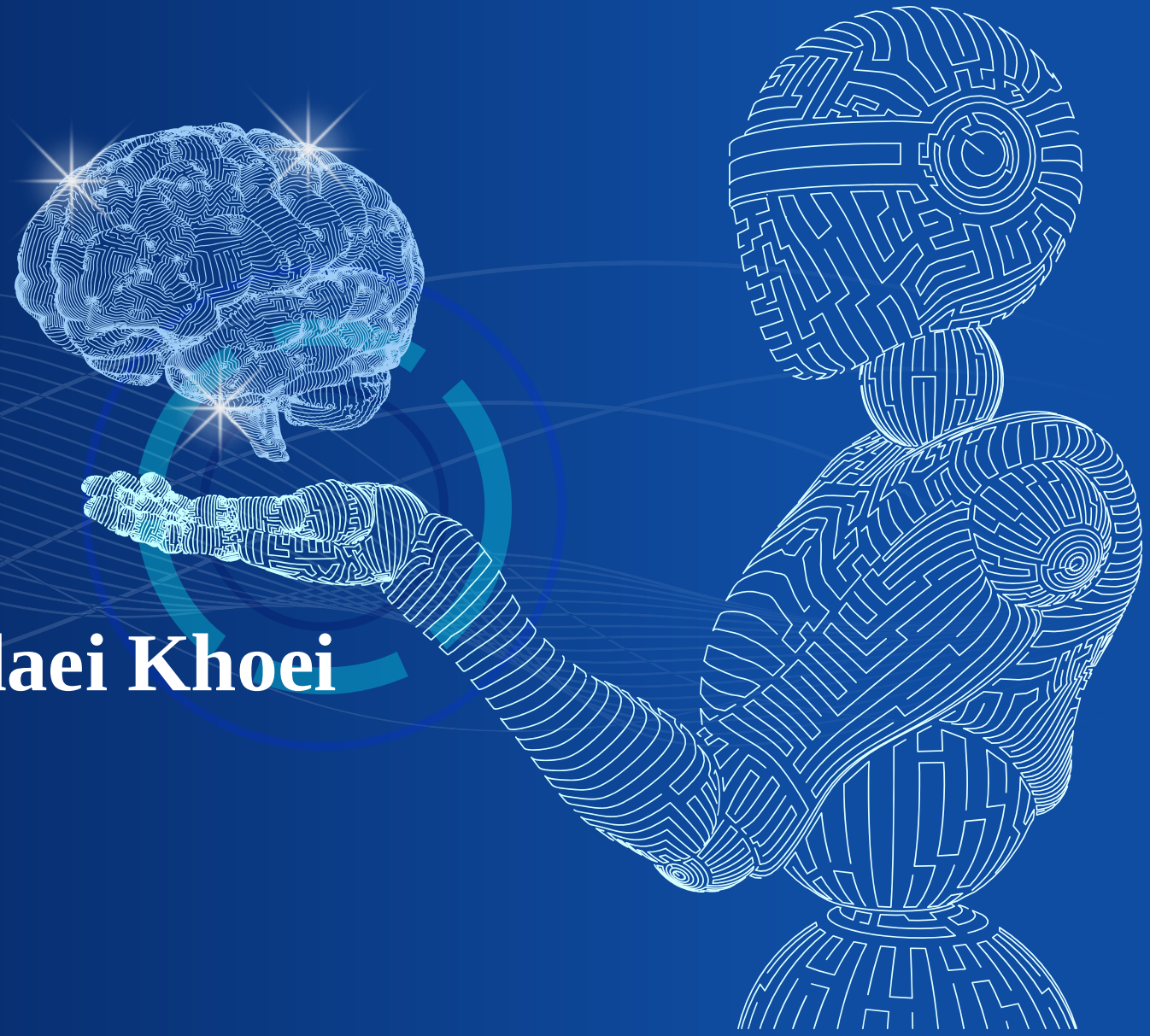
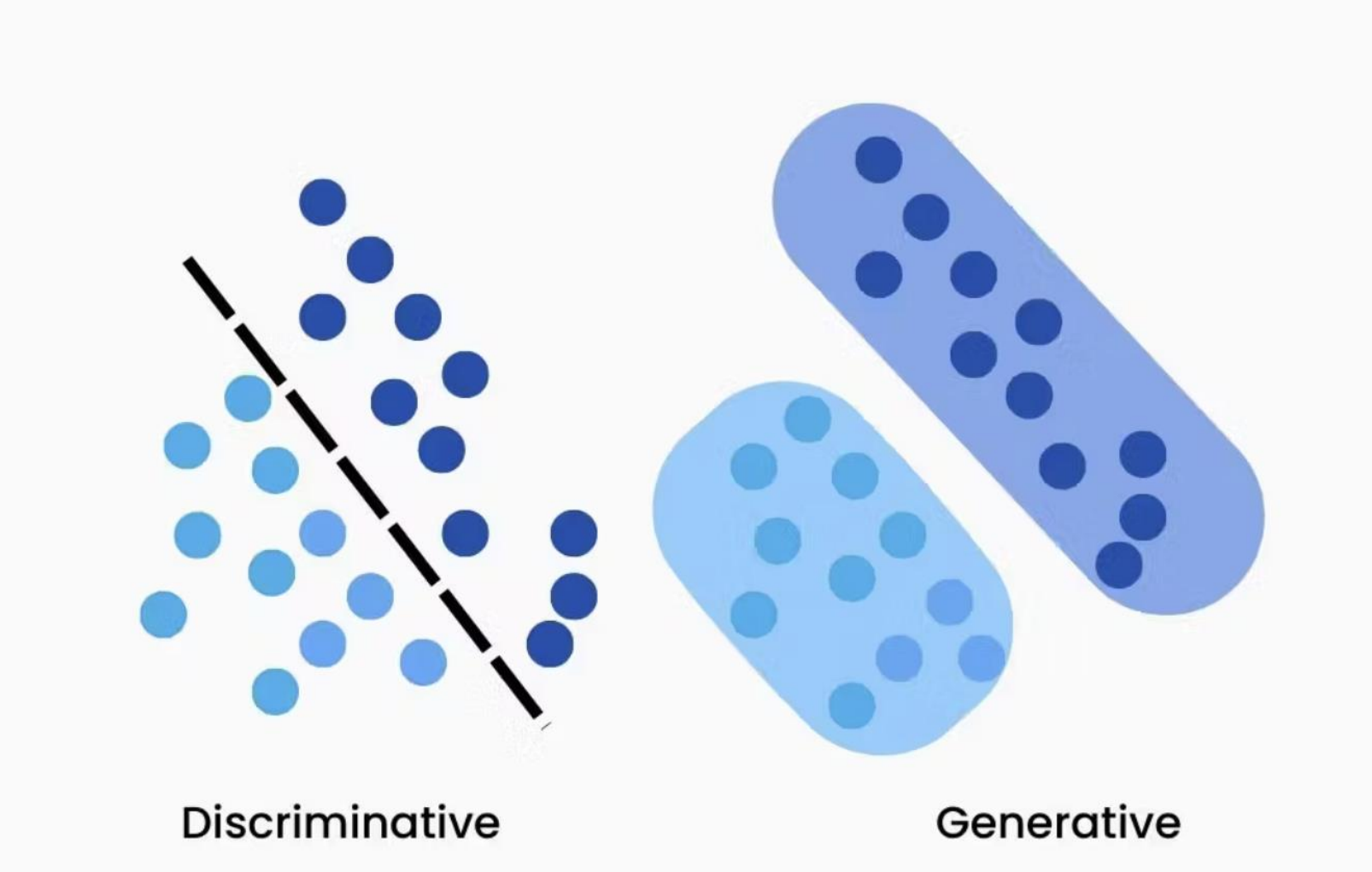
CS 7150: Deep Learning



Presented by: Dr. Tala Talaei Khoei

Discriminative vs. Generative Models

- Generative and discriminative models are two fundamental classes of models in the world of AI.

