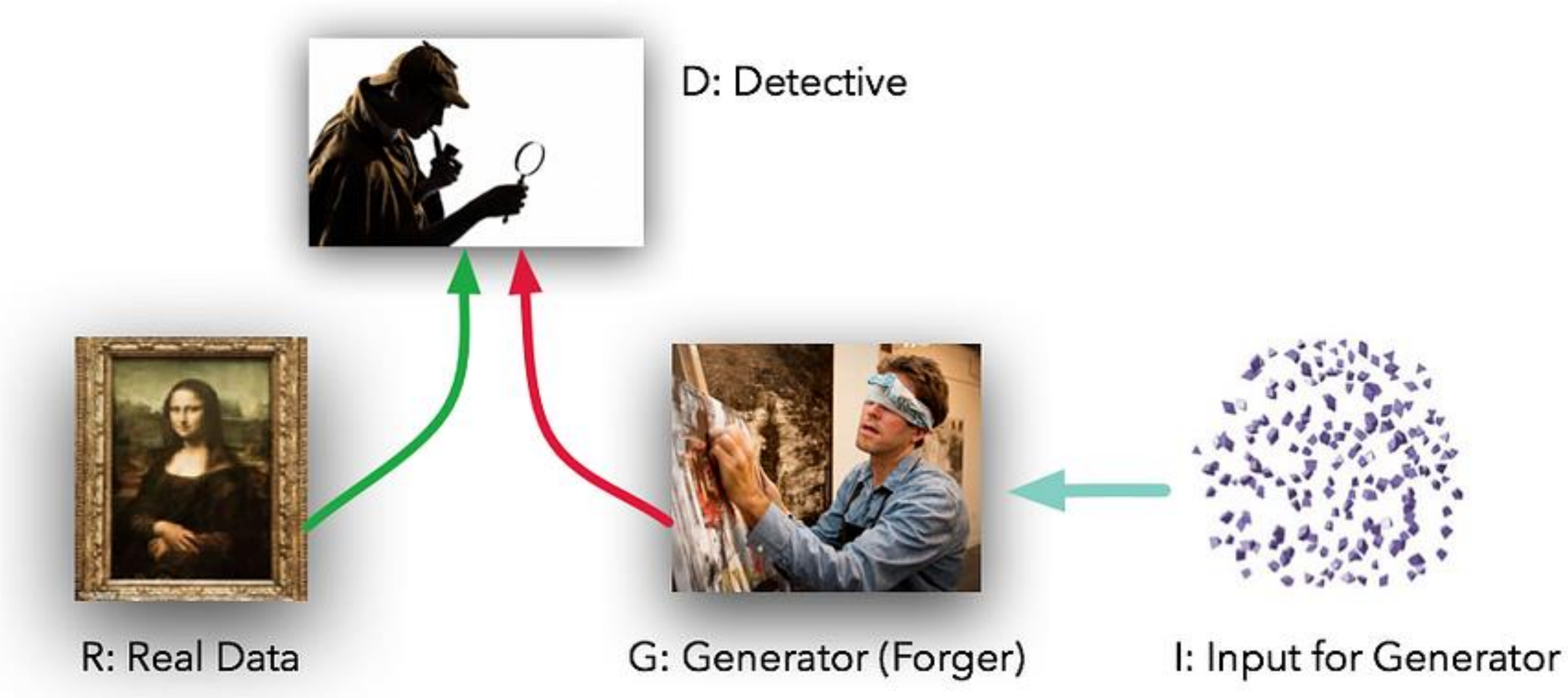
Figure 1. Examples of generated images from text descriptions. Left: captions are from zero-shot (held out) categories, unseen text. Right: captions are from the training set.

1. Introduction

In this work we are interested in translating text in the form of single-sentence human-written descriptions directly into image pixels. For example, "this small bird has a short, pointy orange beak and white belly" or "the petals of this flower are pink and the anther are yellow". The problem of generating images from visual descriptions gained interest in the research community, but it is far from being solved.

