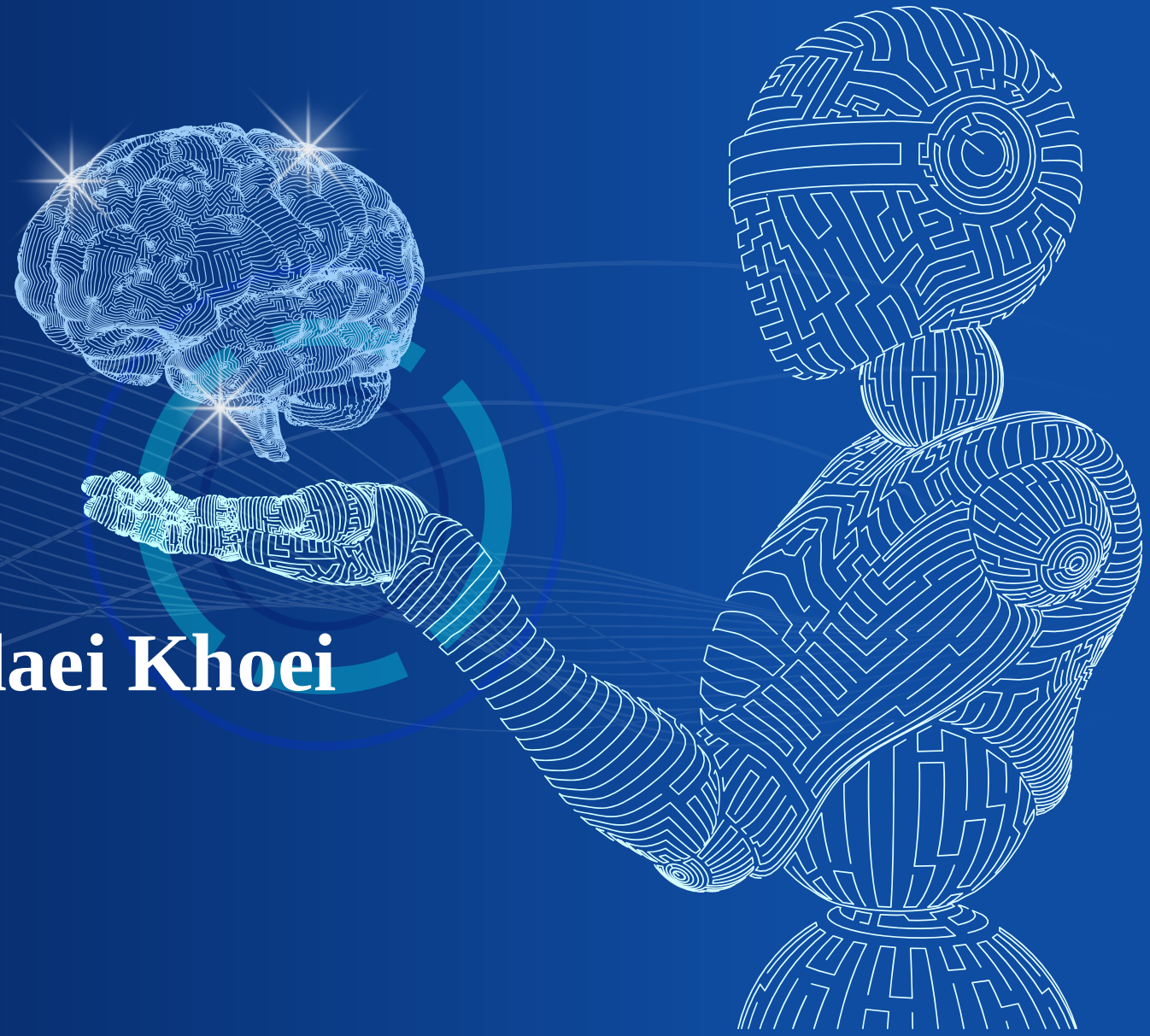
CS 7150: Deep Learning



Presented by: Dr. Tala Talaei Khoei

Supervised Learning

X_1	X_2	X_3	X_p	Y

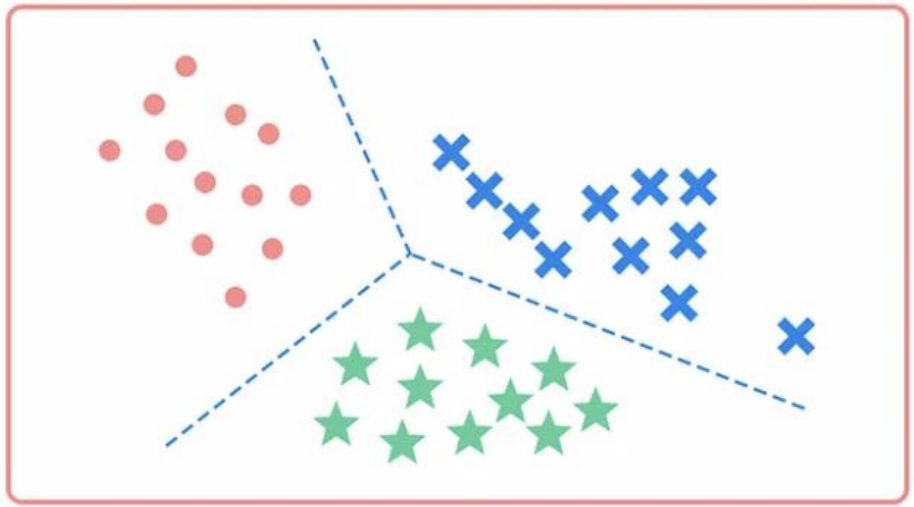
Target

Un-Supervised Learning

X_1	X_2	X_3	X_p	Y

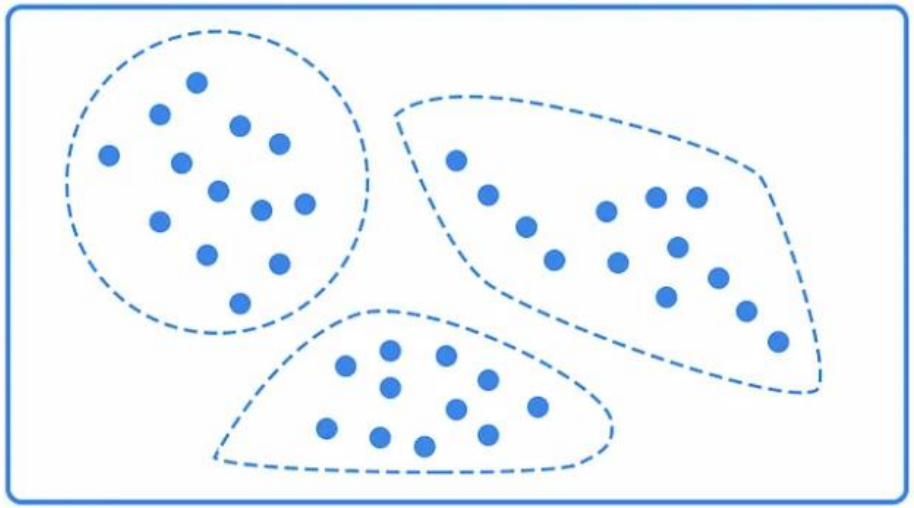
No Target

Classification



Supervised learning

Clustering



Unsupervised learning

Unsupervised Evaluation Metrics

- ***Silhouette Score:*** It computes the similarity of each data point in one cluster to data points of the neighboring clusters. The higher values indicate that the data points are well-clustered and have clear separation.

```
from sklearn.metrics import silhouette_score
data = [
    [5.1, 3.5, 1.4, 0.2],
    [4.9, 3. , 1.4, 0.2],
    [4.7, 3.2, 1.3, 0.2],
    [4.6, 3.1, 1.5, 0.2],
    [5. , 3.6, 1.4, 0.2],
    [5.4, 3.9, 1.7, 0.4],
]
clusters = [1, 1, 2, 2, 3, 3]

s = silhouette_score(data, clusters, metric="euclidean")
```

Unsupervised Evaluation Metrics

- ***Fowlkes-Mallows Scores*** measure the correctness of the cluster assignments using pairwise precision and recall. A higher score signifies higher correctness.

```
from sklearn.metrics import fowlkes_mallows_score
labels = [0, 0, 0, 1, 1, 1]
labels_pred = [1, 1, 2, 2, 3, 3]

FMI = fowlkes_mallows_score(labels, labels_pred)
```

Unsupervised Evaluation Metrics

- ***Calinski-Harabasz Index*:** Calinski-Harabasz Index measures the between-cluster dispersion against within-cluster dispersion. A higher score signifies better-defined clusters.

```
from sklearn.metrics import calinski_harabasz_score
data = [
    [5.1, 3.5, 1.4, 0.2],
    [4.9, 3. , 1.4, 0.2],
    [4.7, 3.2, 1.3, 0.2],
    [4.6, 3.1, 1.5, 0.2],
    [5. , 3.6, 1.4, 0.2],
    [5.4, 3.9, 1.7, 0.4],
]
clusters = [1, 1, 2, 2, 3, 3]

s = calinski_harabasz_score(data, clusters)
```

Homogeneity, Completeness, and V-measure: These three metrics, homogeneity, completeness, and their harmonic mean (V-measure), assess different aspects of clustering quality.

- Homogeneity computes how pure each cluster is with respect to a single class.
- Completeness measures how well all instances of a given class are assigned to the same cluster.
- V-measure combines both homogeneity and completeness into a single metric.

```
from sklearn.metrics import (
    homogeneity_score,
    completeness_score,
    v_measure_score,
)
labels = [0, 0, 0, 1, 1, 1]
labels_pred = [1, 1, 2, 2, 3, 3]

HS = homogeneity_score(labels, labels_pred)
CS = completeness_score(labels, labels_pred)
V = v_measure_score(labels, labels_pred, beta=1.0)
```

Mutual Information (MI, NMI, AMI) measures the agreement between the cluster assignments. A higher score signifies higher agreement.