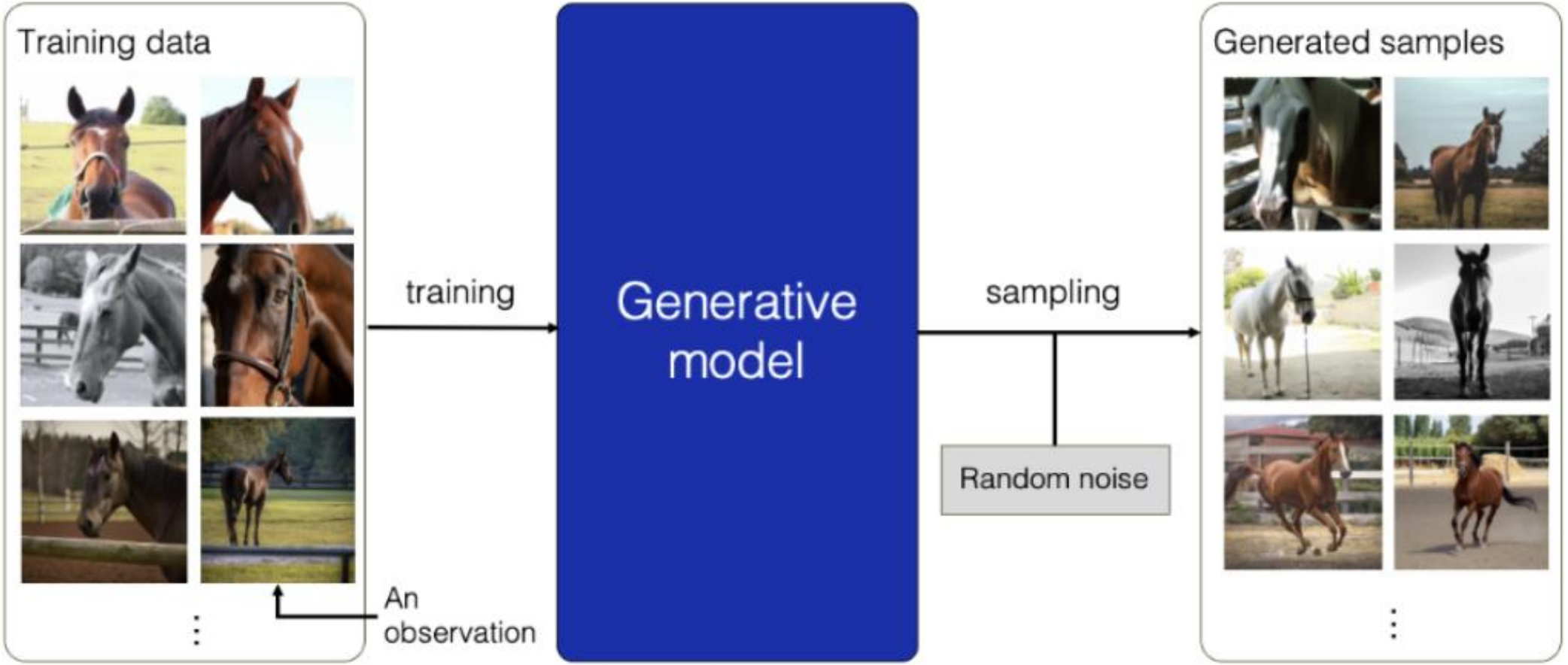


Generative Models

- Generative models dive deep into the underlying distribution of the input data. By understanding the joint probability distribution of the input data and labels, they gain the remarkable ability to generate new samples that closely resemble the training data. These models provide a window into the intricate patterns and structures within the data, allowing for creative data synthesis and augmentation.
- As the name suggests, generative models can be used to generate new data points. These models are usually used in unsupervised machine learning problems. Generative models go in-depth to model the actual data distribution and learn the different data points, rather than model just the decision boundary between classes.
- These models are prone to outliers, which is their only drawback when compared to discriminative models. The mathematics behind generative models is quite intuitive too. The method is not direct like in the case of discriminative models. To calculate *P(Y|X)*, they first estimate the prior probability *P(Y)* and the likelihood probability *P(X|Y)* from the data provided.

$$posterior = \frac{prior \times likelihood}{evidence} \Rightarrow P(Y|X) = \frac{P(Y) \cdot P(X|Y)}{P(X)}$$

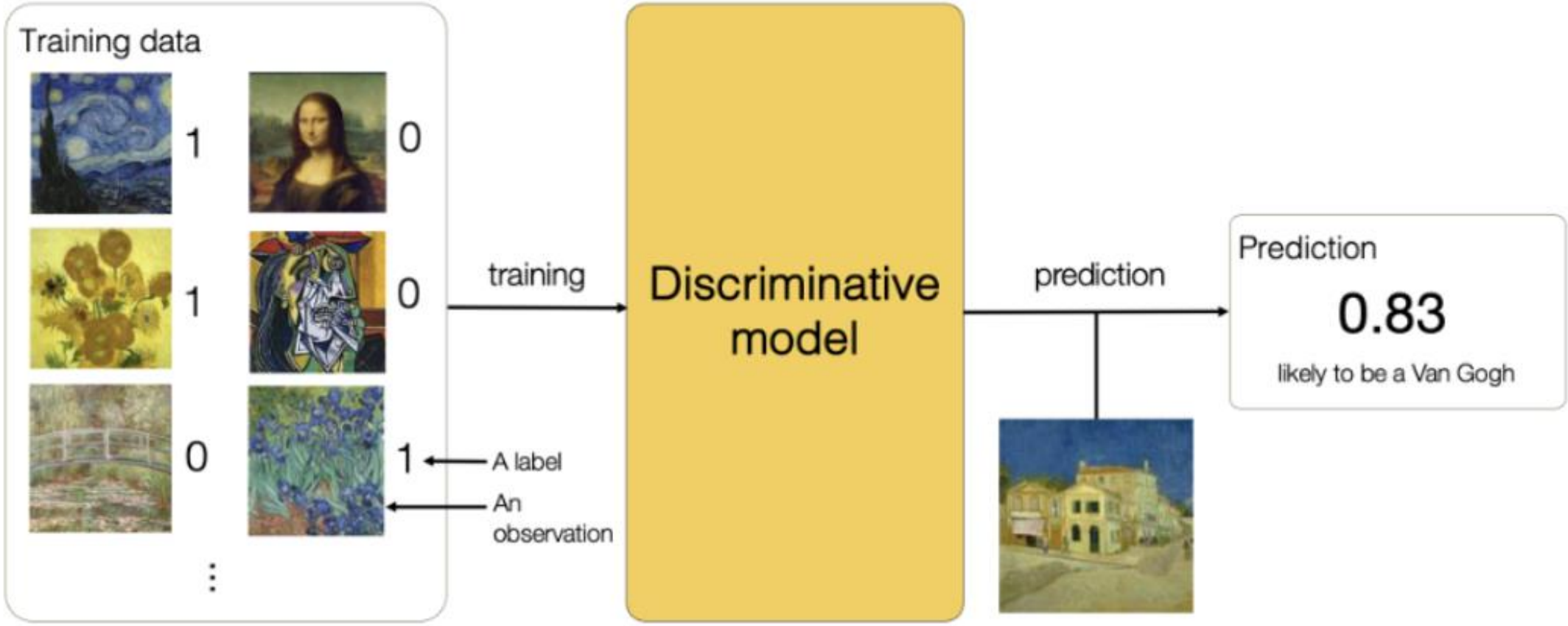
1. ***Bayesian network*:** Also known as Bayes' network, this model uses a directed acyclic graph (DAG) to draw Bayesian inferences over a set of random variables to calculate probabilities. It has many applications like prediction, anomaly detection, time series prediction, etc.
2. ***Autoregressive model:*** Mainly used for time series modeling, it finds a correlation between past behaviors to predict future behaviors.
3. ***Generative adversarial network (GAN)***: It's based on deep learning technology and uses two submodels. The generator model trains and generates new data points and the discriminative model classifies these 'generated' data points into real or fake.


