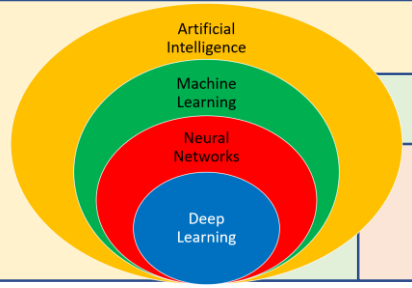


Artificial Intelligence (AI)

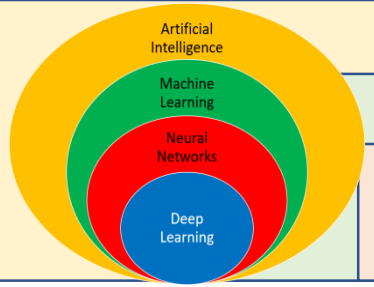
Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

Advanced Artificial Intelligence

Dr. Rastgoo



Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

Generative Models – Part 1: Introduction

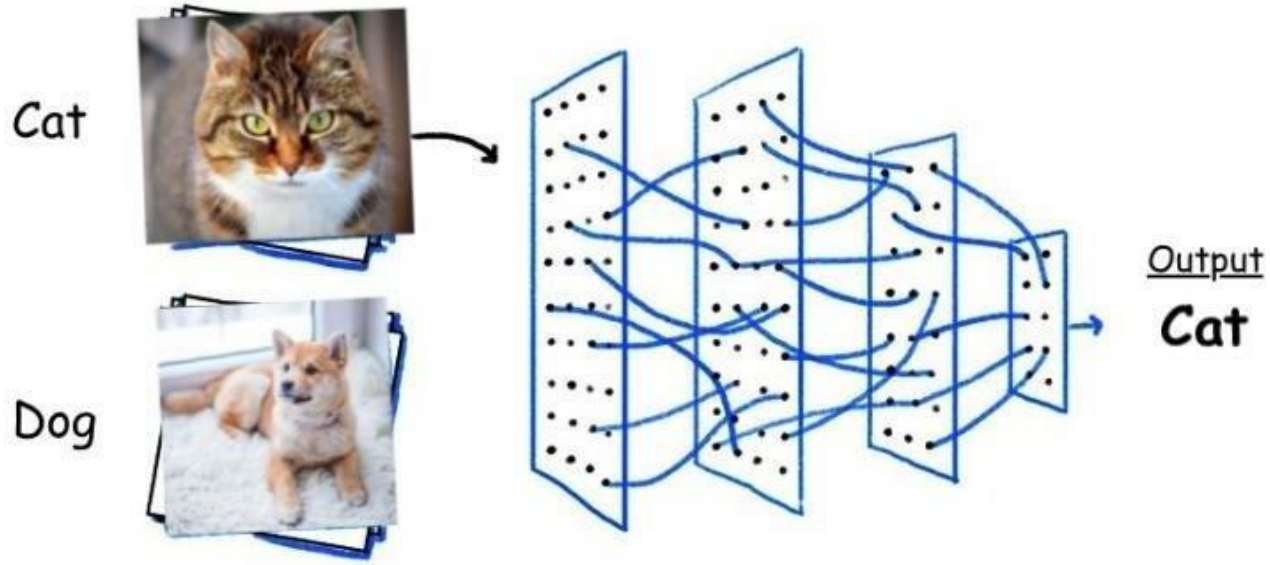
Introduction

- Before we dig deeper, I will present a basic notion of generative modeling by taking an example. Supposing we want to classify an image as a dog or a cat.

How can we solve this problem?

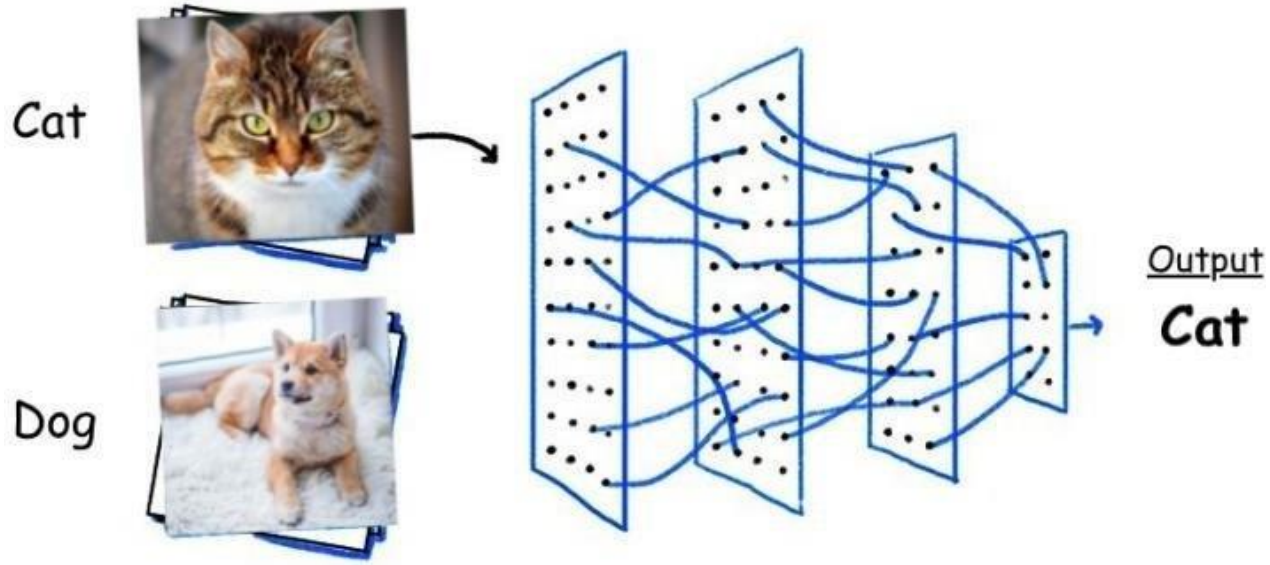
How can we solve this problem?