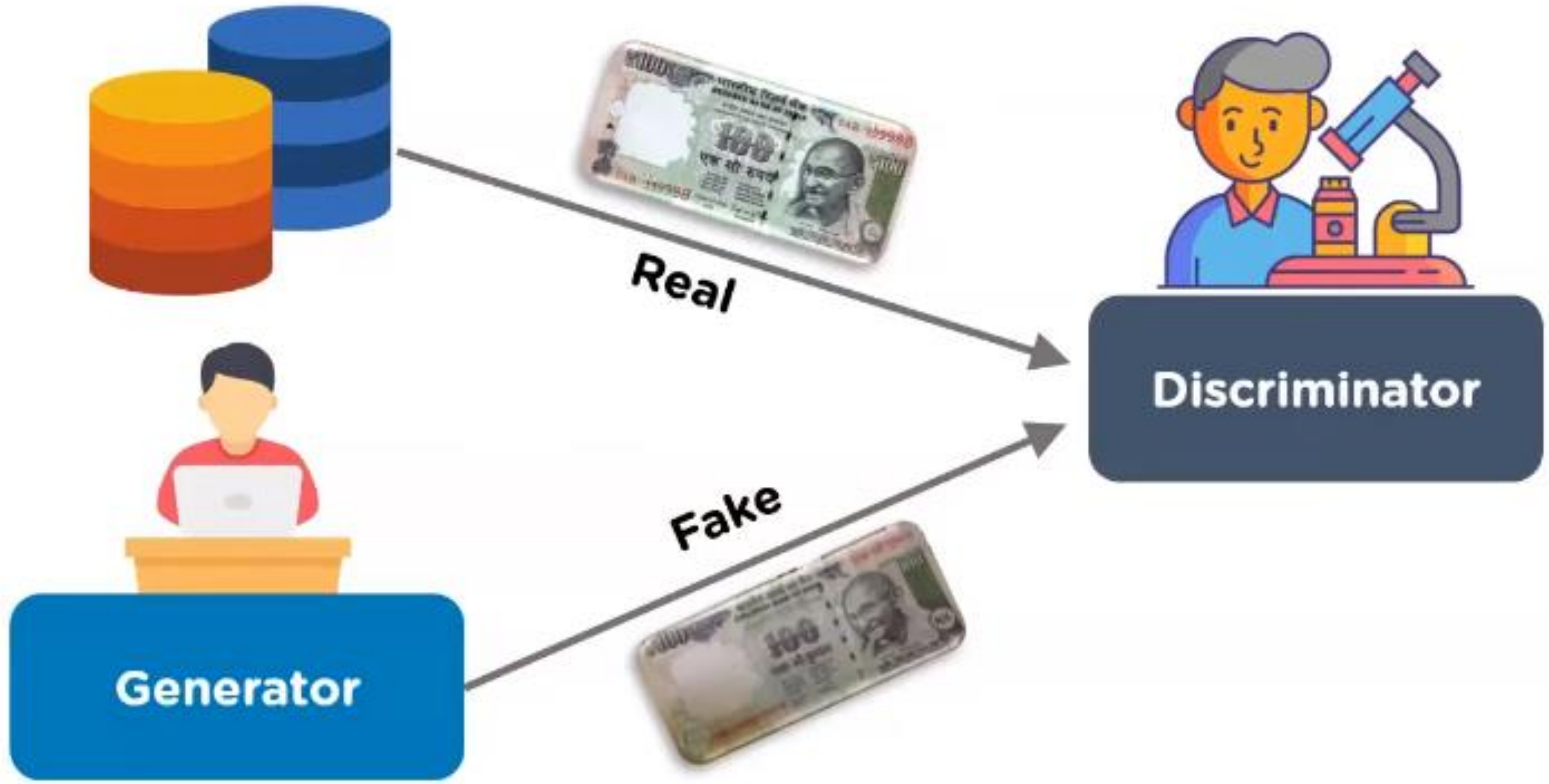
properties of attribute representations are attractive, attributes are also cumbersome to obtain as they may require domain-specific knowledge. In comparison, natural language offers a general and flexible interface for describing objects in any space of visual categories. Ideally, we could have the generality of text descriptions with the discriminative power of attributes.

GAN


D: Detective
R: Real Data
G: Generator (Forger)
I: Input for Generator

GAN

Generative Adversarial Networks consist of two models that compete with each other to analyze, capture and copy the variations within a dataset