Training data
training
Generative model
sampling
Random noise
Generated samples
An observation

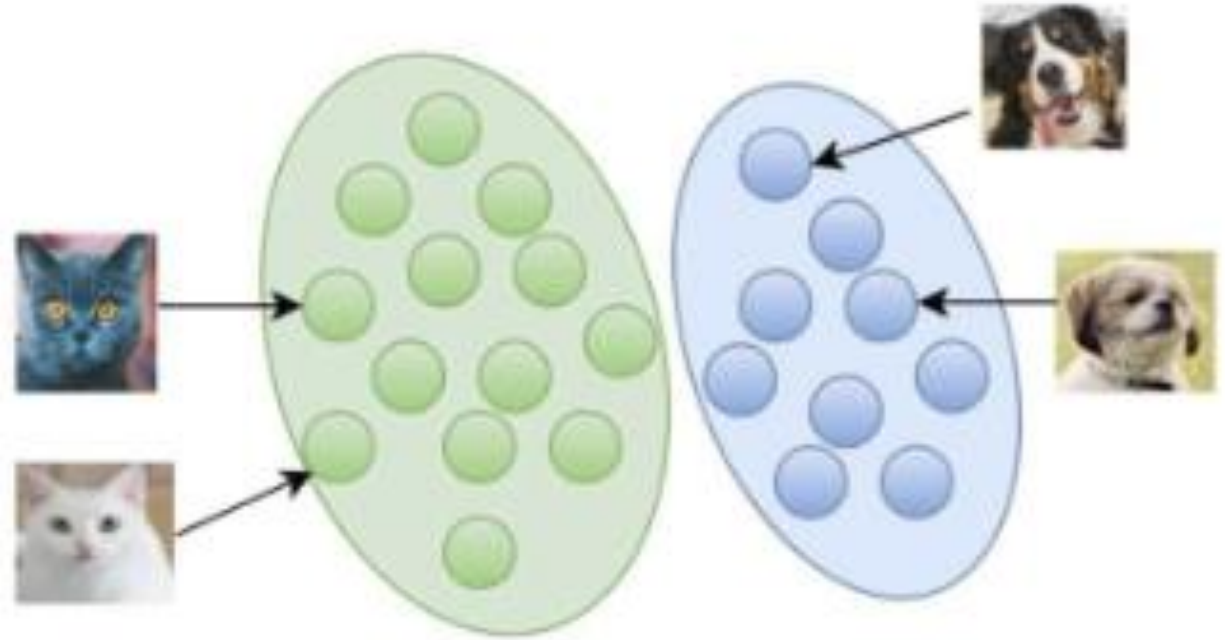
Discriminative Models

Discriminative models, in contrast, are focused on learning the decision boundary that separates different classes within the input data. Instead of modeling the entire data distribution, they concentrate on capturing the conditional probability distribution of labels given the input data. Discriminative models excel at classification tasks by effectively distinguishing between different classes. The majority of discriminative models, aka conditional models, are used for supervised machine learning. They do what they 'literally' say, separating the data points into different classes and learning the boundaries using probability estimates and maximum likelihood. Outliers have little to no effect on these models. They are a better choice than generative models, but this leads to misclassification problems which can be a major drawback.

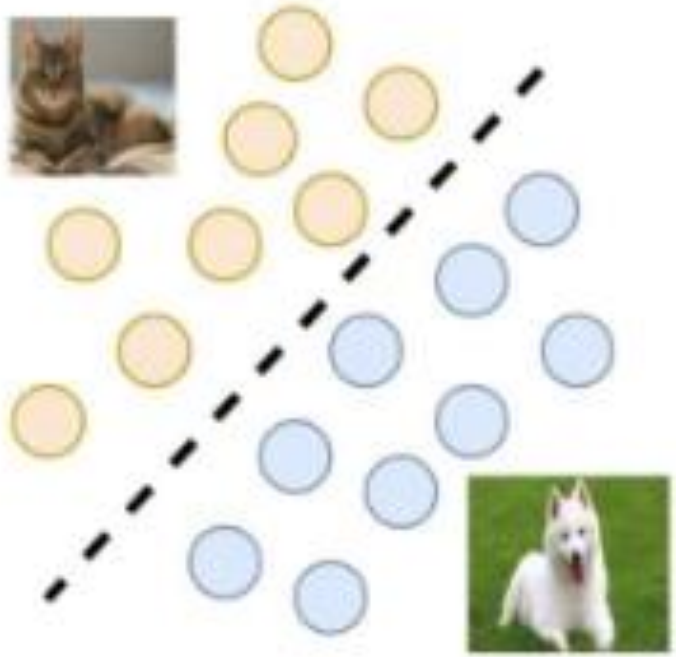
1. ***Logistic regression:*** Logistic regression can be considered the linear regression of classification models. The main idea behind both the algorithms is similar, but while linear regression is used for predicting a continuous dependent variable, logistic regression is used to differentiate between two or more classes.
2. ***Support vector machines:*** This is a powerful learning algorithm with applications in both regression and classification scenarios. An n-dimensional space containing the data points is divided into classes by decision boundaries using support vectors. The best boundary is called a hyperplane.
3. ***Decision trees:*** A graphical tree-like model is used to map decisions and their probable outcomes. It could be thought of as a robust version of If-else statements.


Training data
1
0
1
0
0
1
A label
An observation
training
Discriminative model
prediction
Prediction
0.83
likely to be a Van Gogh