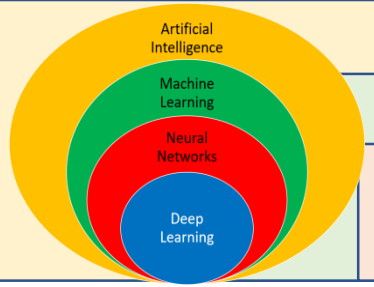
- We can solve this problem simply by feeding the image as an input to a convolutional neural network that can output the image's corresponding category.



How can we solve this problem?

- What if we want this process to be reversed? We describe what we want as an input to the model and get the image as output. This is generative modeling in its simplest and most informal form.





Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

Introduction

- Most machine learning practitioners are exposed to the classification or the regression tasks in machine learning first due to its extensive scope and straightforward approach.

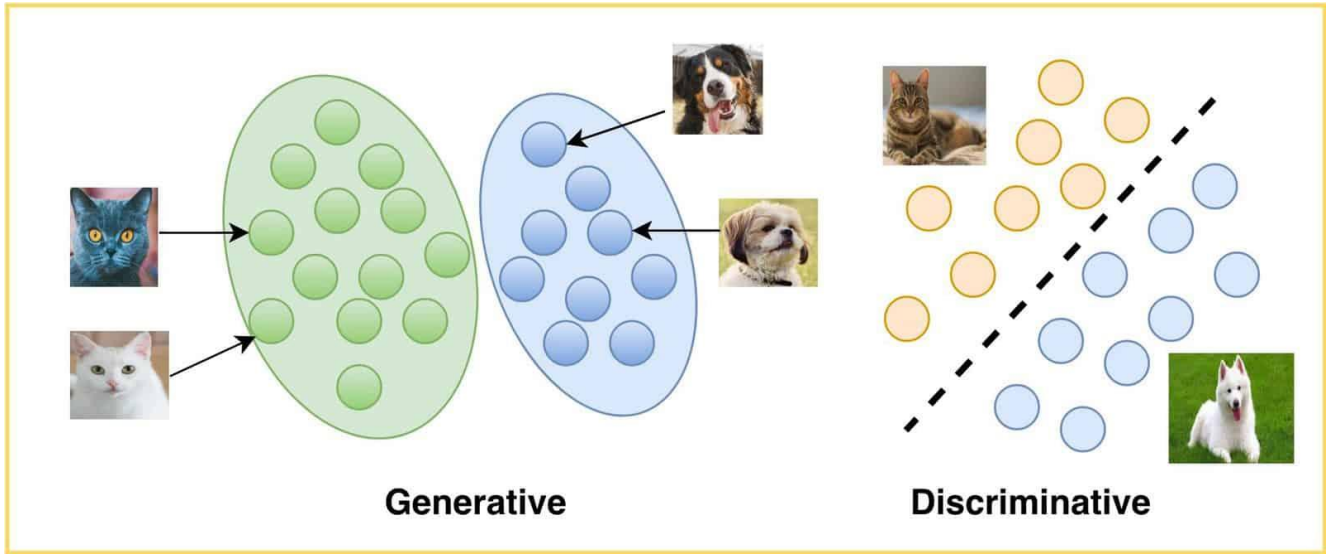
- However, the topic we will cover is not widely known amongst machine learning practitioners.

- So let's dive deep into the concepts of generative modeling and understand how it can affect the future of machine learning.

What is generative modeling?

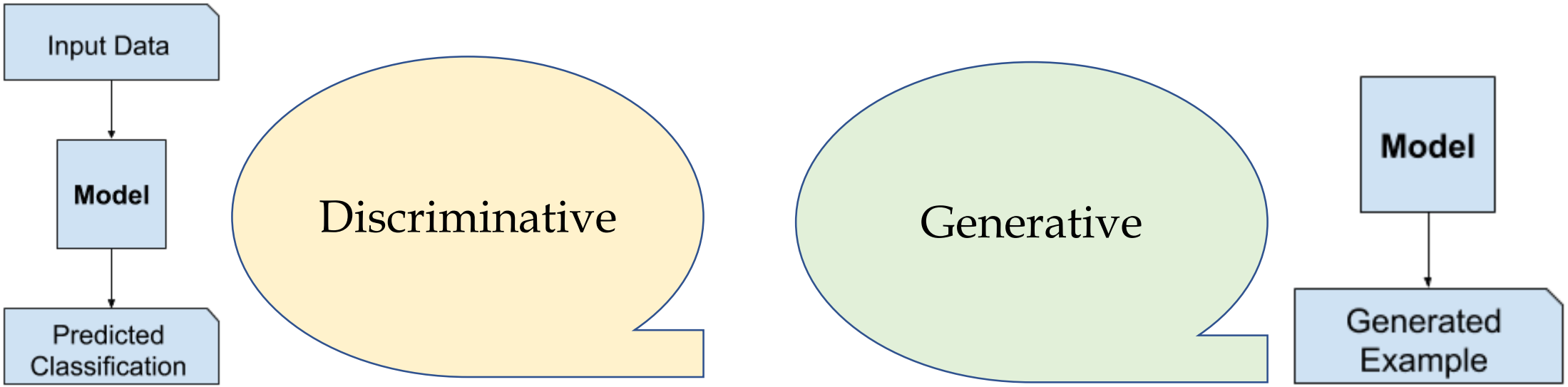
- Generative modeling is an unsupervised form of machine learning where the model learns to discover the patterns in input data.