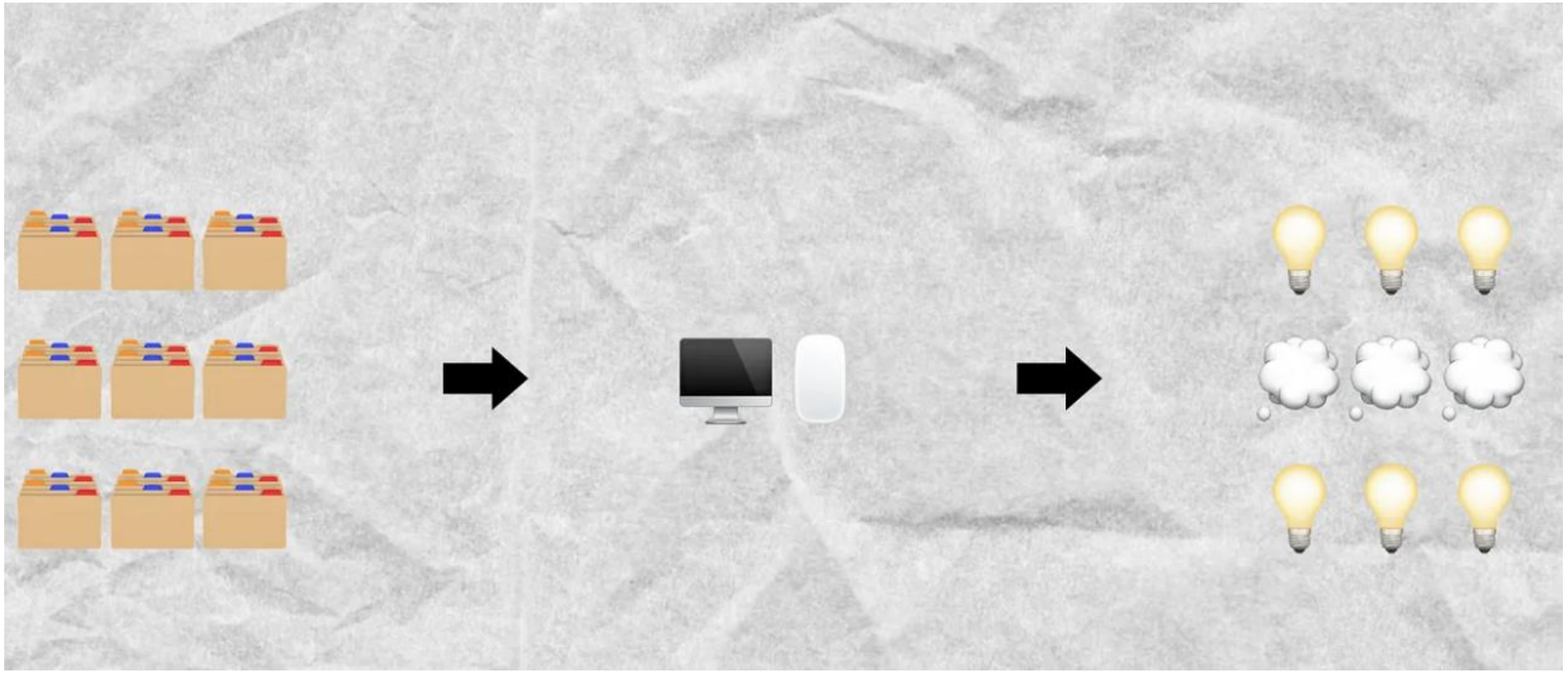
```
from sklearn.metrics import (
    mutual_info_score,
    normalized_mutual_info_score,
    adjusted_mutual_info_score,
)
labels = [0, 0, 0, 1, 1, 1]
labels_pred = [1, 1, 2, 2, 3, 3]

MI = mutual_info_score(labels, labels_pred)
NMI = normalized_mutual_info_score(labels, labels_pred)
AMI = adjusted_mutual_info_score(labels, labels_pred)
```

Rand Index (RI, ARI) measures the similarity between the cluster assignments by making pair-wise comparisons. A higher score signifies higher similarity.

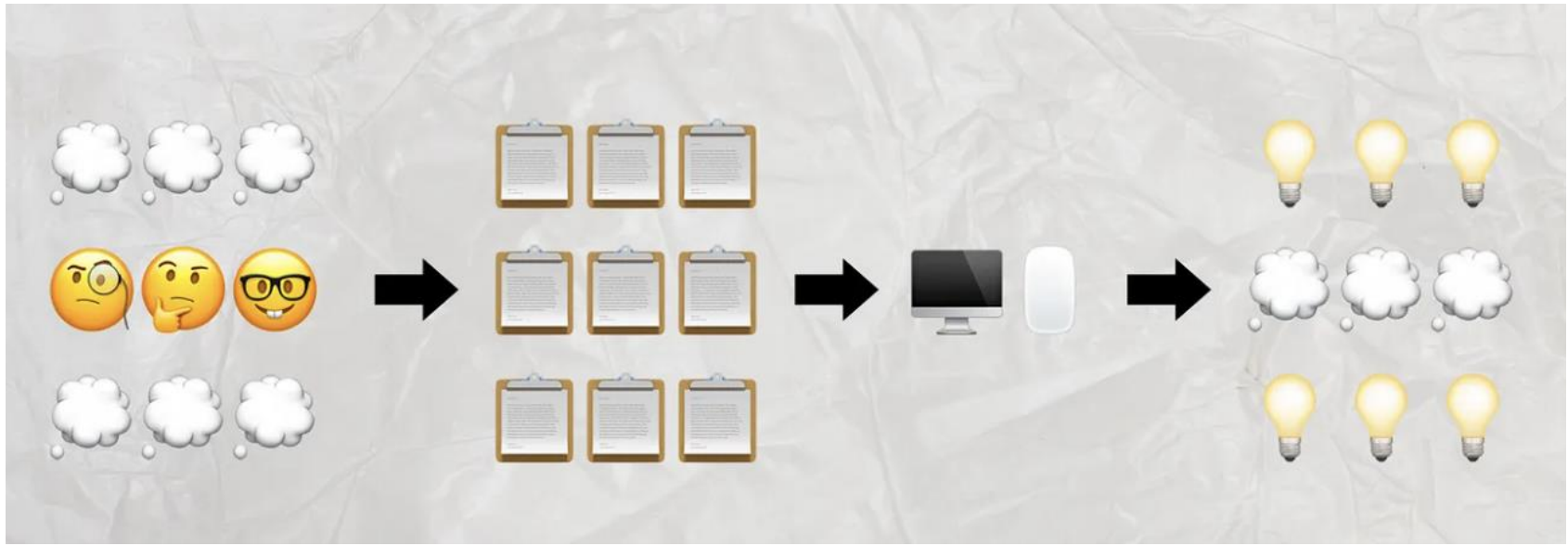
```
from sklearn.metrics import rand_score, adjusted_rand_score
labels = [0, 0, 0, 1, 1, 1]
labels_pred = [1, 1, 2, 2, 3, 3]

RI = rand_score(labels, labels_pred)
ARI = adjusted_rand_score(labels, labels_pred)
```

Expert Systems vs Machine Learning

- The field of **artificial intelligence (AI)** is concerned with developing systems that emulate human thought and behavior. These systems try to imitate the decision-making, pattern recognition, and thought processes of human beings.
- One subfield of AI, called **machine learning**, uses computers to extract meaningful information from data. Experts feed data into computers. These computers learn to pick up patterns and make decisions about the data without necessarily being told what to do.



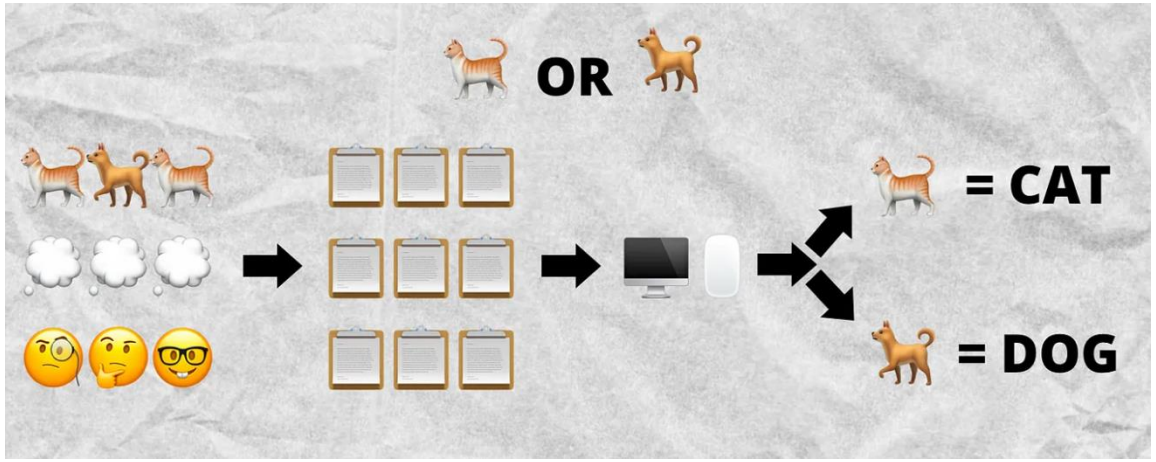
- Before the development of machine learning, however, **expert systems** were used to gain insight from data. Expert systems differ from machine learning approaches in AI. An expert system requires developers to create a strict set of rules to imitate the decision-making processes of experts in the field. Machine learning approaches require less structure; we simply feed data into the machine and see what insights the machine gains.



Example: Image Classification

- *Given an image from a dataset composed only of dog and cat pictures, how might someone determine whether the shown image is of a cat or a dog?*
- Their thought process might go something like this:
  - *How much fur does the animal have?*
  - *Does the animal have a long, thin tail?*
  - *Does the animal have pointy, upright ears?*
  - *Is the animal's nose small and triangular shaped?*

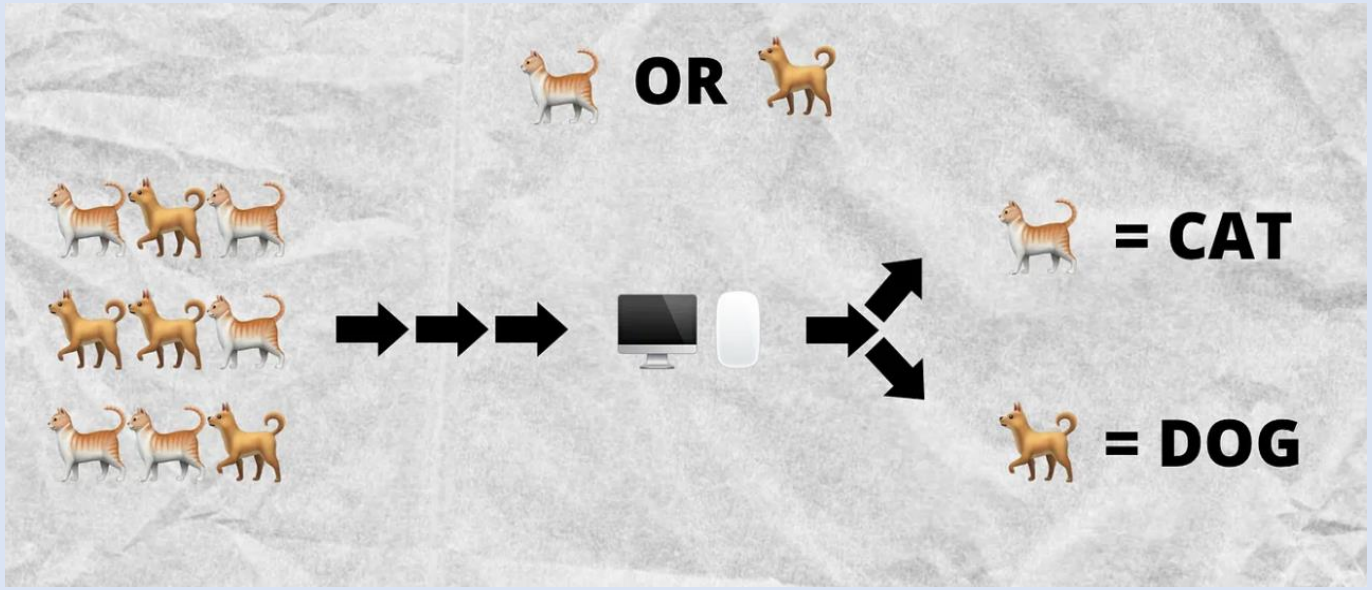
In an expert based classification system, this thought process would have to be systematically written out. The set of rules has to be clear and highly structured for the task and also account for deviations from common cases. Most cats and dogs have four legs and fur. What if the image showed an animal with an amputated leg or a hairless cat? What if a dog also has a long, thin tail, pointy, upright ears, and a small size? Those cases would also have to be accounted for and written into the set of rules.