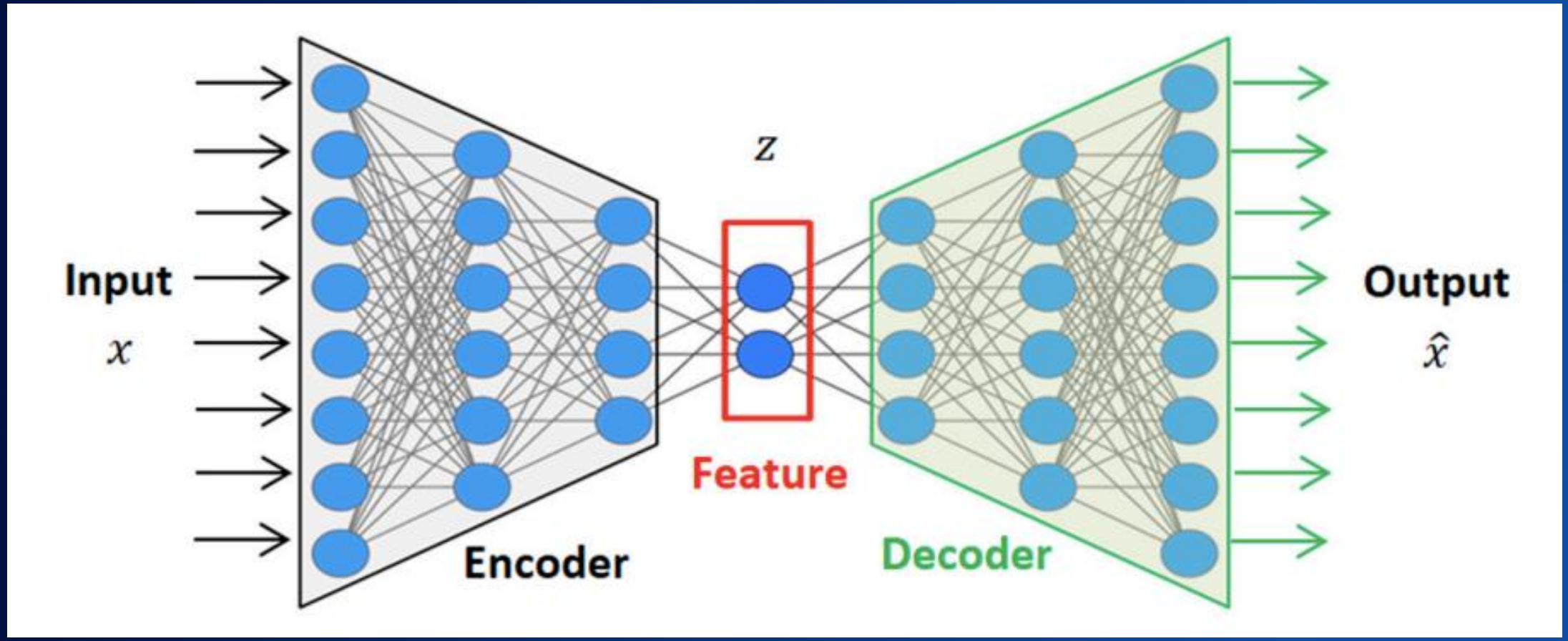
Params	GENERATIVE ALGORITHM	DISCRIMINATIVE ALGORITHM
Objective	Models the joint probability distribution of the input characteristics and labels in a generative manner.	Modeling the conditional probability distribution of labels given input attributes is the main focus of discriminative modeling.
Methodology	Creates new samples by learning the distribution of the underlying data.	Acquires the threshold of judgment that distinguishes various classes or categories.
Application	Text generation and image synthesis are examples of generative tasks.	Used in activities like sentiment analysis and image categorization.
Strength	Effective with inadequate or missing data	Excellent at distinguishing between classes or categories is discrimination.
Weakness	May have trouble distinguishing different classes in large datasets.	Less useful when dealing with incomplete or missing data.