- Using this knowledge, the model can generate new data on its own, which is relatable to the original training dataset.



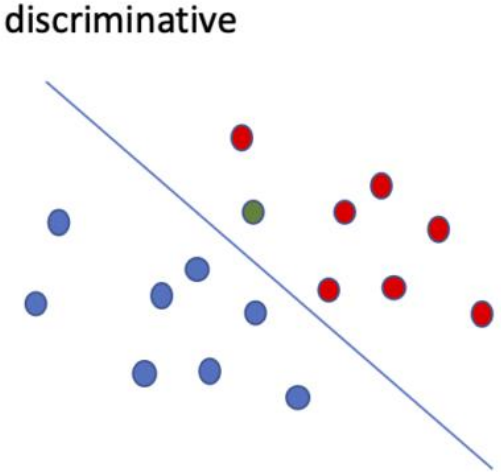
Discriminative and generative modeling

- Machine learning models can broadly be categorized into two categories:

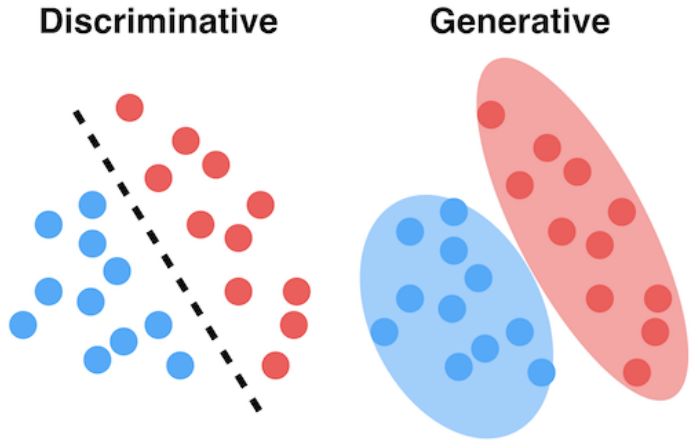


Discriminative and generative modeling

- In simple terms, as the name suggests, the discriminative model aims to discriminate between multiple data instances.
- It takes input data for training and makes predictions for unseen data.
- Most of the classification and regression techniques fall under this category.



Discriminative and generative modeling



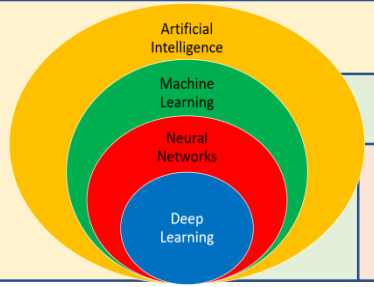
- To understand it mathematically, let's take an example of a set of data instances X and a corresponding set of labels Y.
- The discriminative model aims to capture p(Y | X), that is, the conditional probability of Y given X.
- In contrast, the generative model aims to capture joint probability p(X,Y), or just p(X) if there is no Y present.

Joint probability is the probability of two events occurring simultaneously. Conditional probability is the probability of one event occurring in the presence of a second event.

Why generative modeling?

- **The future of AI is generative, not discriminative.**

- There have been many recent advancements in the field of generative modeling since the last decade.
- The most trending topic in generative modeling is GAN. ("The most interesting idea in the last 10 years in Machine Learning")
- Another modern example of deep learning generative modeling algorithms include the Variational Autoencoder, or VAE.



Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

Why generative modeling?

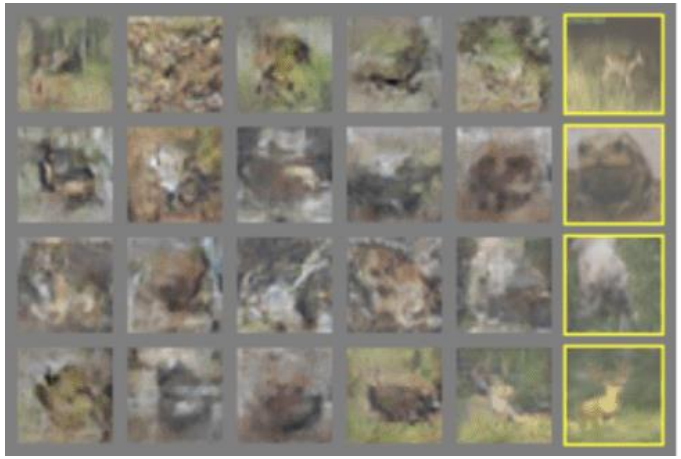
- GANs have transformed the world of deep learning and are considered the most remarkable invention in artificial intelligence.