Example: Image Classification (…..)

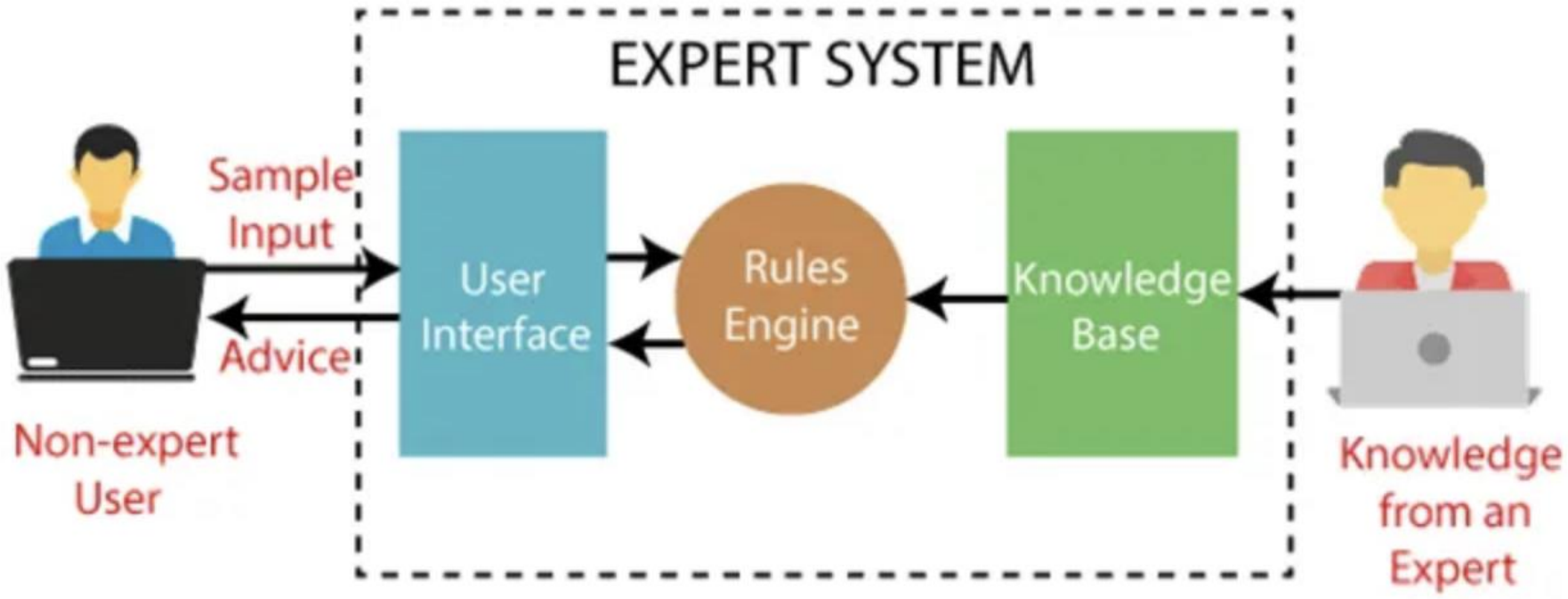
The task of generating a set of rules is the most difficult part of expert based learning systems, since this step "can require impractical amounts of human intervention and ingenuity" * to account for all the decision-making process and aberrations from common cases. For most tasks, rule-based expert systems offer an impractical and time-consuming solution.

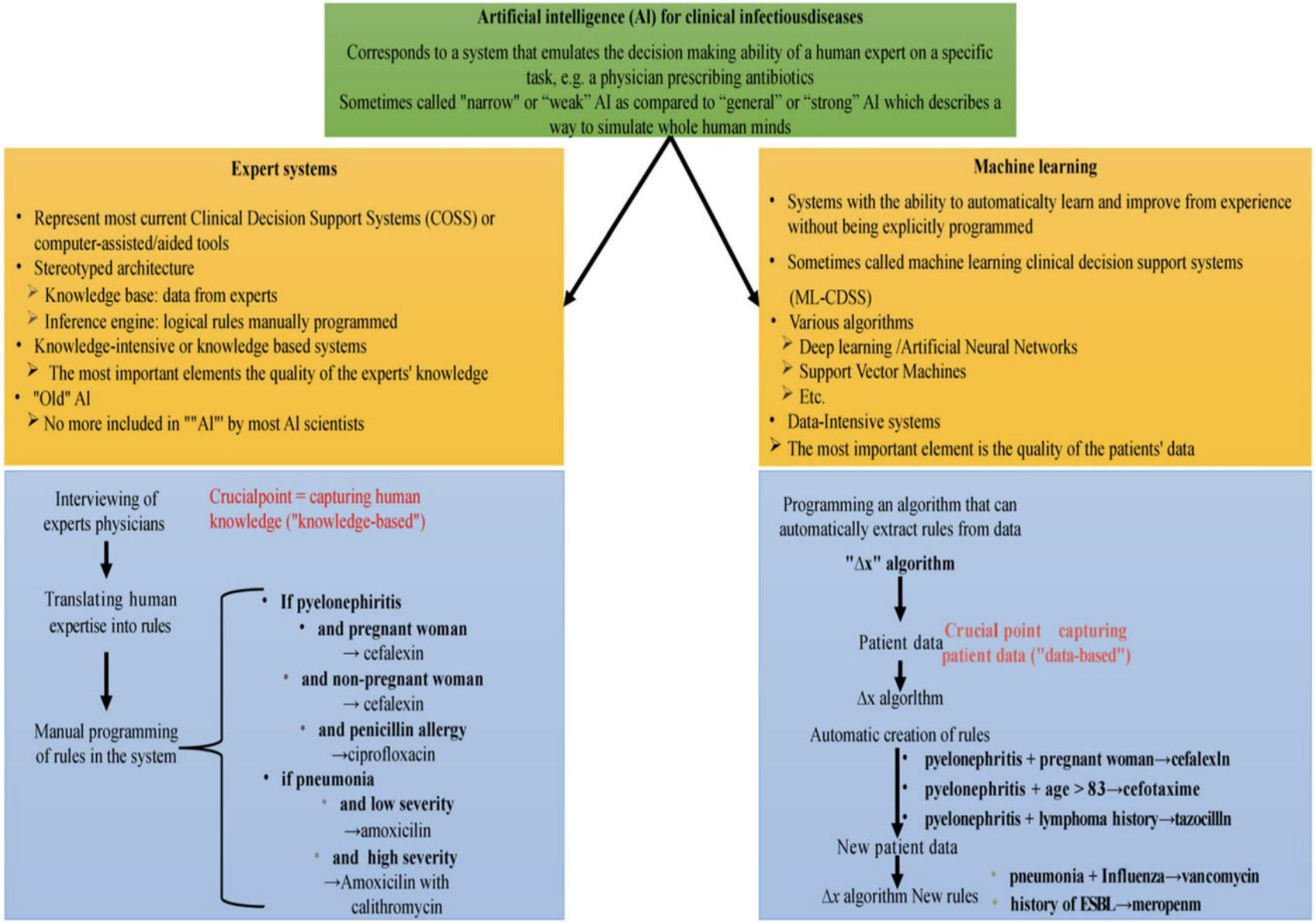
In a machine learning algorithm, the cat vs dog classification task would be easier. Imagine you showed a child multiple images of cats and dogs, each time pointing out the correct word for the animal in the image. Over time, the child intuitively picks up on features that distinguish cats from dogs. Intuitively, when that child encounters new instances of cats and dogs, they'll be able to identify each new example. In a machine learning task, machines will learn in much the same way as people do.