Generative

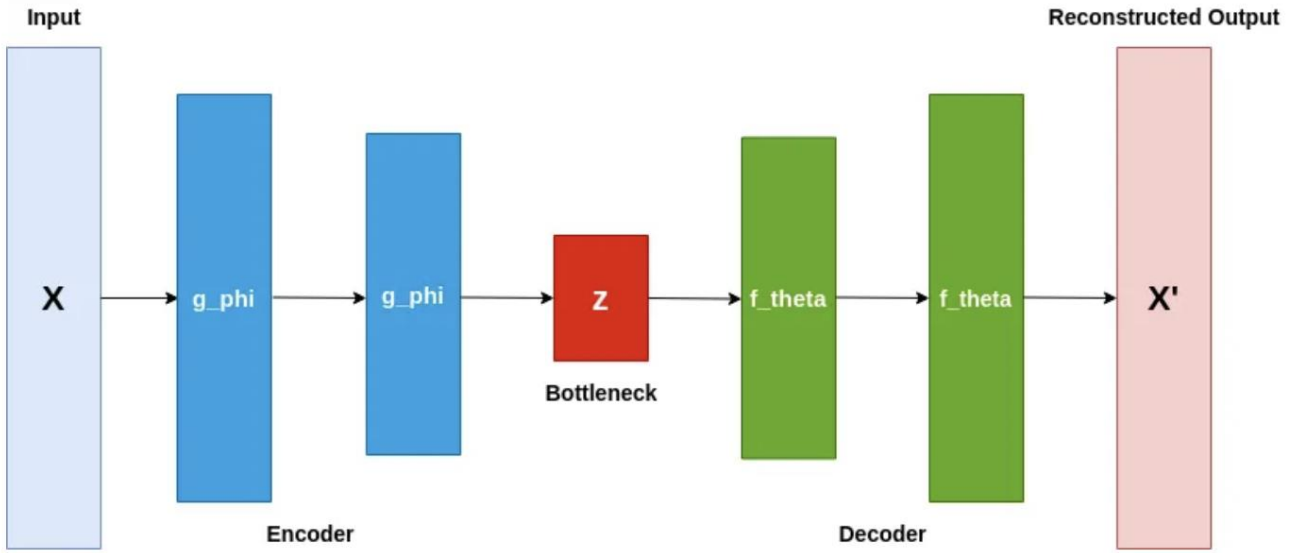


Discriminative

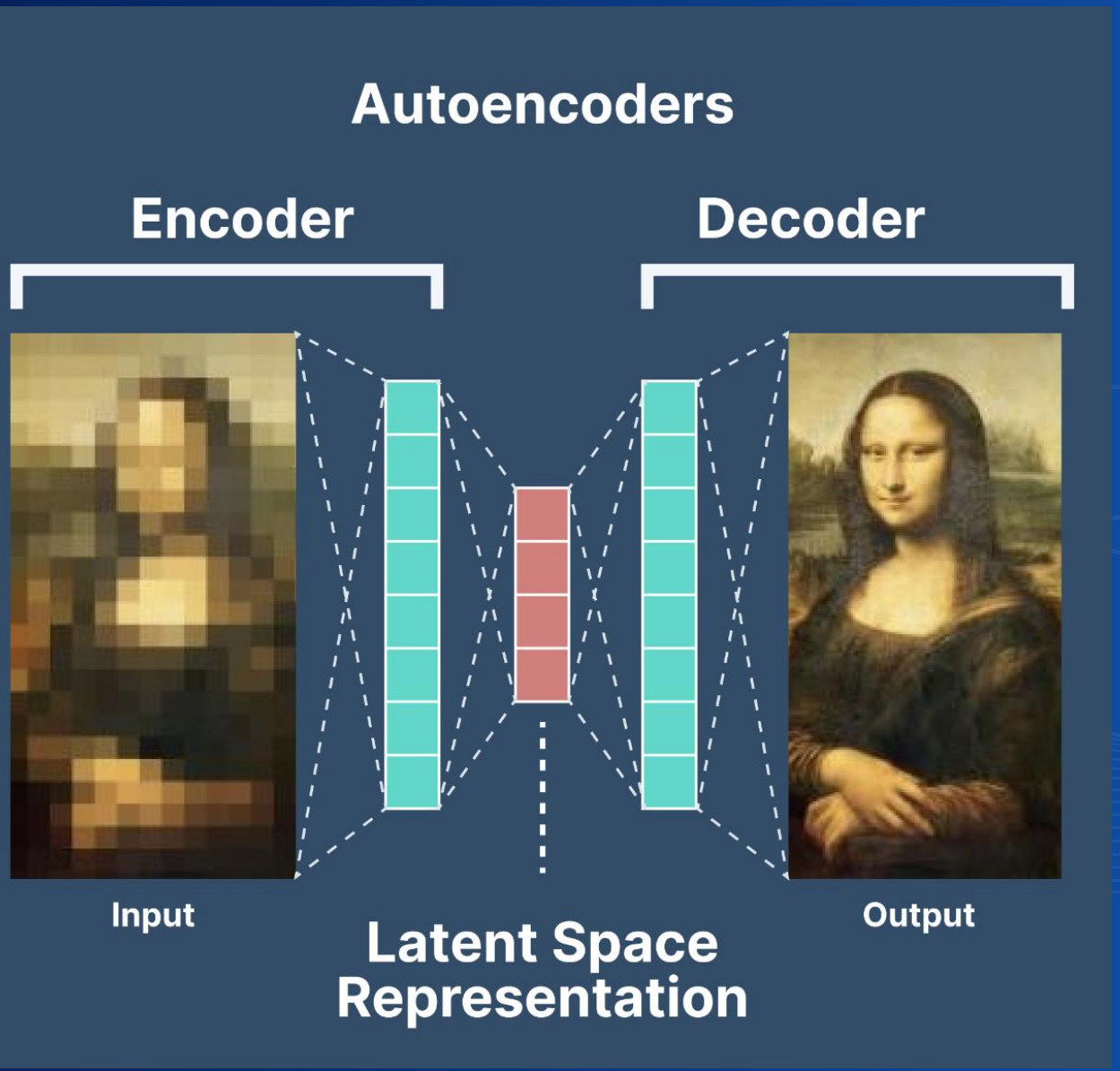

Input
x
z
Feature
Encoder
Decoder
Output
$\hat{x}$

Introduction to Auto-Encoders

- Autoencoders are neural network-based models that are used for unsupervised learning purposes to discover underlying correlations among data and represent data in a smaller dimension. The autoencoders frame unsupervised learning problems as supervised learning problems to train a neural network model. The input only is passed a the output. The input is squeezed down o a lower encoded representation using an encoder network, then a decoder network decodes the encoding to recreate back the input.
- The encoding produced by the encoder layer has a lower-dimensional representation of the data and shows several interesting complex relationships among data.



Autoencoders



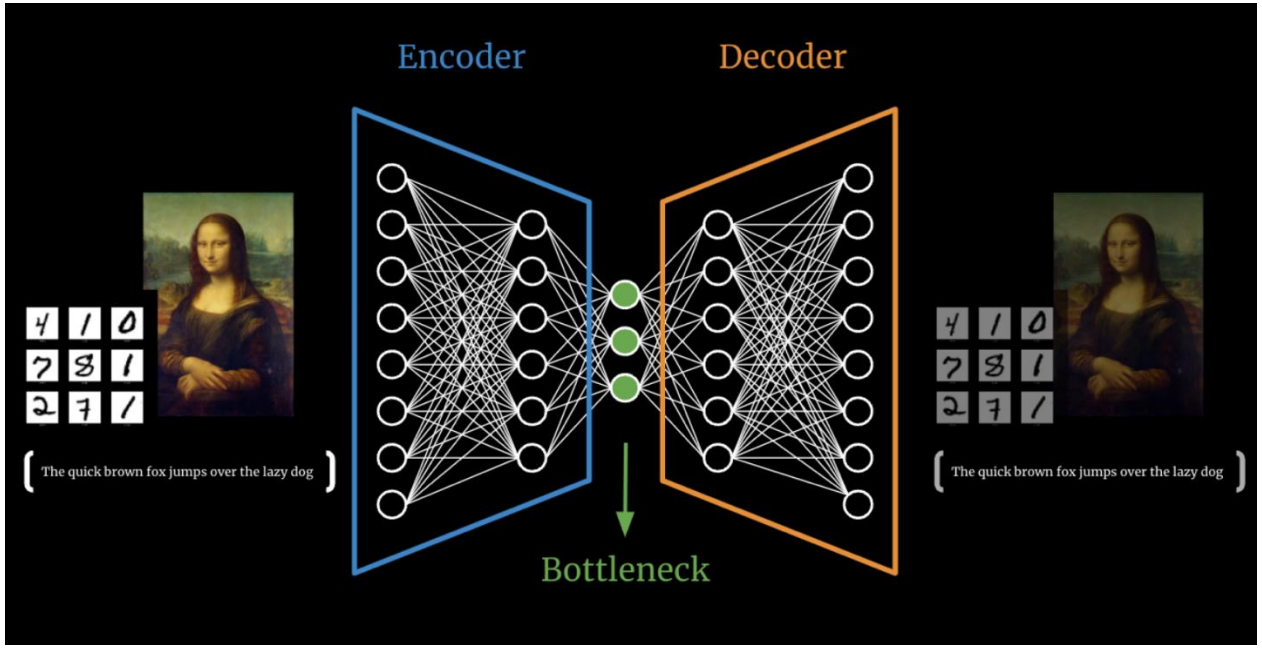
The architecture of autoencoders

Autoencoders consist of 3 parts:

1. **Encoder**: A module that compresses the train-validate-test set input data into an encoded representation that is typically several orders of magnitude smaller than the input data.

2. **Bottleneck:** A module that contains the compressed knowledge representations and is therefore the most important part of the network.

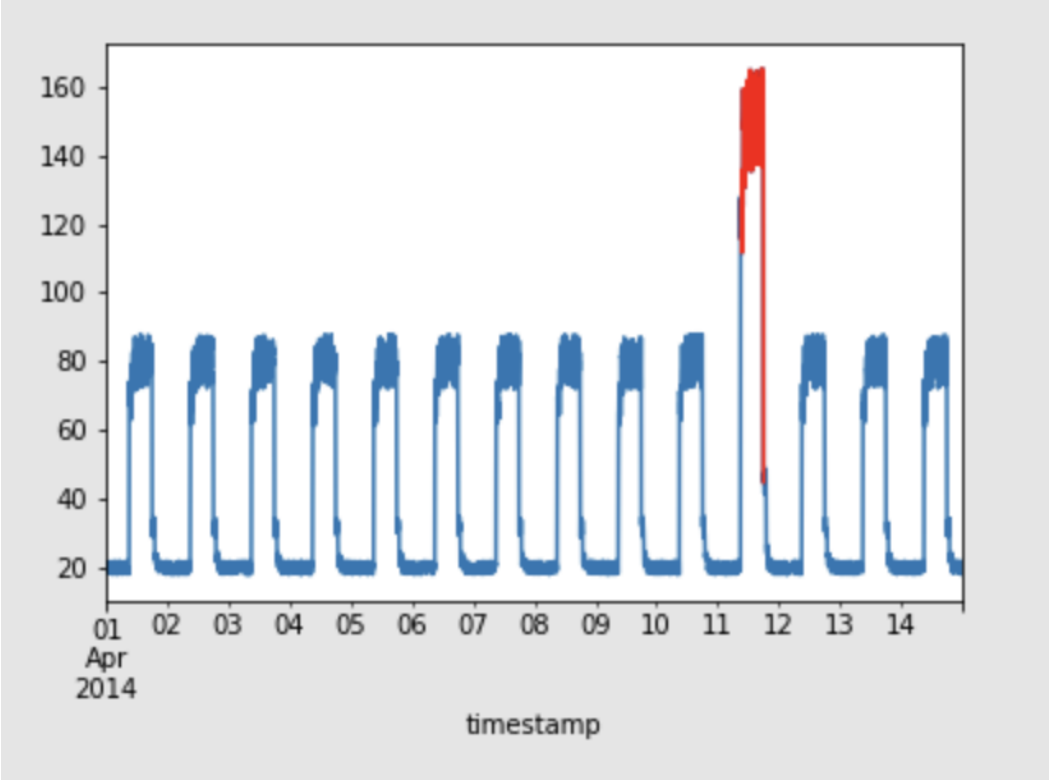
3. **Decoder:** A module that helps the network "decompress" the knowledge representations and reconstructs the data back from its encoded form. The output is then compared with a ground truth.