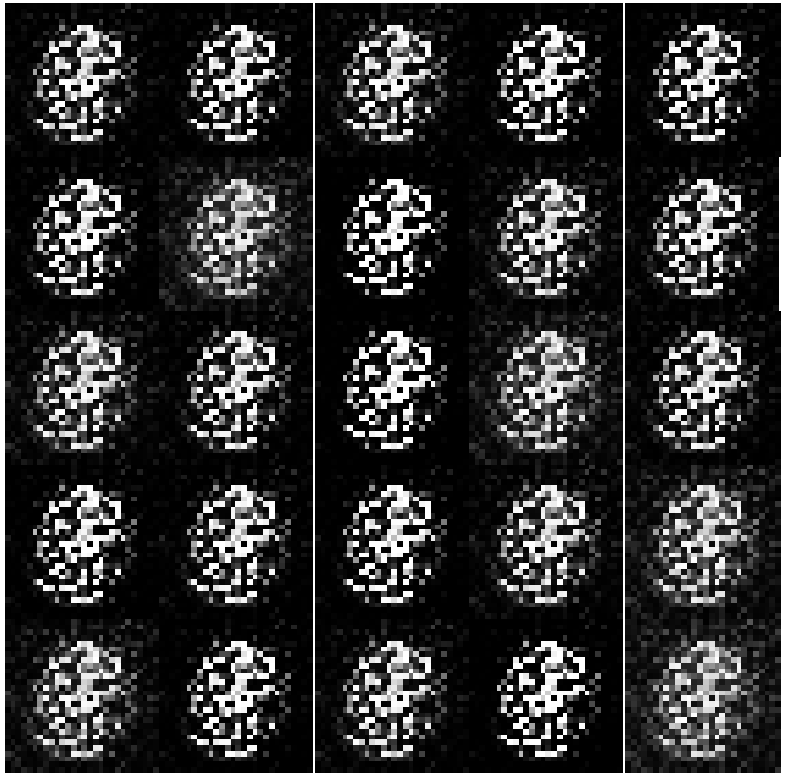


GAN

Deep Convolutional GAN (DCGAN)

Conditional Gan (CGAN)

Vanilla GAN



Epoch 1

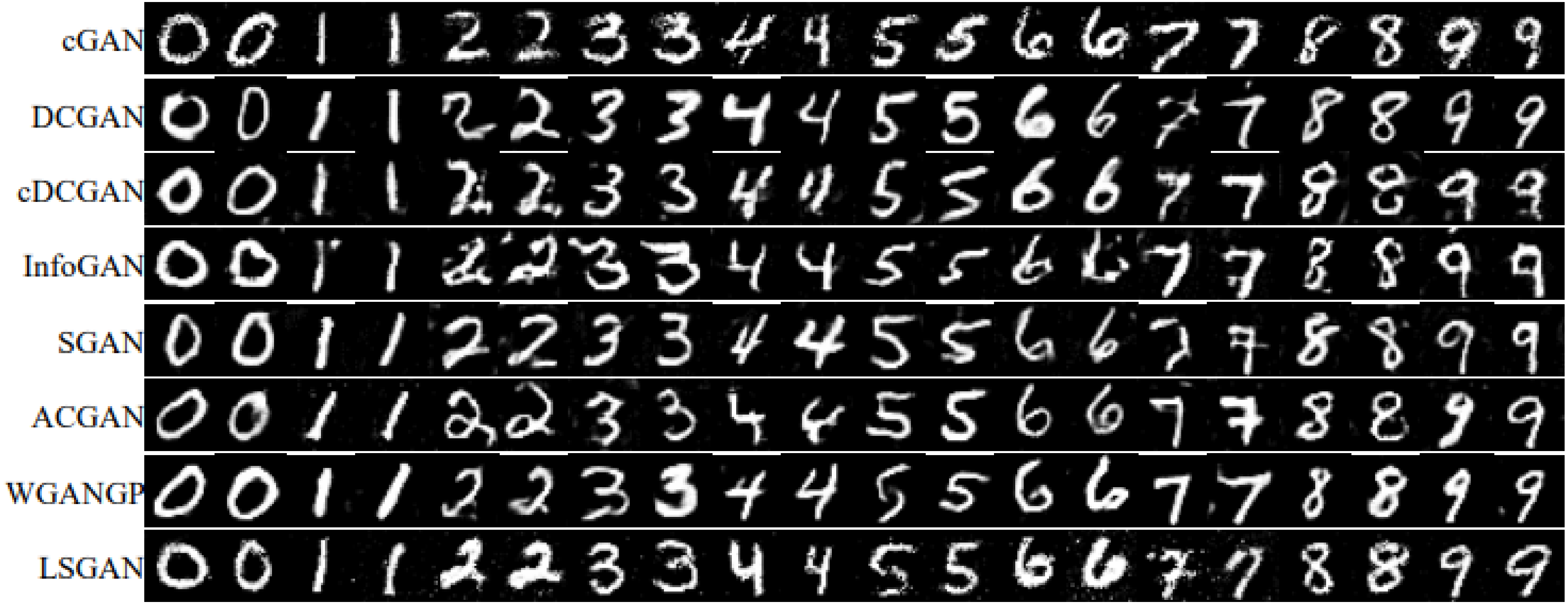
- CycleGAN
- StyleGAN
- pixelRNN
- text-2-image
- DiscoGAN
- lsGAN