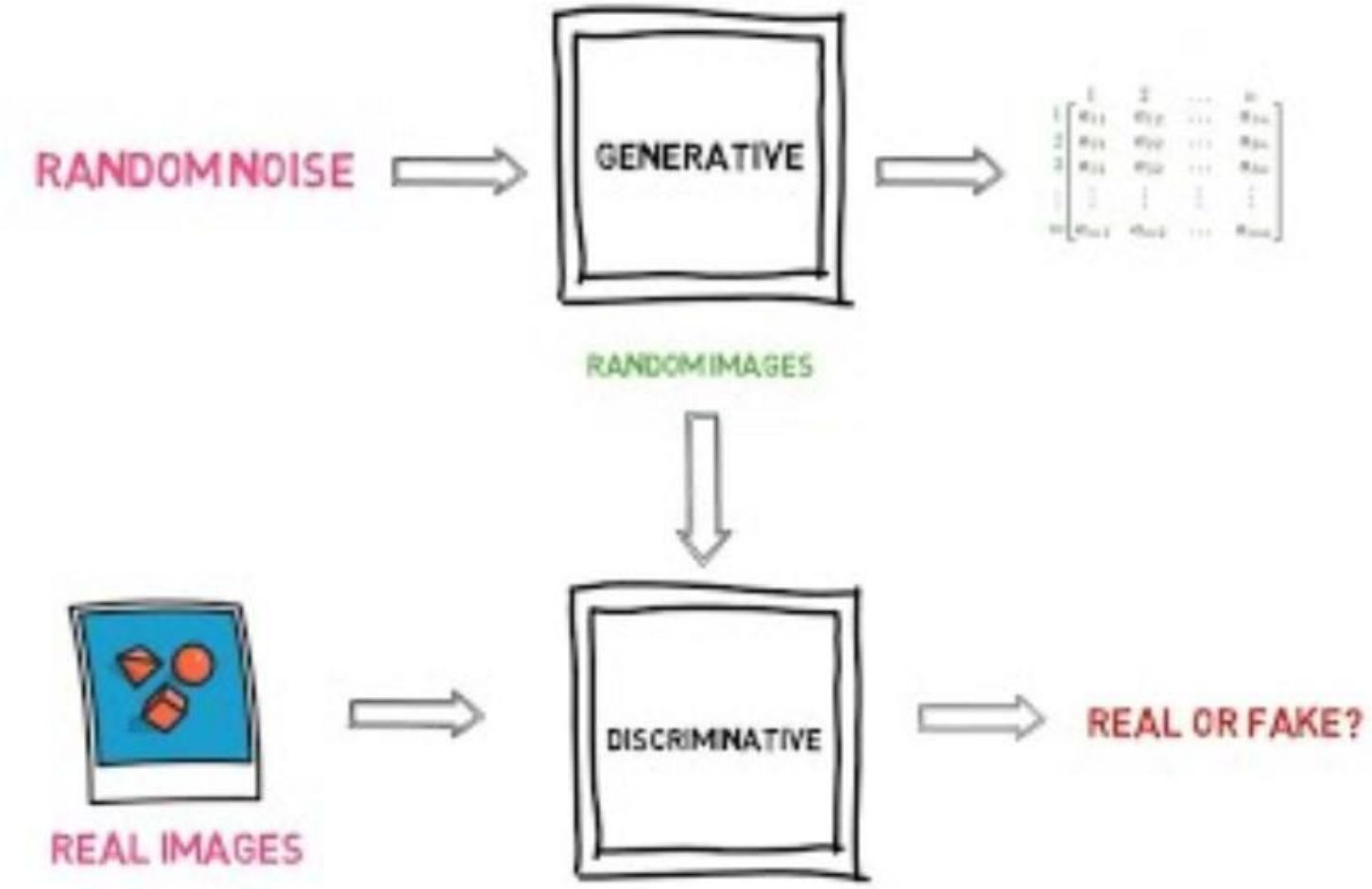
Expert Systems in Medical Sciences

One common implementation of expert systems is in diagnosing diseases, where the knowledge base contains medical information, and a doctor enters a patient's symptoms into the user interface (Copley). The expert system will then be able to make a diagnosis based on the query and prior information. Some diseases that expert systems can diagnose are septicemia, urinary tract infection, and chronic prostatitis (Winstanley). Dendral, as described earlier, can be used in the medical field to help researchers identify organic compounds within samples. Dendral is also the source of many medical expert systems, since its software and knowledge base can be modified to fit the needs of different medical practitioners





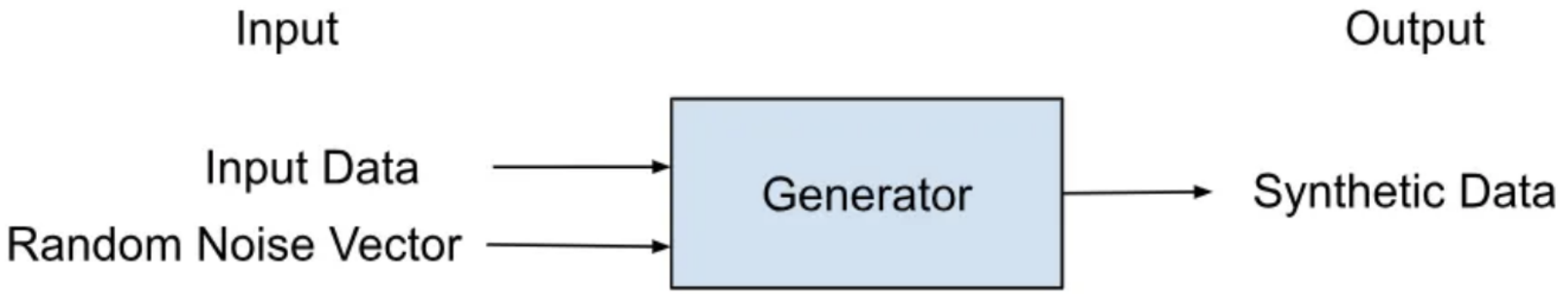
Reference: Kaur, I., Behl, T., Aleya, L., Rahman, H., Kumar, A., Arora, S. and Bulbul, I.J., 2021. Artificial intelligence as a fundamental tool in management of infectious diseases and its current implementation in COVID-19 pandemic. *Environmental Science and Pollution Research*, *28*(30), pp.40515-40532.


RANDOM NOISE
GENERATIVE
RANDOM IMAGES
REAL IMAGES
DISCRIMINATIVE
REAL OR FAKE?

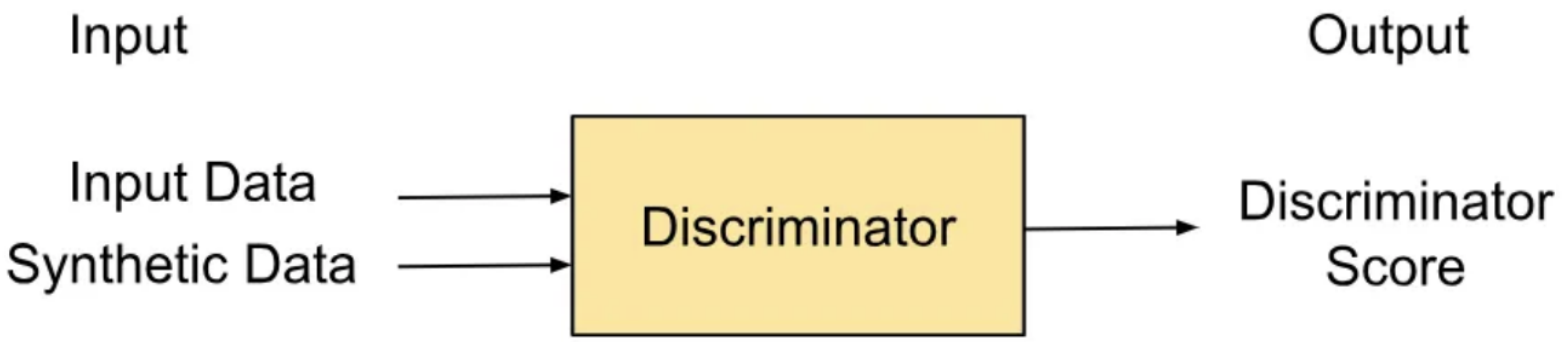
Introduction to Generative adversarial networks (GANs)

- Generative Adversarial Network, commonly known as GAN, is a deep learning framework widely used in image generation. It corresponds to a minimax two-player game. GAN consists of two neural networks, which are known as generator and discriminator. GAN belongs to unsupervised learning.

- The generator receives the input data and random noise vector, generating synthetic data which will be fed into the discriminator.



- The discriminator receives both input data and synthetic data, and it is trained to differentiate whether the given data is real or synthetic data.

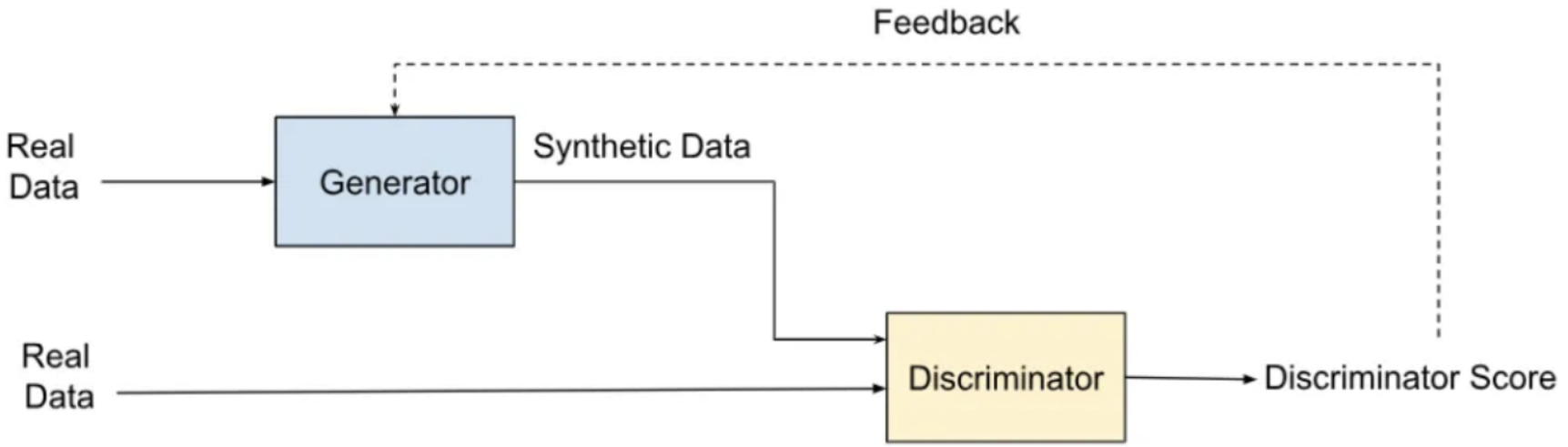


Note:
GANs introduce the concept of **adversarial learning**, as they lie in the rivalry between two neural networks. These techniques have enabled researchers to create realistic-looking but entirely computer-generated photos of people's faces. They have also allowed the creation of controversial "deepfake" videos. Actually, GANs can be used to imitate any data distribution (image, text, sound, etc.).

- An example of GANs' results from 2018 is given **Figure:** these images are fake yet very realistic. The generation of these fictional celebrity portraits, from the database of real portraits Celeba-HQ composed of 30,000 images, took 19 days. The generated images have a size of 1024×1024.

How does it work?

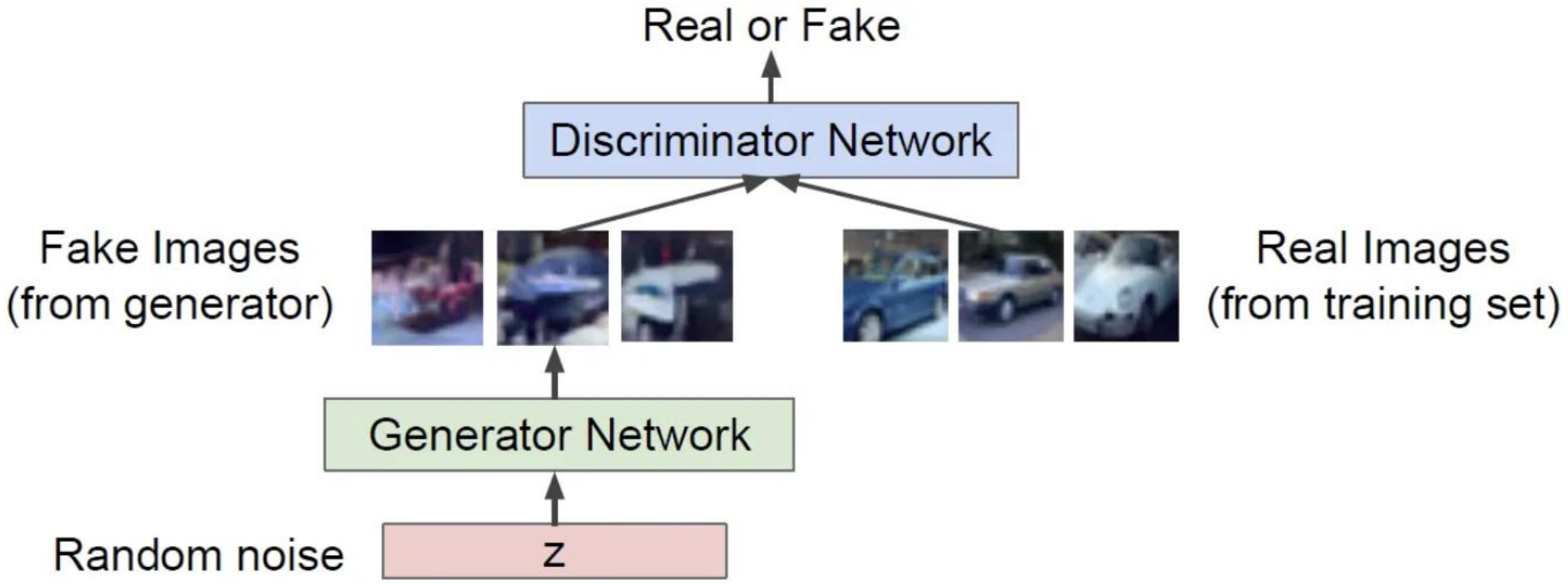
- During training, the training of generator and discriminator models takes place alternately. When the generator is training, the discriminator's training stops and vice versa.
- The generator model tries its best to create synthetic data that is indistinguishable from real data to fool the discriminator, while the discriminator tries its best to catch and identify the synthetic data created by the generator model.
- When the discriminator successfully detects synthetic data generated by the generator, then feedback is given to the generator to improve the quality of data generated. Eventually, a point of equilibrium will reach when the discriminator obtains discrimination score of 0.5 (50%) for the input to be either fake or real.