Applications of Auto-Encoders

Anomaly Detection

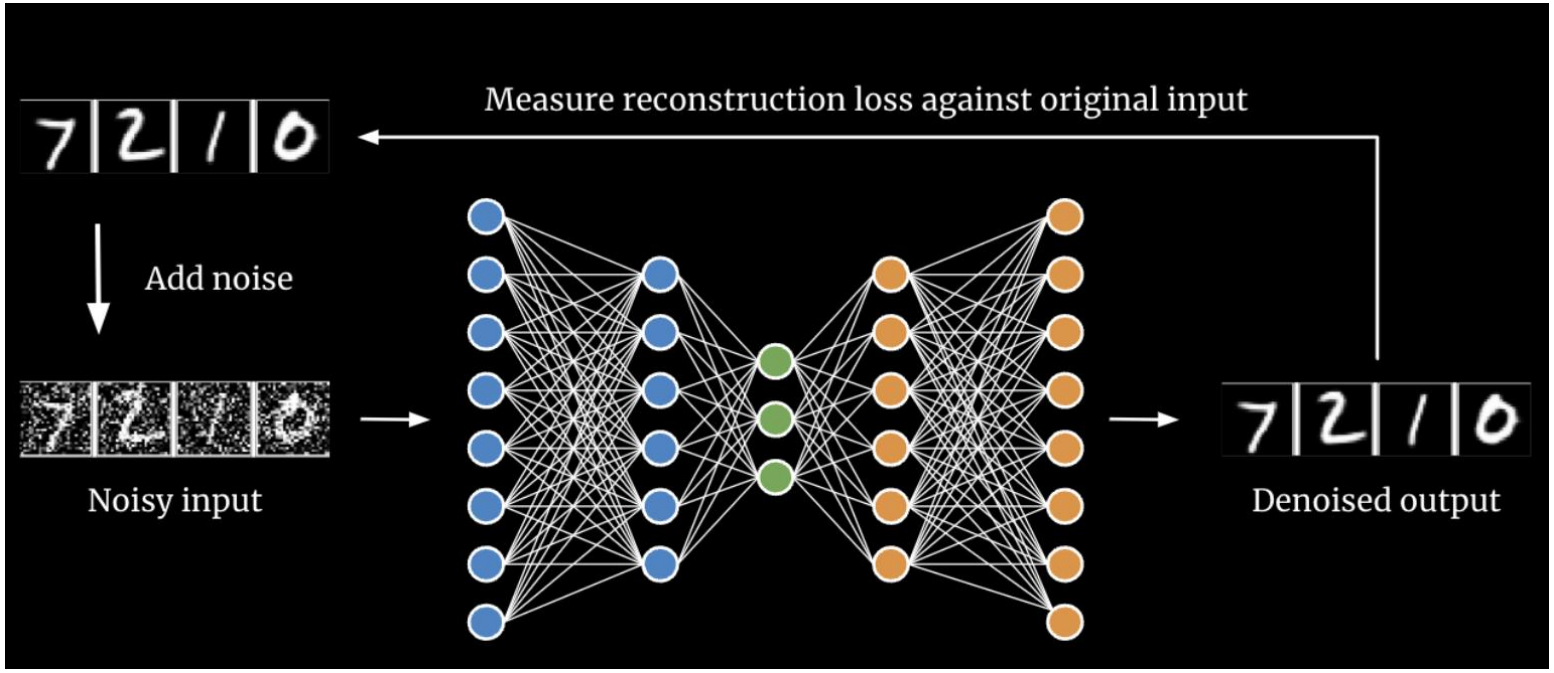
- Anomaly detection is a perfect example, where we train the model on normal instances so that if we feed it any outliers, they will be detected easily. In this case, we can use the reconstruction loss as a metric to detect outliers.



Applications of Auto-Encoders

Image Denoising

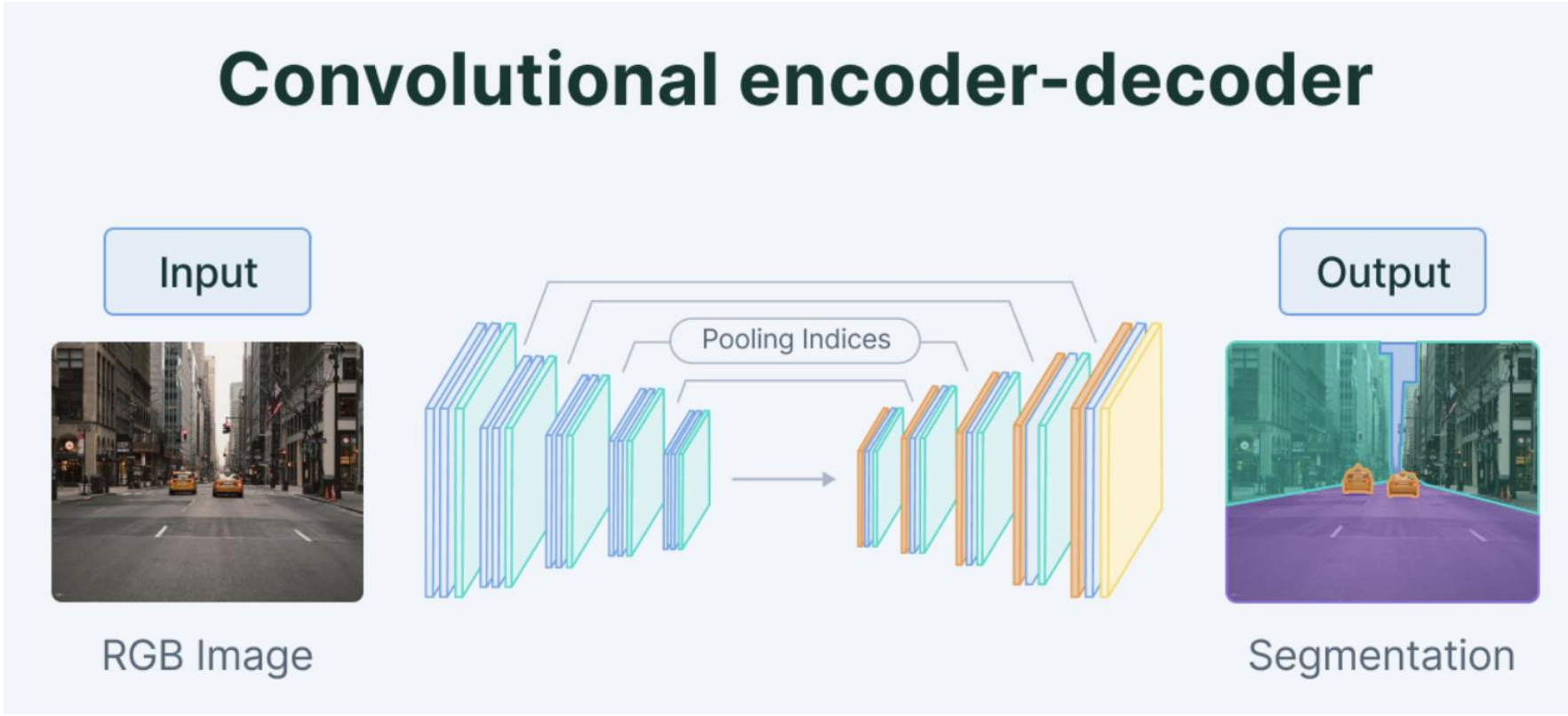
- We already talked about the concept of training autoencoders where the input and outputs are identical and the task is to reproduce the input as closely as possible. However, one way to develop a generalizable model is to slightly augment the input data but still keep the original input as our target output. Imagine that you start with a normal MNIST digit and add noise to them and run that noisy image through the autoencoder model. The only difference is that instead of reconstructing the noisy images, you try and reconstruct the original images. By training our autoencoder model this way, we force the model to get rid of the noise and eventually our model becomes really good at denoising images. That is how denoising autoencoders are used.