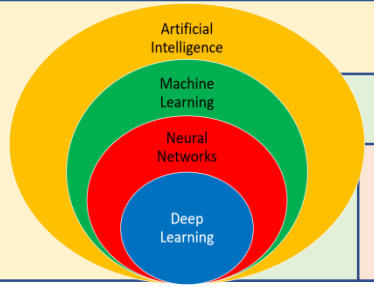
- GAN never fails to surprise from generating portraits to the whole new virtual world.

- In 2018, a French art collective named Obvious used GAN to generate a portrait sold for half a million dollars.

Why generative modeling?

- Imagine what a future would be if one can use generative modeling to make a complete 2-hour movie using existing movies, compose a piece of music, or maybe write a novel.
- The innovation will match human expertise, which is the end goal of an AI system.





Artificial Intelligence (AI)

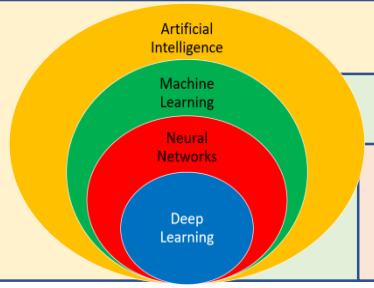
Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

Why generative modeling?

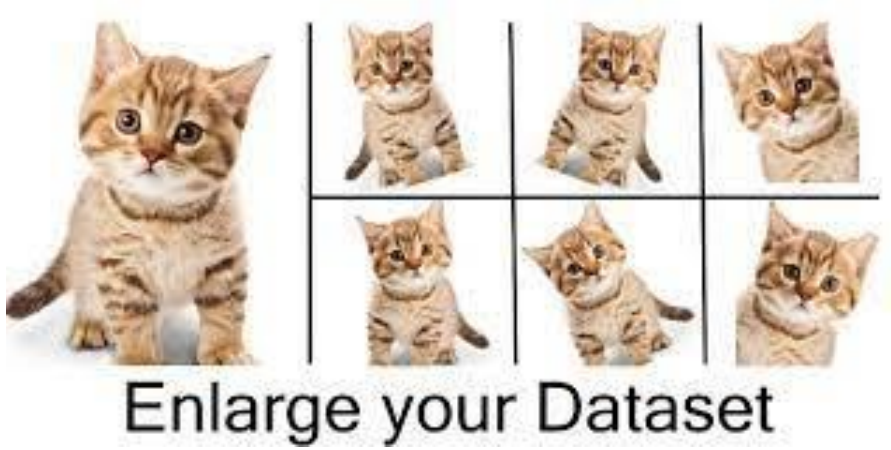
- One more important application of generative modeling is data augmentation.
- Of course, the most crucial aspect for applying any ML technique is training data.
- Still, there are certain areas where we have a lot of constraints associated with data availability.
- One such sector is the medical field.
- There are a lot of applications of machine learning in the medical field, from diagnosing disease to finding its cure. But medical datasets are harder to collect.



Why generative modeling?

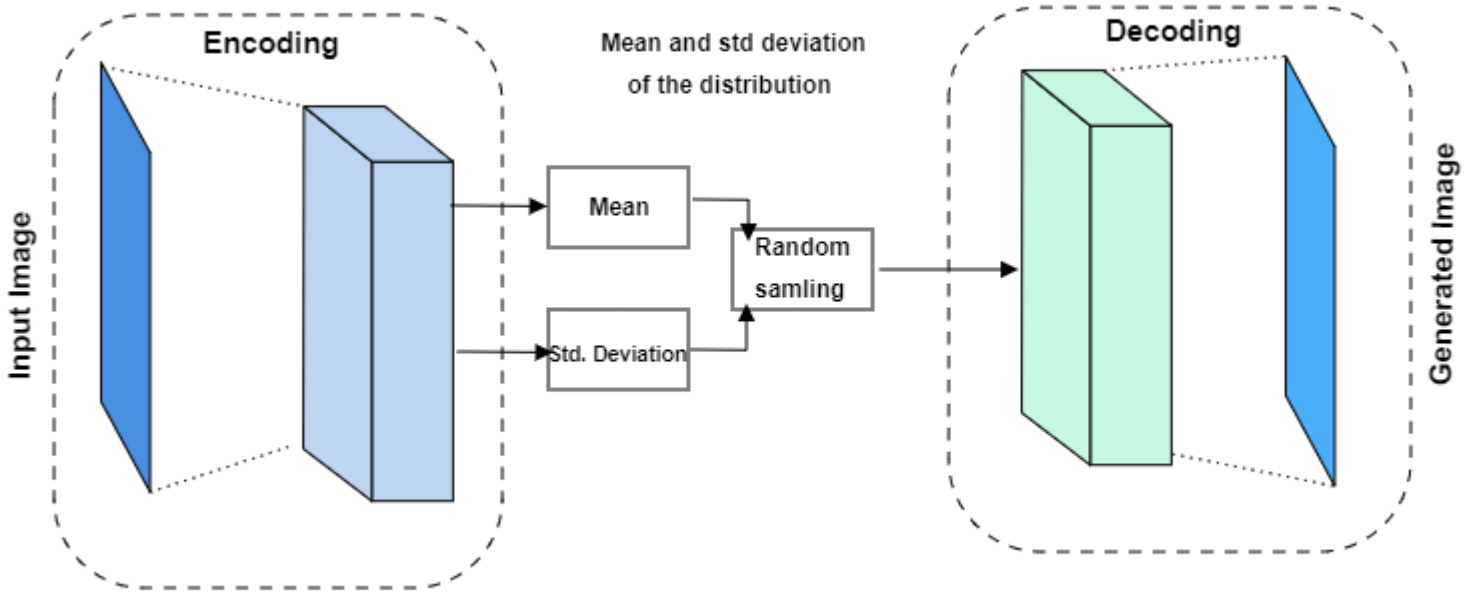
- Here generative modeling plays a vital role in producing synthetic data to enlarge the training dataset.