WGAN (Wasserstein GAN):

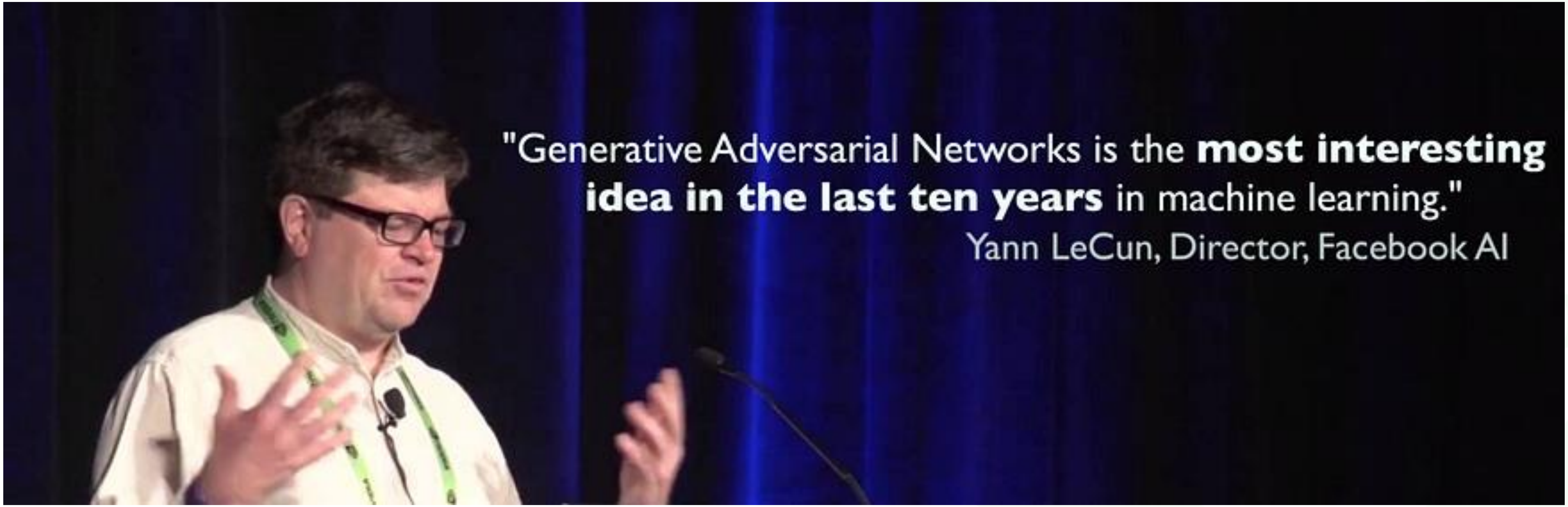
BigGAN:

StarGAN:

GAN



GAN


"Generative Adversarial Networks is the most interesting idea in the last ten years in machine learning."
Yann LeCun, Director, Facebook AI