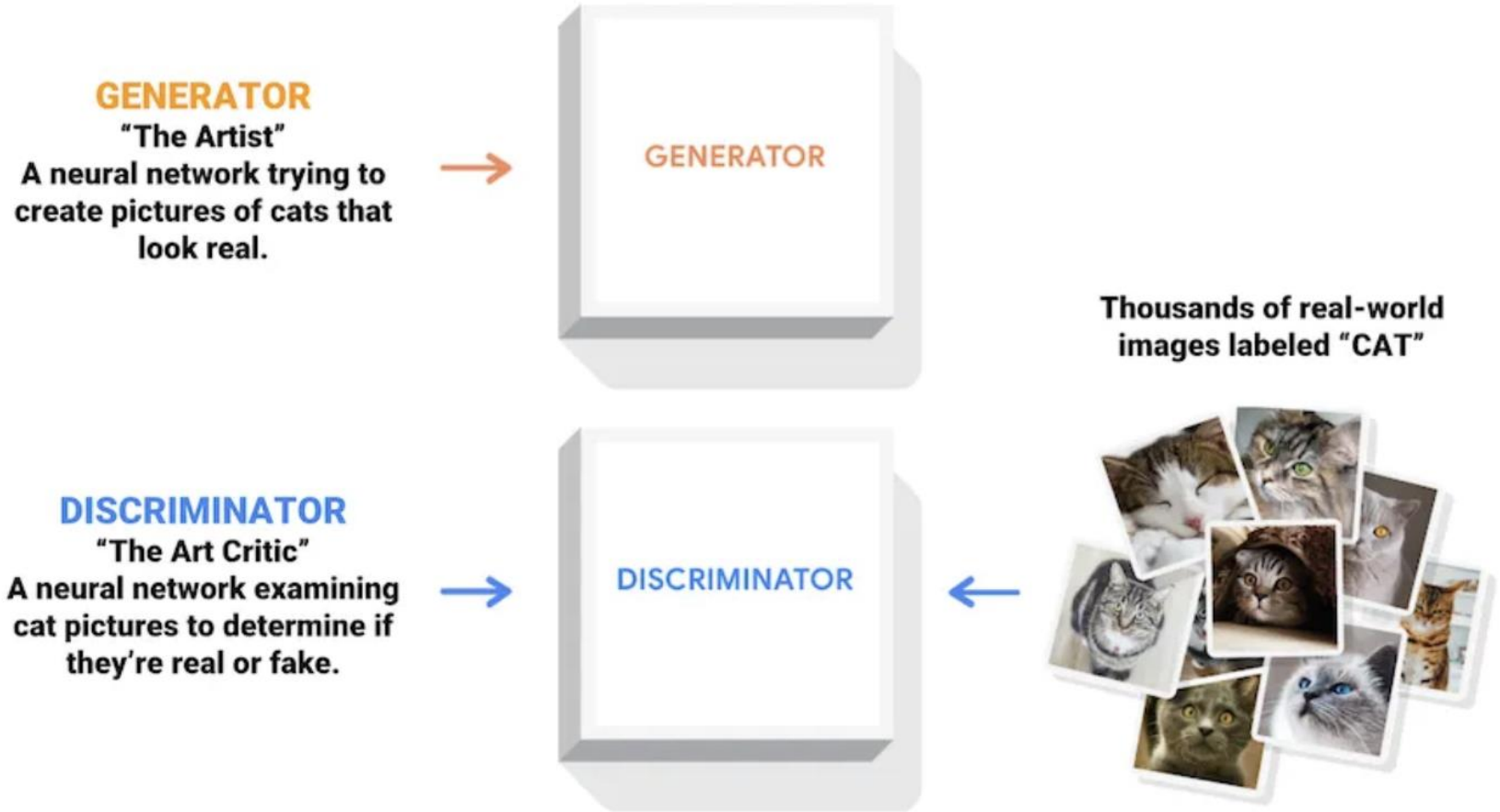
Details

- **Generative adversarial networks** (GANs) are a generative model with implicit density estimation, part of unsupervised learning and are using two neural networks. Thus, we understand the terms “generative” and “networks” in “generative adversarial networks”.

1) The principle: generator vs discriminator

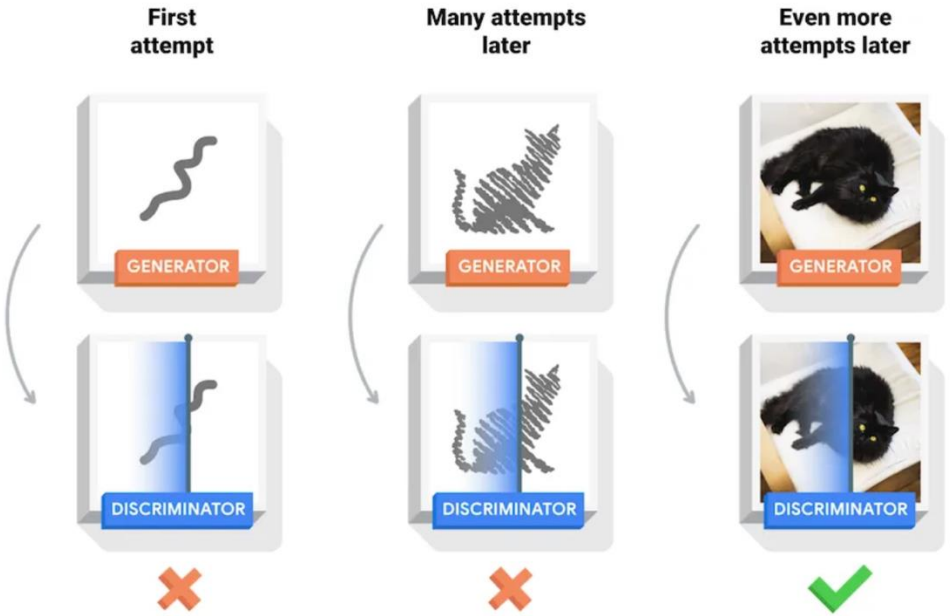


- The principle is a two-player game: a neural network called the generator and a neural network called the discriminator. The **generator** tries to fool the discriminator by generating real-looking images while the **discriminator** tries to distinguish between real and fake images. Hence, we understand the term “adversarial” in “generative adversarial networks”



- At the bottom left of **Figure** , we can see that our generator samples from a simple distribution: random noise. The generator can be interpreted as an artist and the discriminator as an art critic.

- During training, the generator progressively becomes better at creating images that look real, while the discriminator becomes better at telling them apart. The process reaches equilibrium when the discriminator can no longer distinguish real from fake images. See Figure . Thus, if the discriminator is well trained and the generator manages to generate real-looking images that fool the discriminator, then we have a good generative model: we are generating images that look like the training set.
- After this training phase, we only need the generator to sample new (false) realistic data. We no longer need the discriminator. Note that the random noise guarantees that the generator does not always produce the same image (which can deceive the discriminator).
- At the beginning of the training in Figure , the generator only generates a random noise that does not look like the training data.



Mathematically: the two-player minimax game

- The generator *G* and the discriminator *D* are jointly trained in a **two-player minimax game** formulation. The minimax objective function is:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_d} \left(G_{\theta_g}(z) \right) \right) \right]$$

Equation (1)

where θ_g is the parameters of *G* and θ_d is the parameters of *D*.

In the following, we simply refer to $D\_\{\theta\_d\}$ as *D* and $G\_\{\theta\_g\}$ as *G*.

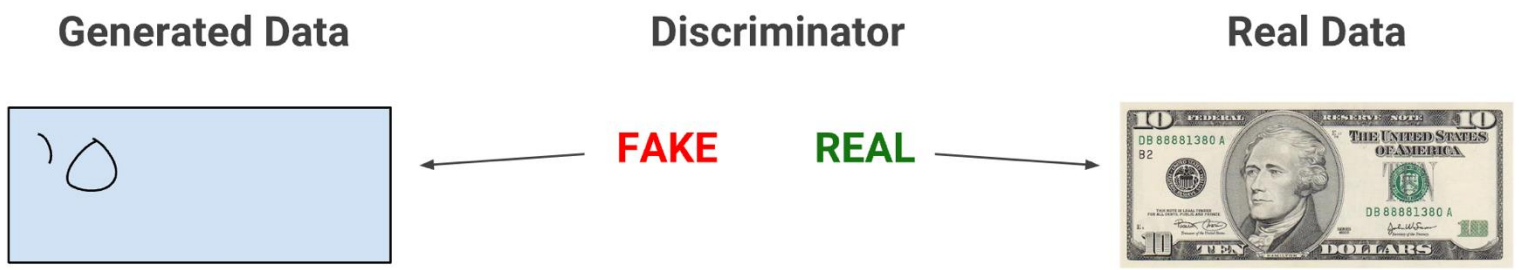
By definition, *D* outputs the likelihood of real image in interval *[0, 1]*:
- *D(x)* equals *1* (or is close to *1*) if *D* considers that *x* is a real data,
- *D(x)* equals *0* (or is close to *0*) if *D* considers that *x* is a fake data (e.g. a generated data).

- Because $x \sim p\_\{data\}$, *x* is a real data. By definition of *G*, *G(z)* is a fake generated data. For example, *x* would be a real-life image of a cat and *G(z)* would be a fake generated image of a cat. Thus, *D(x)* is the output of the discriminator for a real input *x* and *D(G(z))* is the output of the discriminator for a fake generated data *G(z)*.