- Extending the training dataset by producing the synthetic data is called Data Augmentation.



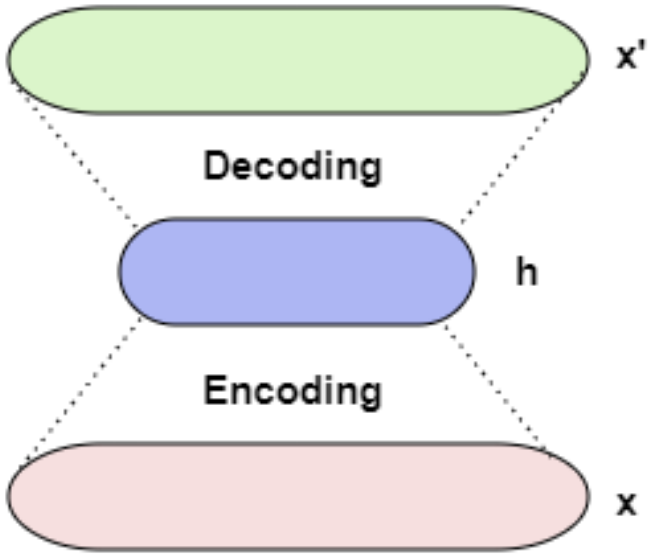
Generative Models – Part 2: Variational Auto-Encoder (VAE)

Variational Autoencoders



Variational Autoencoders: Introduction

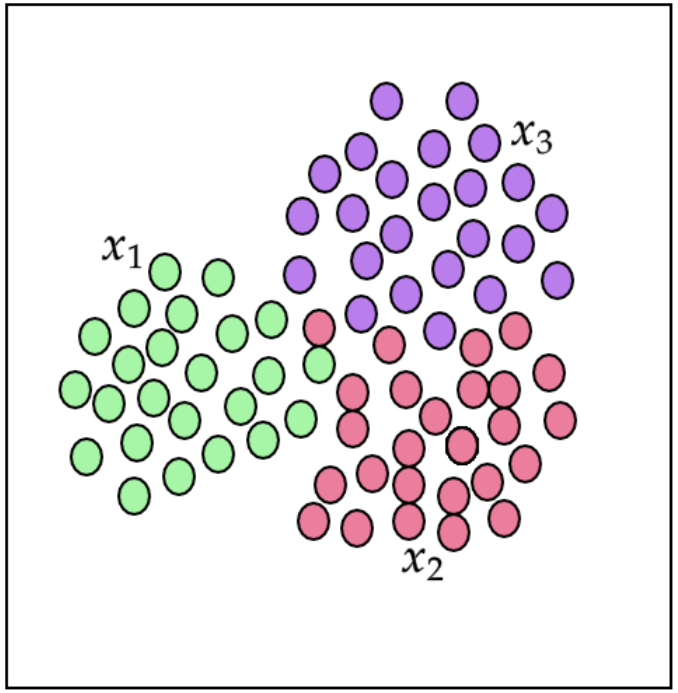
- Before we talk about the VAE(variational autoencoders), let's first get a basic notion of what an autoencoder is and how it is different from the VAE.



- An autoencoder is a type of feed-forward neural network which does the following:
- Encodes the input x into a hidden representation h by performing various operations.
- Decodes the hidden representation again into x'.