Applications of Auto-Encoders

Image Segmentation

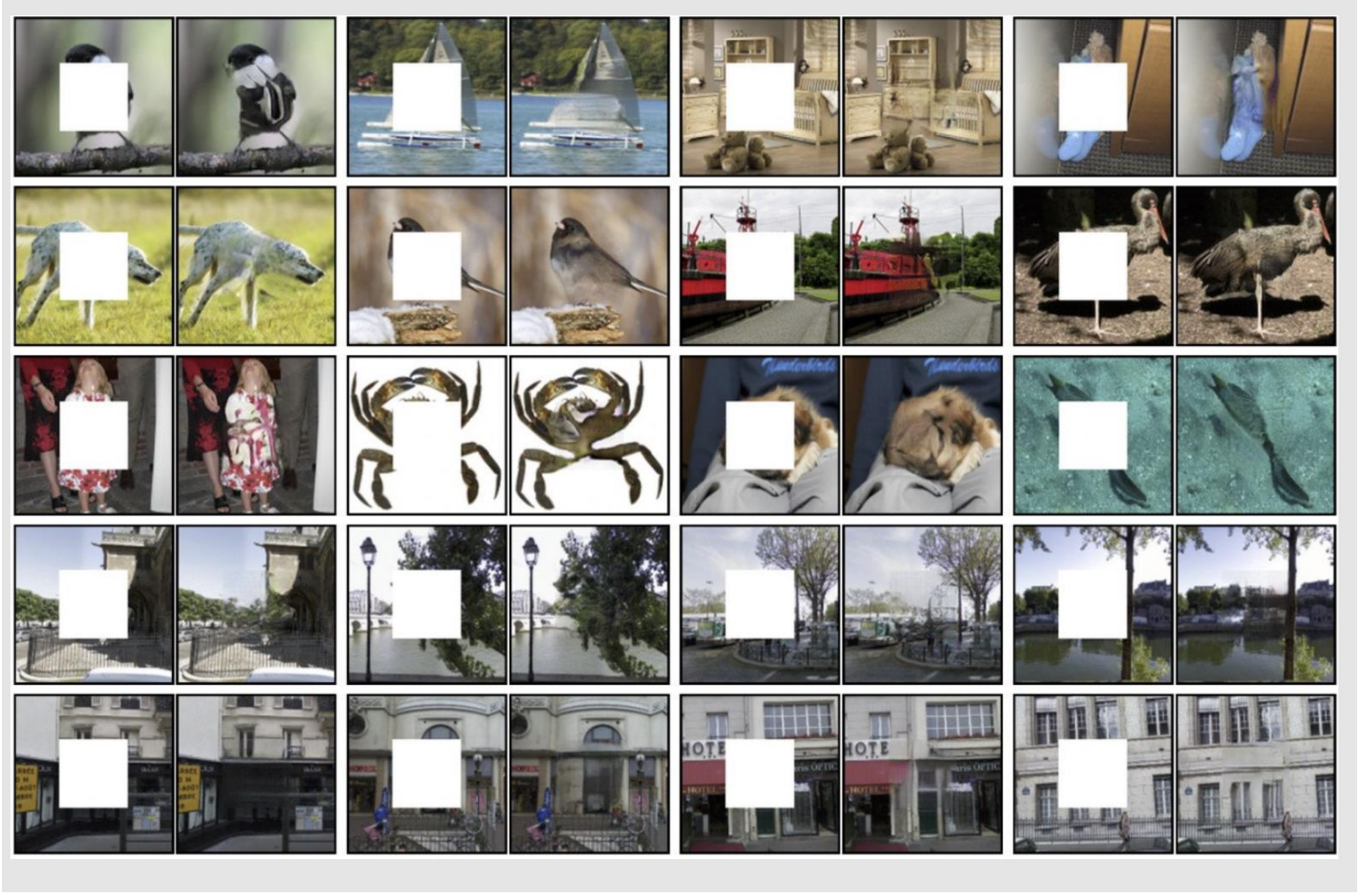
Image segmentation is another example where autoencoders are used to take image inputs preserving spatial representation and aim to output semantic segmented counterparts of the image. This technique is used in self-driving cars to segment different objects in their environment.



Applications of Auto-Encoders

Neural Inpainting

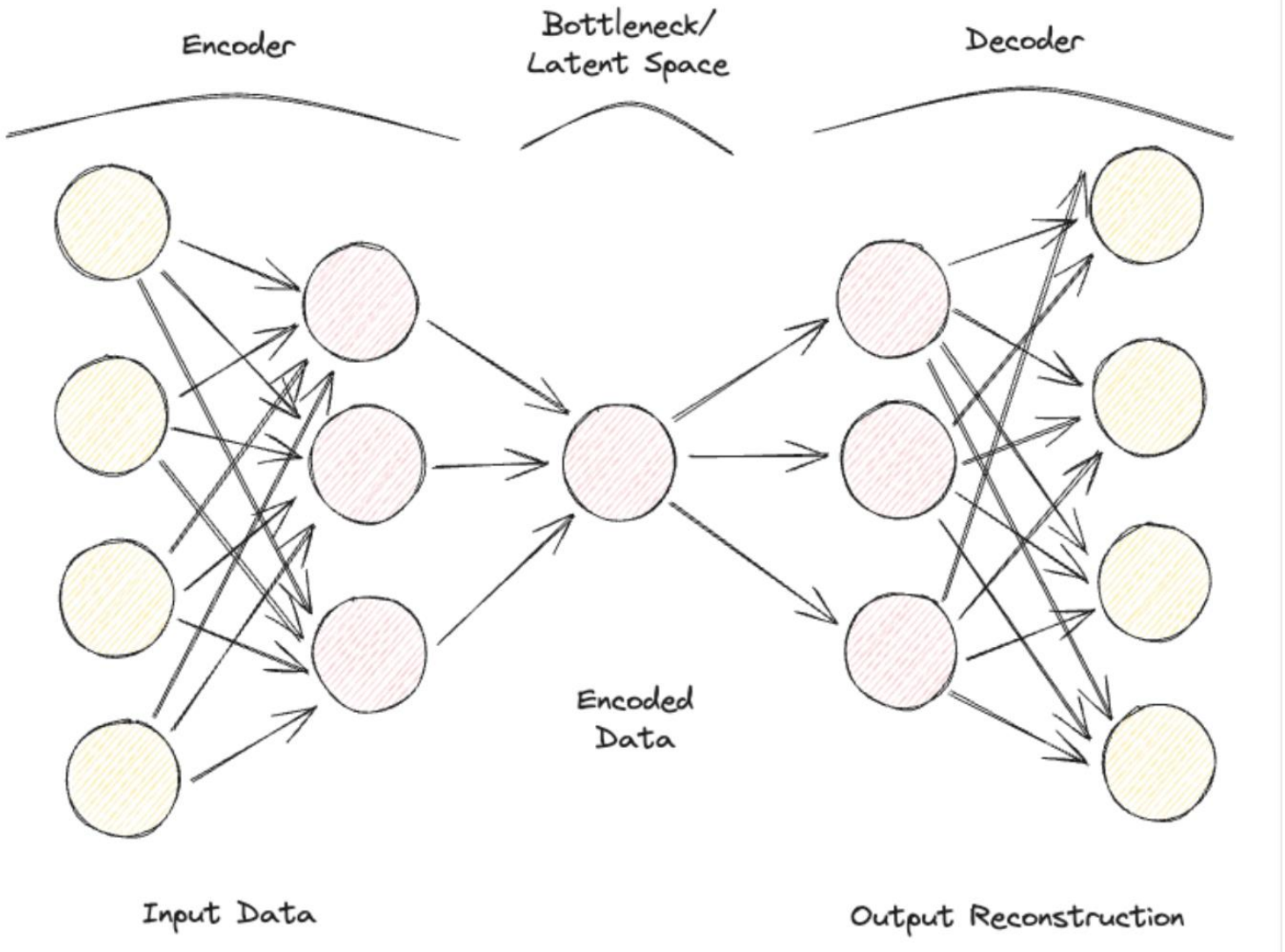
Lastly, I am going to talk about neural inpainting approaches where instead of adding noise to our inputs, you basically block parts of the image and force the AI model to reconstruct the missing parts. This approach simply powers many apps we use these days including Magic Eraser on the Pixel phones which uses similar machine learning concepts.