The two-player minimax game from **Equation (1)** was written such that ϑ_g and ϑ_d evolve so that the following points **)** are true:

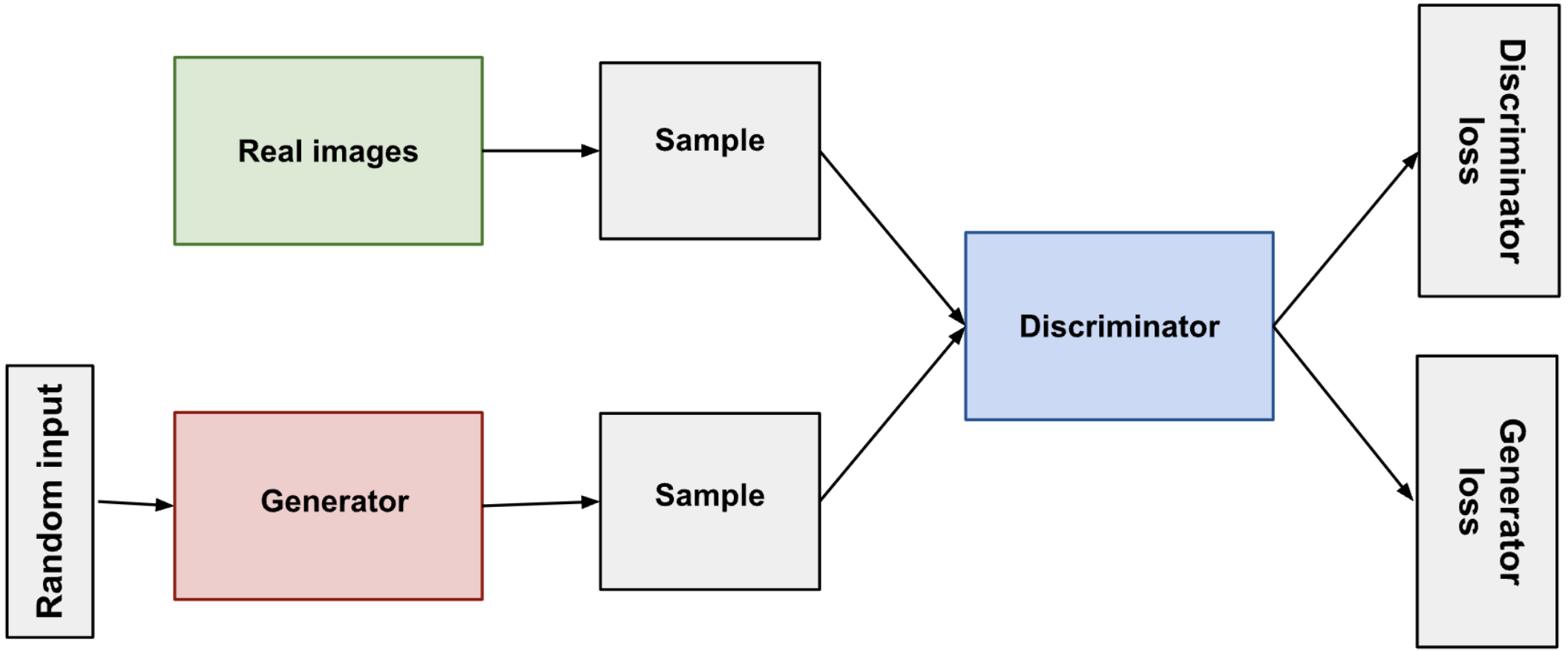
1. The discriminator *D* tries to distinguish between real data *x* and fake data *G(z)*.
   More precisely, the discriminator *D* plays with ϑ_d (ϑ_g being fixed) to maximize the objective function such that *D(x)* is close to *1* (*x* being real data) and such that *D(G(z))* is close to *0* (a generated data is detected as false).
2. The generator *G* tries to fool the discriminator *D* into thinking that its fake generated data is real.
   More precisely, the generator *G* plays with ϑ_g (ϑ_d being fixed) to minimize the objective function such that *D(G(z))* is close to *1* (a false generated data is detected as true by the discriminator).

- When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake:



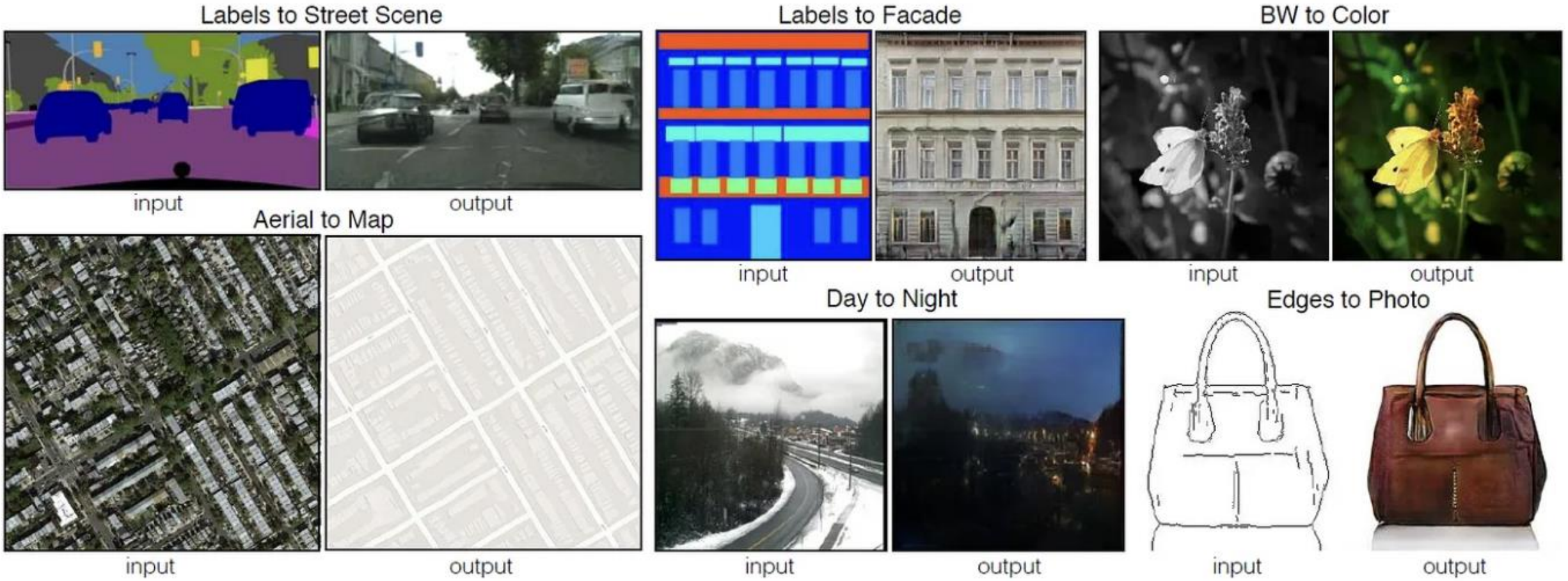
- As training progresses, the generator gets closer to producing output that can fool the discriminator:

- Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases.


Real images
Sample
Random input
Generator
Sample
Discriminator
Discriminator loss
Generator loss

Example

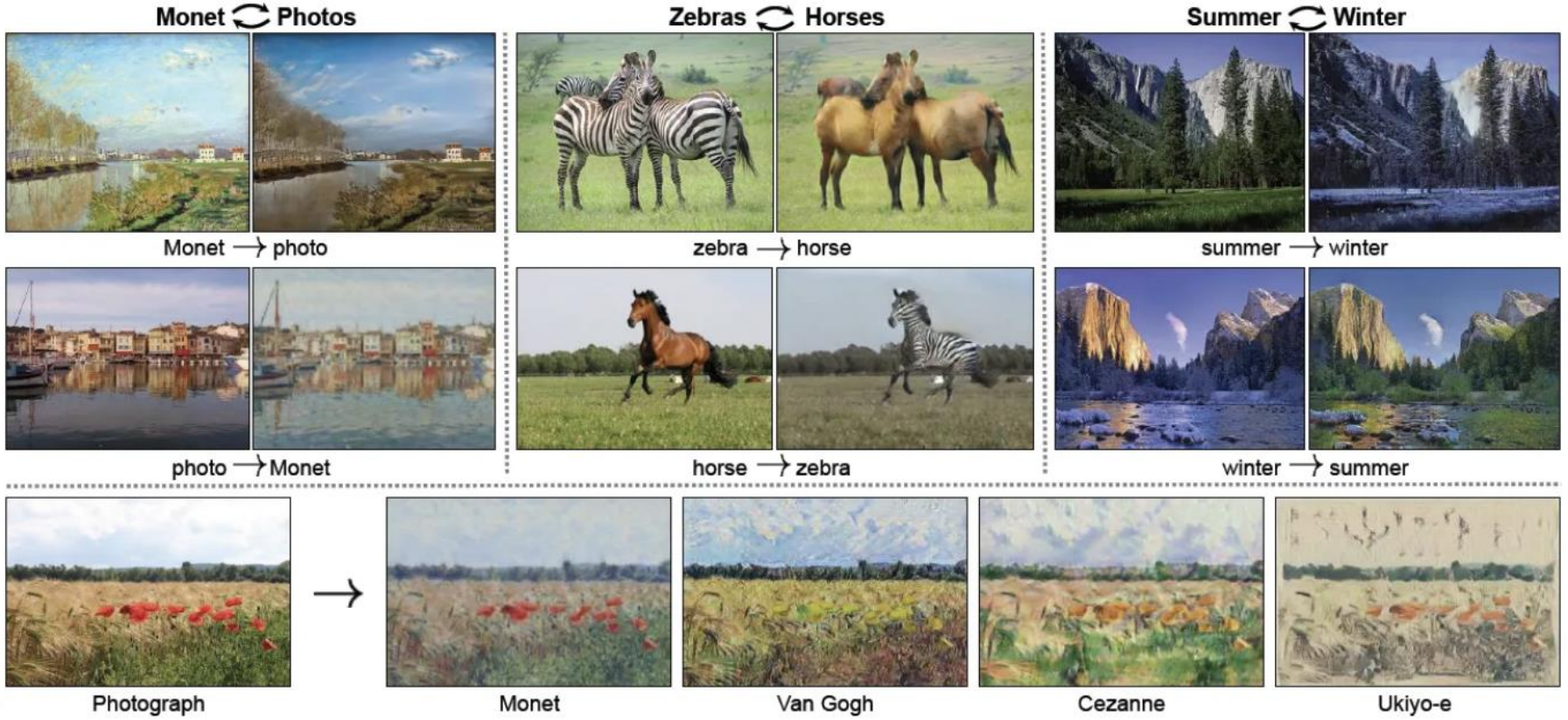
The aerial to map feature can be very useful to Google Maps or similar applications and the edges to photo feature can help designers.



Another Example

- These real images are transposed into realistic fictional images — or vice versa — with the CycleGan developed by researchers at the University of Berkeley. The concept, called **image-to-image translation**, is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using a training set of aligned image pairs.

https://www.tensorflow.org/tutorials/generative/cyclegan



Please refer to the following code of the paper "Generative Adversarial Networks (NIPS 2016 tutorial)":

- **https://www.tensorflow.org/tutorials/generative/cyclegan**

Steps to Train Generative Adversarial Networks

Step 1: Define the problem

Do you want to generate fake images or fake text. Here you should completely define the problem and collect data for it.