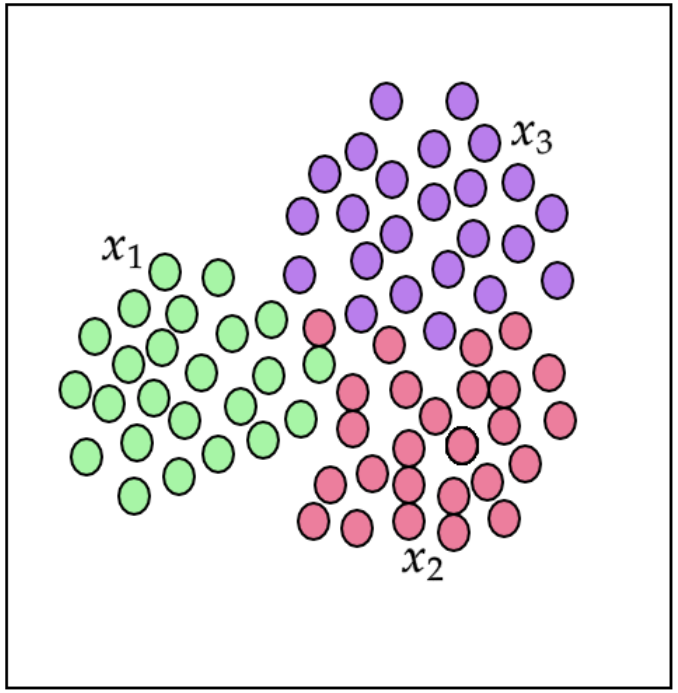
Variational Autoencoders: Introduction

- The x′ is a reconstructed version of the original input x. The autoencoder aims to minimize a specific loss function to ensure that the x′ is close to our original input x.
- Now, let us take an example of images belonging to three categories, say, x1, x2, and , x3, and we have trained an autoencoder on these examples.
- Let us assume latent representation has two dimensions, and its distribution looks like this.



Variational Autoencoders: Introduction

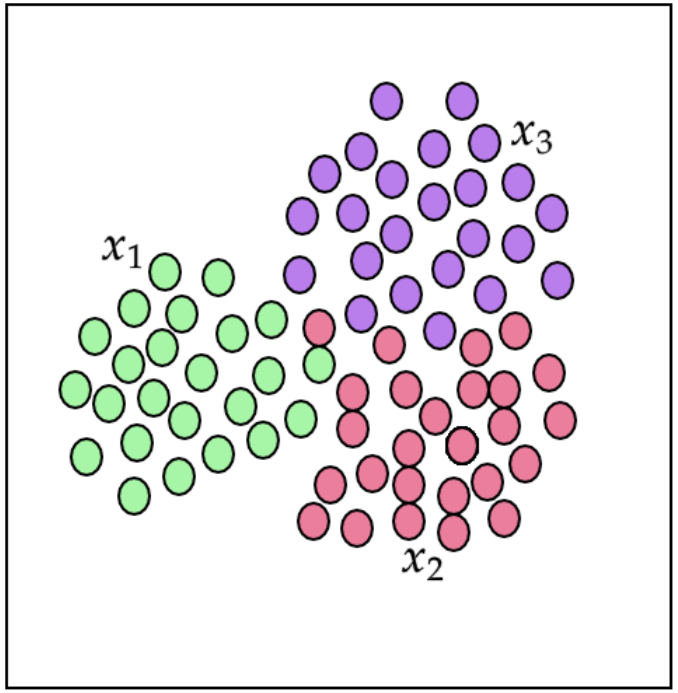
- Notice that these latent vectors have formed clusters according to their categories.
- And from here starts the role of generative modeling.
- Using one of these latent vectors, one can generate the image belonging to one of the three classes.



Variational Autoencoders: Introduction

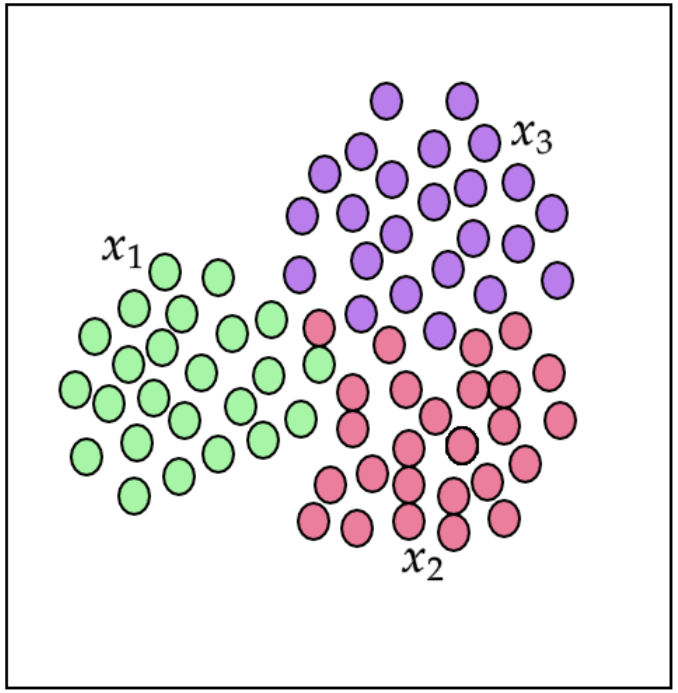
- Variational autoencoders or VAE provides us probabilistic approach to representing these latent vectors.

- In a normal autoencoder, we will try to represent each attribute of the latent state by a single variable.