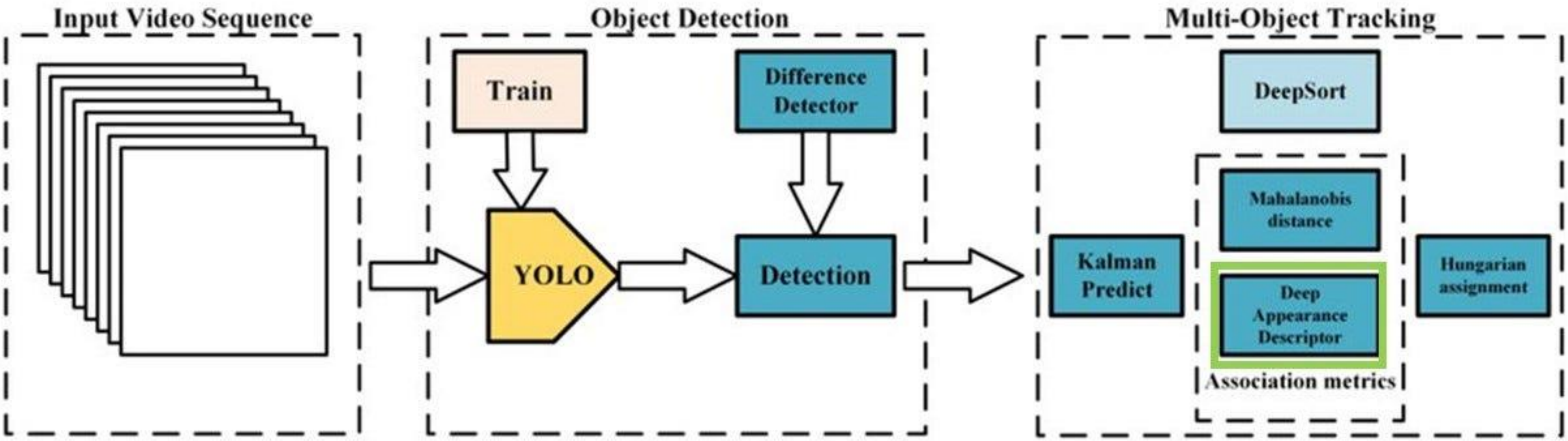
DeepSORT

DeepSORT

DeepSORT

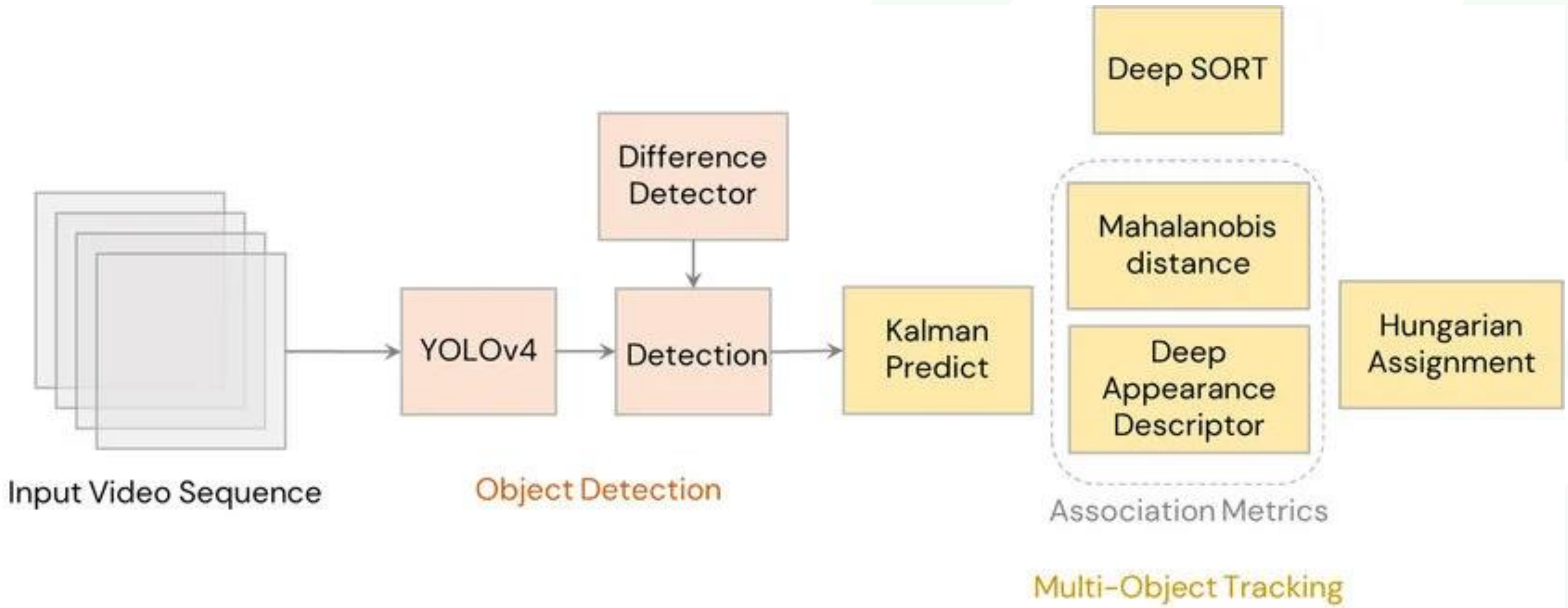
DeepSORT

Where is the Deep Learning in all of this?