Step 2: Define architecture of GAN

Define how your GAN should look like. Should both your generator and discriminator be multi-layer perceptron, or convolutional neural networks? This step will depend on what problem you are trying to solve.

Step 3: Train Discriminator on real data for n epochs

Get the data you want to generate fake on and train the discriminator to correctly predict them as real. Here value n can be any natural number between 1 and infinity.

Step 4: Generate fake inputs for generator and train discriminator on fake data

Get generated data and let the discriminator correctly predict them as fake.

Step 5: Train generator with the output of discriminator

Now when the discriminator is trained, you can get its predictions and use it as an objective for training the generator. Train the generator to fool the discriminator.

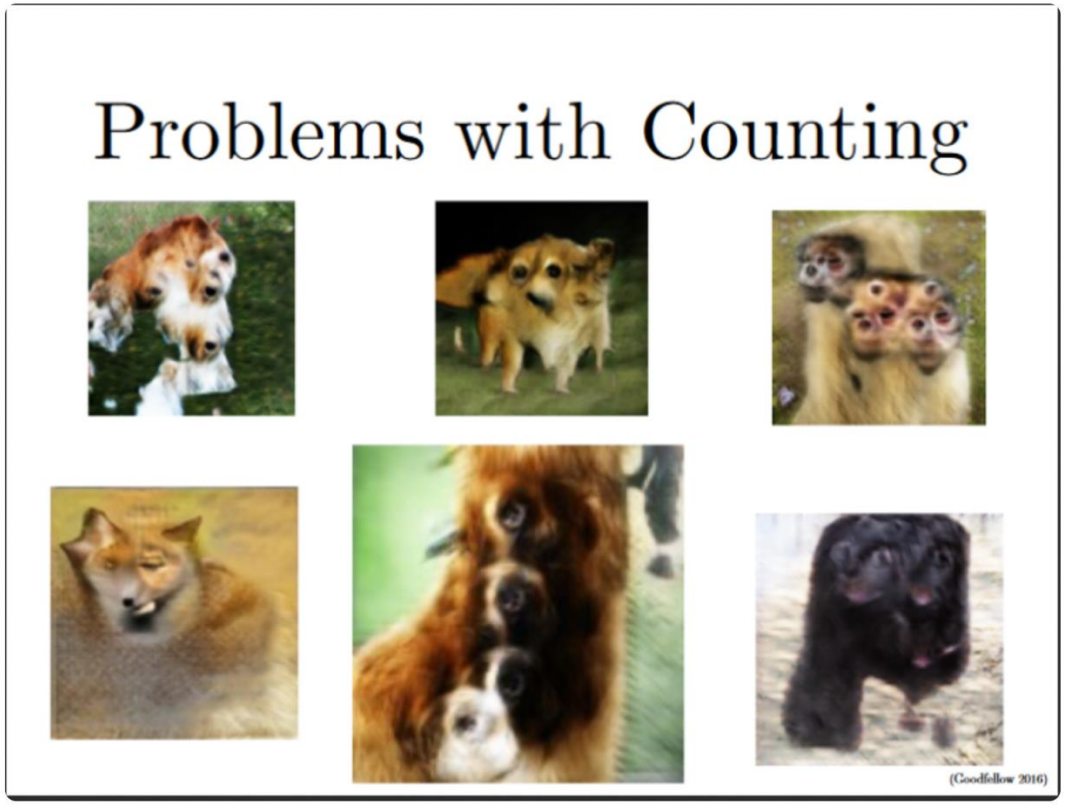
Repeat step 3 to step 5 for a few epochs.

Step 6: Check if the fake data manually if it seems legit. If it seems appropriate, stop training, else go to step 3.

This is a bit of a manual task, as hand evaluating the data is the best way to check the fakeness. When this step is over, you can evaluate whether the GAN is performing well enough

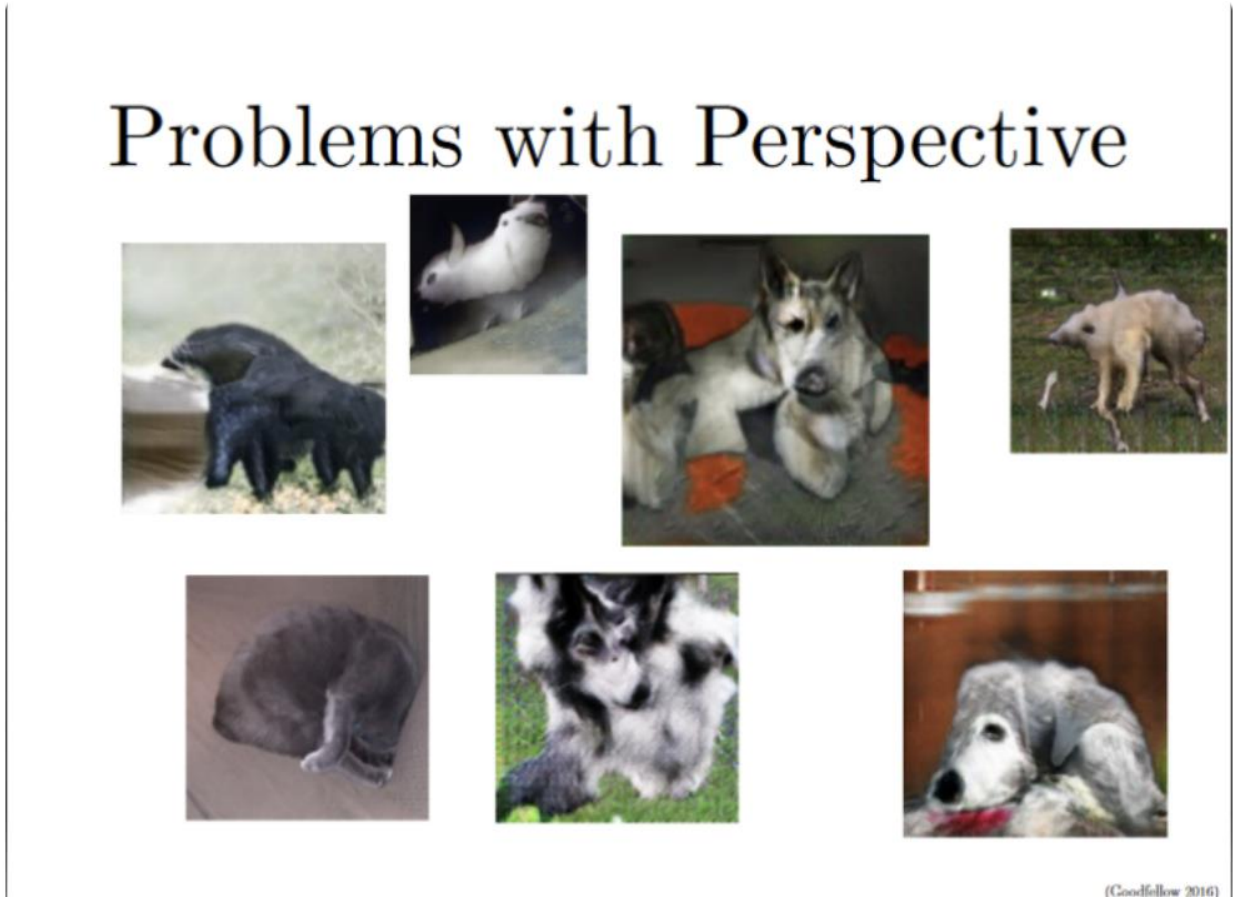
Challenges with Generative Adversarial Networks

- **Problem with Counting**: GANs fail to differentiate how many of a particular object should occur at a location. As we can see below, it gives more number of eyes in the head than naturally present.



Challenges with Generative Adversarial Networks

- **Problems with Perspective:** GANs fail to adapt to 3D objects. It doesn't understand perspective, i.e.difference between frontview and backview. As we can see below, it gives flat (2D) representation of 3D objects.



Challenges with Generative Adversarial Networks

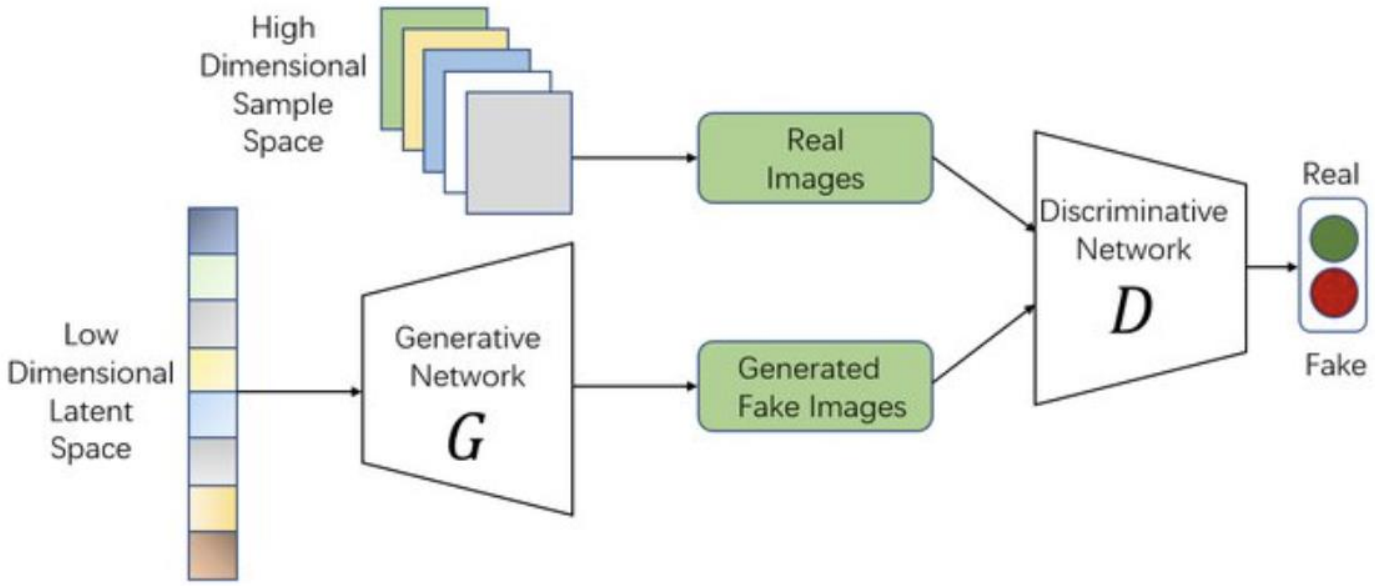
- **Problems with Global Structures:** Same as the problem with perspective, GANs do not understand a holistic structure. For example, in the bottom left image, it gives a generated image of a quadruple cow, i.e. a cow standing on its hind legs and simultaneously on all four legs. That is definitely not possible in real life!

Applications of GANs	
	Generate new data from available data
	Generate realistic pictures of people that have never existed.
	Gans is not limited to Images, It can generate text, articles, songs, poems, etc.
	Generate Music by using some clone Voice
	Text to Image Generation (Object GAN and Object Driven GAN)
	Creation of anime characters in Game Development and animation production.
	Image to Image Translation
	Low resolution to High resolution
	Prediction of Next Frame in the video
	Interactive Image Generation
	Speech

What are the types of generative adversarial networks?