Variational Autoencoders: Introduction

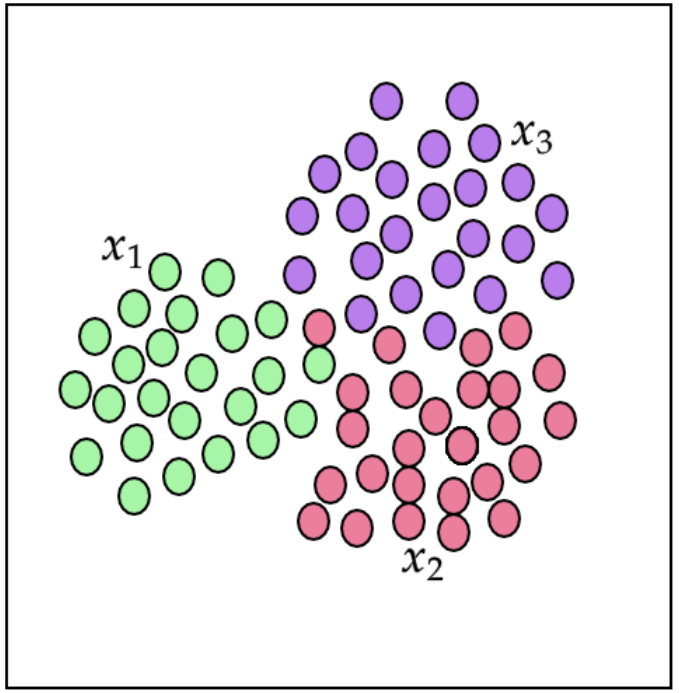
- In VAE, we will try to formulate probability distribution for each attribute of latent representation.
- In VAE, we will sample from these latent learned distributions, resulting in new images.
- Now, when we vary the sampling process randomly among these distributions, we get unique results every time, and hence we use the term variational in VAE.

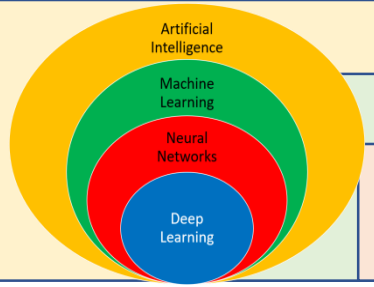


Variational Autoencoders: Introduction

- To represent the probability distribution of each class of examples in latent space, we will use (mean) and sigma (standard deviation).

- We will learn and by backpropagating through the network.





Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

Implementation

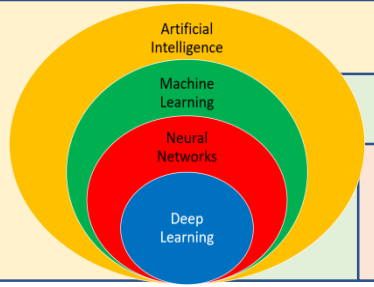
```
#Importing essential libraries to be used
import tensorflow as tf

import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras import layers,datasets,metrics
from tensorflow.keras.models import Model
from tensorflow.keras import backend as K
```

Implementation

- The datasets module from tf.keras provides a few toy datasets, one of which is MNIST.
- Therefore, we will load the standard MNIST dataset.
- Further, we will be dividing each pixel value by 255 to scale the pixel values in the range of 0 and 1.
- By doing this, training the model will be a lot more feasible.



Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

Implementation

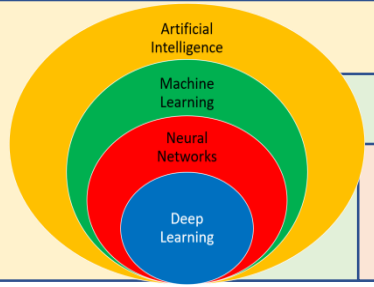
```
(x_train, y_train), (x_test, y_test) = datasets.mnist.load_data()

x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))
```

Implementation

- As VAE is a generative model, self-supervised to be specific, we will not need y labels.

- Further, let's visualize the training data using matplotlib.



Artificial Intelligence (AI)

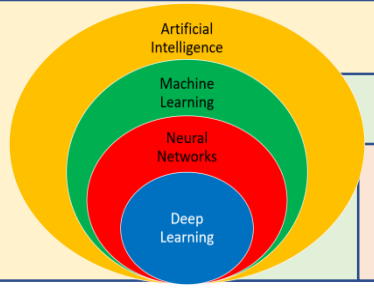
Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

```
n=10
figure = np.zeros((28*10, 28*10))
temp=0
for i in range(n):
  for j in range(n):
    data = x_train[temp].reshape(28,28)
    figure[i*28 : (i + 1)*28,
           j*28 : (j + 1)*28] = data
    temp+=1

plt.figure(figsize=(10, 10))
plt.imshow(figure, cmap='Greys_r')
plt.axis('off')
plt.show()
```



Artificial Intelligence (AI)

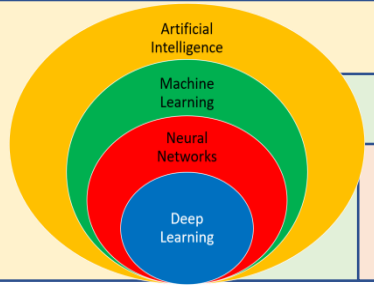
Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

Implementation

- Now that we have our data ready, let's define a sample method that we will use for sampling.
- We give mean and standard deviation as input to this method and get a sample from normal distribution as output.
- Note that mean and standard deviation are the learnable parameters.



Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

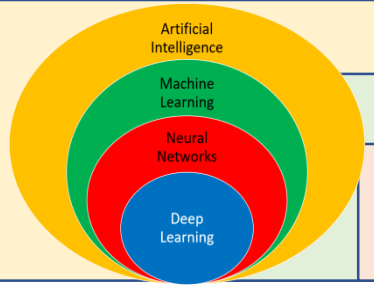
```
def sample(args):
  mean,std = args
  epsilon = K.random_normal(shape=(K.shape(mean)[0], K.int_shape(mean)[1]))
  return mean + K.exp(std/2)*epsilon
```

Implementation

- Finally, it's time to define the model.
- Again, we will be using functional API from Keras.
- So, first, we will define the encoder model, then the decoder model, and finally, using these two models, we will define VAE.