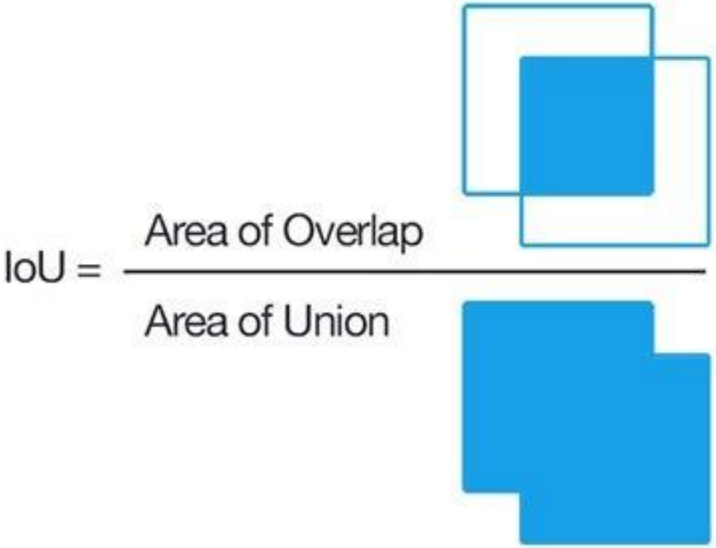
https://www.youtube.com/watch?v=GLfDnhojk-k

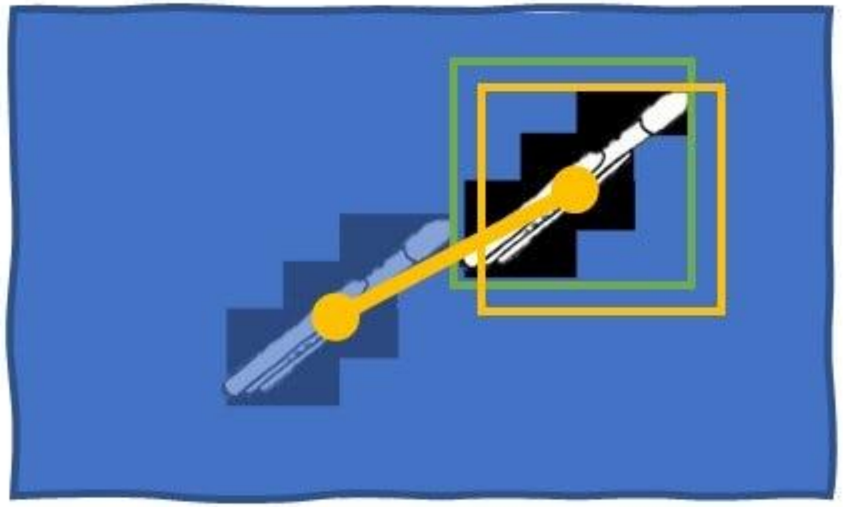
DEEP SORT

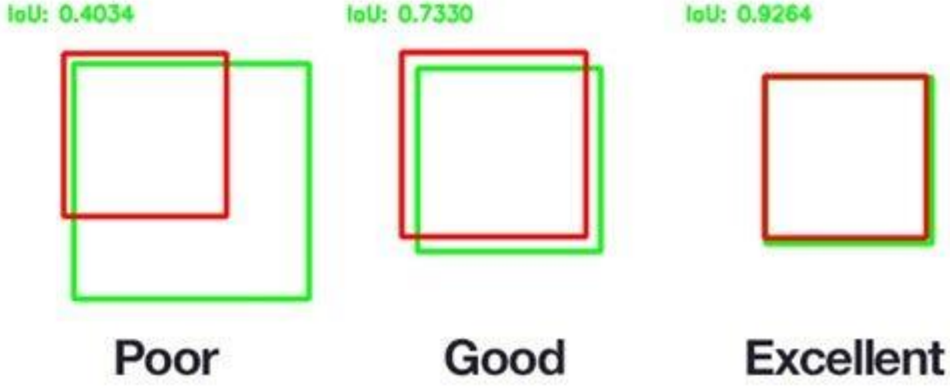

Deep SORT
Difference Detector
YOLOv4
Detection
Kalman Predict
Mahalanobis distance
Deep Appearance Descriptor
Hungarian Assignment
Input Video Sequence
Object Detection
Association Metrics
Multi-Object Tracking

Simple Online Real-time Tracking

3.Target Association







https://github.com/hardik0/Multi-Object-Tracking-Google-Colab/blob/main/YOLOv4-DeepSORT.ipynb