There are different types of GAN models depending on the mathematical formulas used and the different ways the generator and discriminator interact with each other.

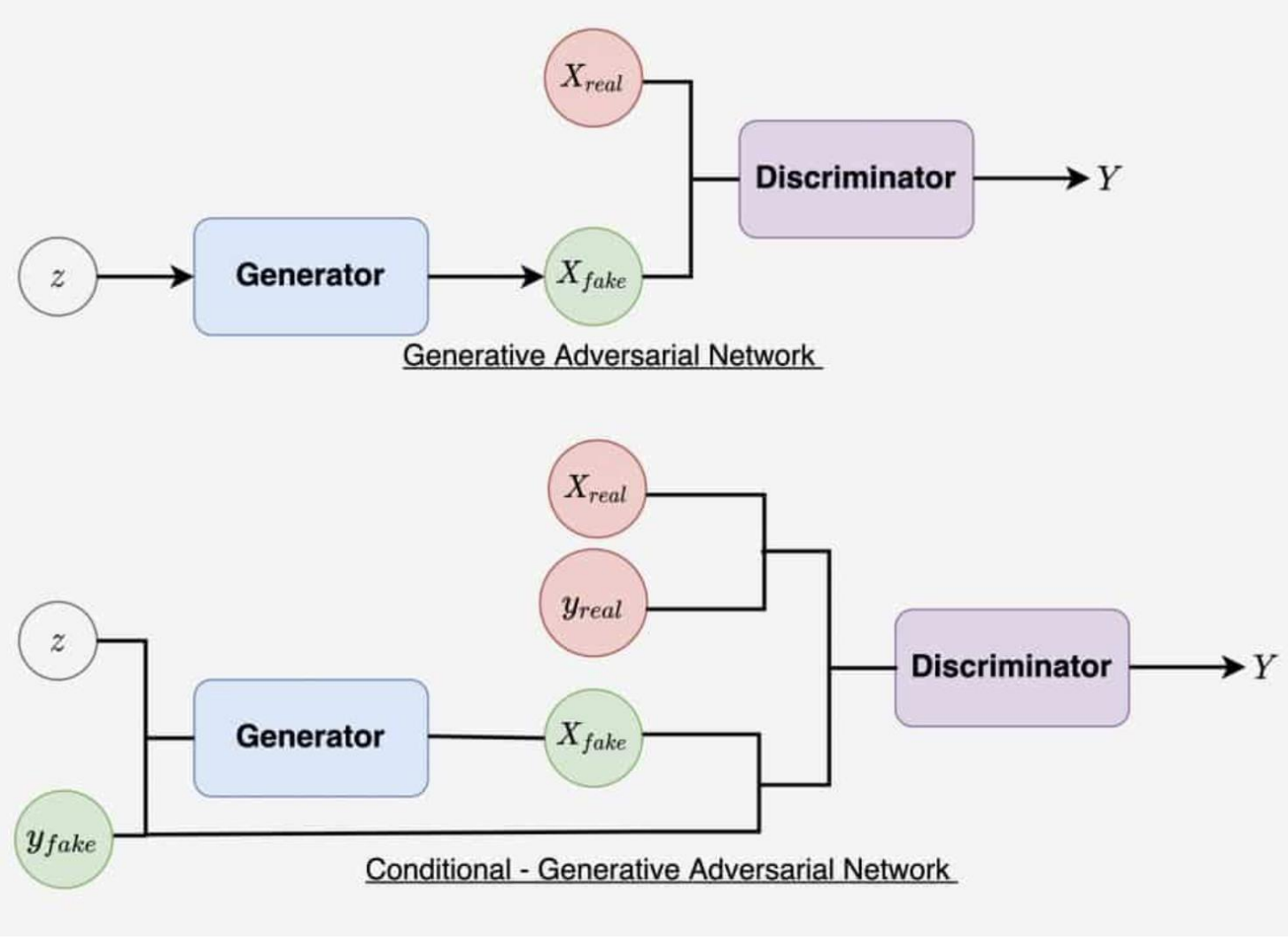
Vanilla GAN: This is the basic GAN model that generates data variation with little or no feedback from the discriminator network. A vanilla GAN typically requires enhancements for most real-world use cases.



What are the types of generative adversarial networks?

Conditional GAN

A conditional GAN (cGAN) introduces the concept of conditionality, allowing for targeted data generation. The generator and discriminator receive additional information, typically as class labels or some other form of conditioning data.
For instance, if generating images, the condition could be a label that describes the image content. Conditioning allows the generator to produce data that meets specific conditions.

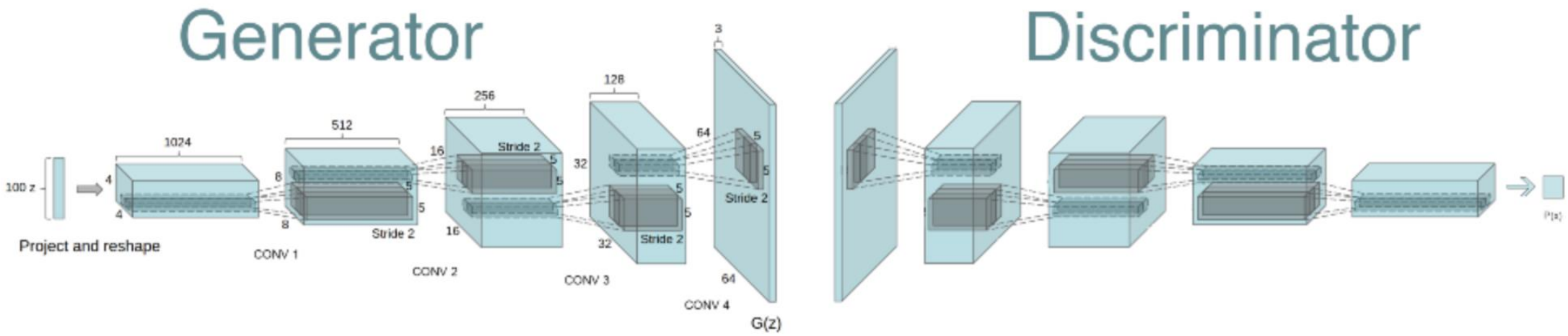


What are the types of generative adversarial networks?

Deep Convolutional GAN

Recognizing the power of convolutional neural networks (CNNs) in image processing, Deep convolutional GAN (DCGAN) integrates CNN architectures into GANs.

With DCGAN, the generator uses transposed convolutions to upscale data distribution, and the discriminator also uses convolutional layers to classify data. The DCGAN also introduces architectural guidelines to make training more stable.

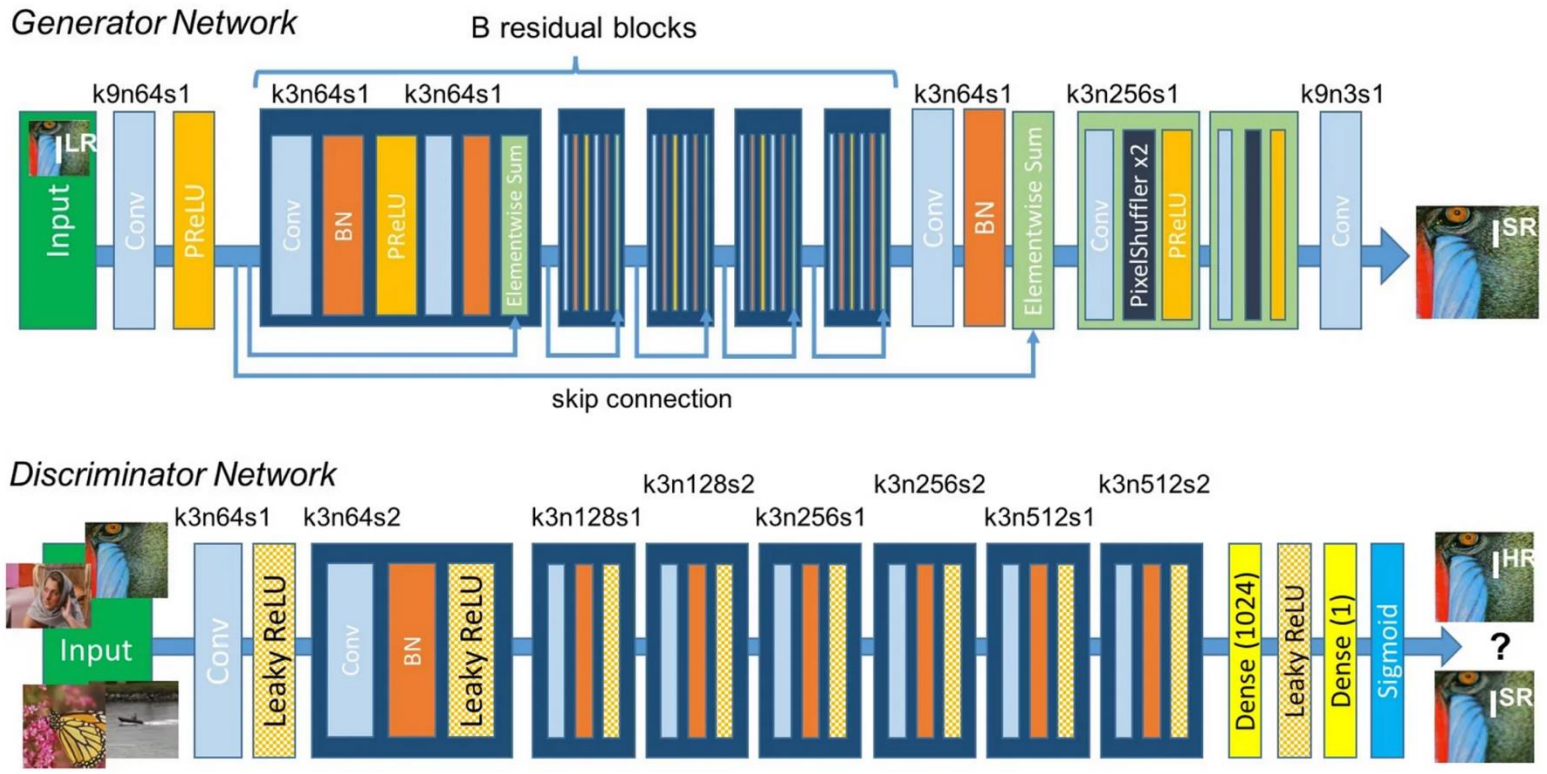


What are the types of generative adversarial networks?

Super-resolution GAN

Super-resolution GANS (SRGANs) focus on upscaling low-resolution images to high resolution. The goal is to enhance images to a higher resolution while maintaining image quality and details.

Laplacian Pyramid GANs (LAPGANs) address the challenge of generating high-resolution images by breaking down the problem into stages. They use a hierarchical approach, with multiple generators and discriminators working at different scales or resolutions of the image. The process begins with generating a low-resolution image that improves in quality over progressive GAN stages.



Please refer to Codes on Canvas


THANK YOU
Any Questions?