Implementation

- The input to the encoder model will be the image.
- The encoder model, in our case, has one hidden layer of dimension 128.
- The output layer of the encoder has mean and std deviation, which we will be using a vector of dimension 4 to represent.
- And, finally, we will use a sample method for sampling.



Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

```
#Encoder model
input_layer = layers.Input(shape=(784,), name='input_layer’)
hidden_layer = layers.Dense(128, activation='relu', name='encoding')(input_layer)
latent_mean = layers.Dense(4, name='mean')(hidden_layer)
latent_std = layers.Dense(4, name='var')(hidden_layer)
latent_sample = layers.Lambda(sample, output_shape=(4,))([latent_mean, latent_std])
encoder_model = Model(input_layer, [latent_mean, latent_std, latent_sample],name='encoder_model')
```

Implementation

- Now we will move to the decoder part.
- Remember, we will feed a sampled vector using μ and σ, which have a dimension of 4 as an input to the decoder.
- Again, the decoder is also a three-layered network with a hidden layer with dimension 128 and an output layer that returns the reconstructed version of the input image as an output.
- We use sigmoid as an activation function for the final layer because pixel values are between 0 and 1.

Implementation

```
#Decoder model
decoder_input = layers.Input(shape=(4, ), name='decoder_input’)
decoder_hidden = layers.Dense(128, activation='relu', name='decoding')(decoder_input)
decoder_output = layers.Dense(784, activation='sigmoid', name='decoded_output')(decoder_hidden)
decoder_model = Model(decoder_input, decoder_output, name='decoder_model')
```

Implementation

- Once we have our encoder and decoder models ready, we will define the VAE.
- The VAE will take an image as an input and return generated image from the decoder.
- The decoder takes a sample from the distribution, which is present at the second index of encoder output

```
#VAE model
vae = Model(input_layer, decoder_model(encoder_model(input_layer)[2]))
```

Implementation

- We will define the custom loss function to train the VAE model.
- First, reconstruction loss is binary cross-entropy loss computed using each input image pixel and generated image.
- It is binary because we have already scaled the images between 0 and 1.
- The second loss function is KL divergence loss.
- KL divergence loss gives us a measure of how one probability distribution is different from the other.

Implementation

- We aim to maximize the KL divergence loss.
- Hence, we introduce a negative sign before it.
- Now our total loss function will be the mean of both the losses.
- And we will aim to minimize this loss function by backpropagating through the VAE model.

Implementation

```
def vae_loss(x, z_decoded, latent_mean=latent_mean, latent_std=latent_std):
    x = K.flatten(x)
    z_decoded = K.flatten(z_decoded)
    recon_loss = metrics.binary_crossentropy(x, z_decoded)
    kl_loss = -1e-4*K.mean(1+latent_std-K.square(latent_mean)-K.exp(latent_std), axis=-1)
    return K.mean(recon_loss + kl_loss)
```

Now, as an optimizer, we are going to use an optimizer.

Implementation

```
vae.compile(optimizer='rmsprop', loss=vae_loss)
vae.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     [(None, 784)]             0
_________________________________________________________________
encoder_model (Model)        [(None, 4), (None, 4), (N 101512
_________________________________________________________________
decoder_model (Model)        (None, 784)               101776
=================================================================
Total params: 203,288
Trainable params: 203,288
Non-trainable params: 0
_________________________________________________________________
```

Implementation

- Finally, we are ready to train our model.

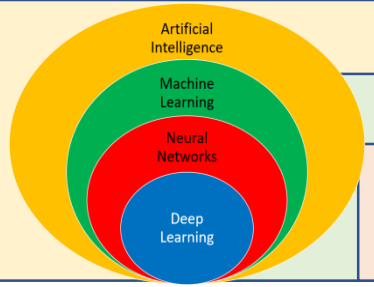
```
vae.fit(x_train, x_train, shuffle=True, epochs=50, batch_size=100)
```

- Once the training is finished, we are ready to generate the data.
- First, we will take a random vector of 4 dimensions over the normal distribution and feed it to the decoder model.

- Here I am plotting 100 such images using the matplotlib library.

```
n=10
data = np.random.normal(size=(n*n,4))
figure = np.zeros((28*10, 28*10))
temp=0
for i in range(n):
    for j in range(n):
        generated_data = decoder_model.predict(np.expand_dims(data[temp],axis=0))
        generated_data = generated_data.reshape(28,28)
        figure[i*28 : (i + 1)*28, j*28 : (j + 1)*28] = generated_data
        temp+=1

plt.figure(figsize=(10, 10))
plt.imshow(figure, cmap='Greys_r')
plt.axis('off')
plt.show()
```

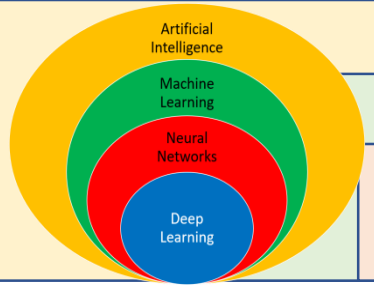


Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)



Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)