Please refer to Codes on Canvas

Types of Auto-Encoder

1. *Undercomplete Autoencoder*
2. *Sparse Autoencoder*
3. *Contractive Autoencoder*
4. *Denoising Autoencoder*
5. *Convolutional Autoencoder*
6. *Variational Autoencoder*

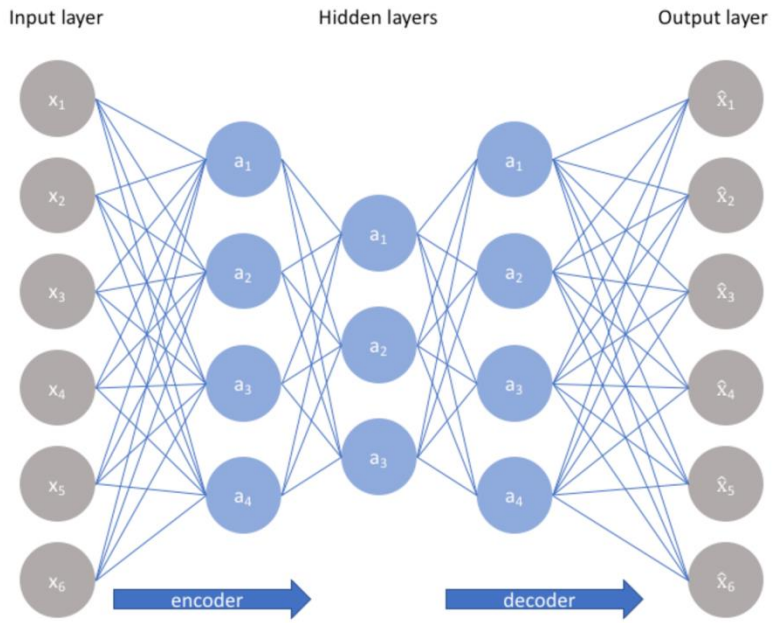


Types of Auto-Encoder

1. *Undercomplete Autoencoder*

This is the simplest version of an autoencoder. In this case, we don’t have an explicit regularization mechanism, but we ensure that the size of the bottleneck is always lower than the original input size to avoid overfitting. This type of configuration is typically used as a dimensionality reduction technique (more powerful than PCA since its also able to capture non-linearities in the data).

The simplest architecture for constructing an autoencoder is to constrain the number of nodes present in the hidden layer(s) of the network, limiting the amount of information that can flow through the network. By penalizing the network according to the reconstruction error, our model can learn the most important attributes of the input data and how to best reconstruct the original input from an "encoded" state. Ideally, this encoding will learn and describe latent attributes of the input data.

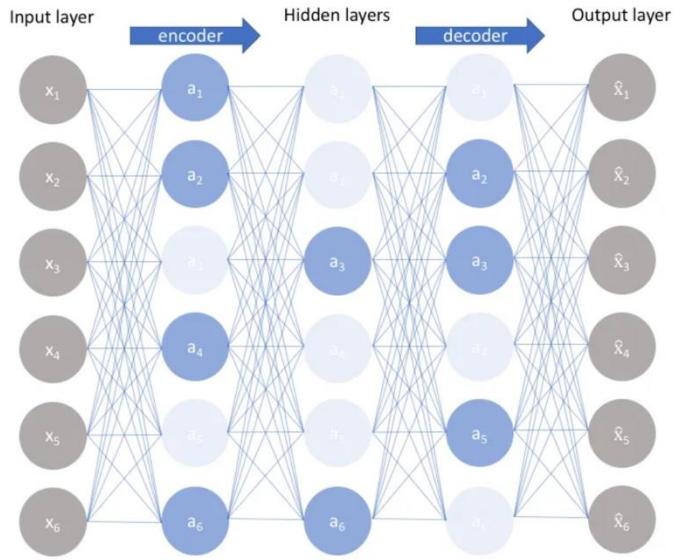


Types of Auto-Encoder

Sparse Autoencoder

A Sparse Autoencoder is quite similar to an Undercomplete Autoencoder, but their main difference lies in how regularization is applied. In fact, with Sparse Autoencoders, we don’t necessarily have to reduce the dimensions of the bottleneck, but we use a loss function that tries to penalize the model from using all its neurons in the different hidden layers .

This penalty is commonly referred to as a **sparsity function**, and it's quite different from traditional regularization techniques since it doesn’t focus on penalizing the size of the weights but the number of nodes activated. In this way, different nodes could specialize for different input types and be activated/deactivated depending on the specifics of the input data. This sparsity constraint can be induced by using L1 Regularization and KL divergence, effectively preventing the model from overfitting.



Sparse Autoencoder

Types of Auto-Encoder

Contractive Autoencoder

The main idea behind Contractive Autoencoders is that given some similar inputs, their compressed representation should be quite similar (neighborhoods of inputs should be contracted in small neighborhood of outputs). In mathematical terms, this can be enforced by keeping input hidden layer activations derivatives small when fed similar inputs.

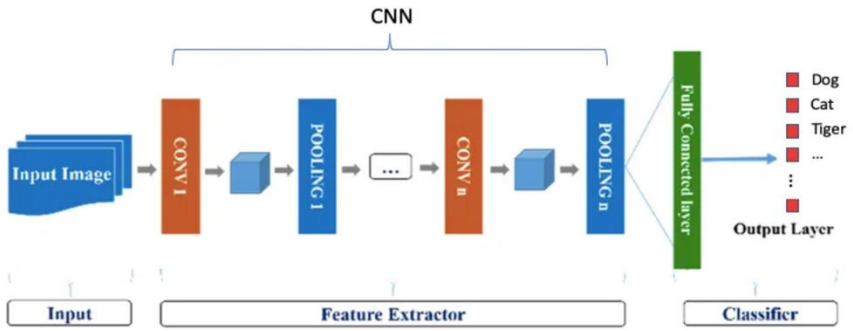
Denoising Autoencoder

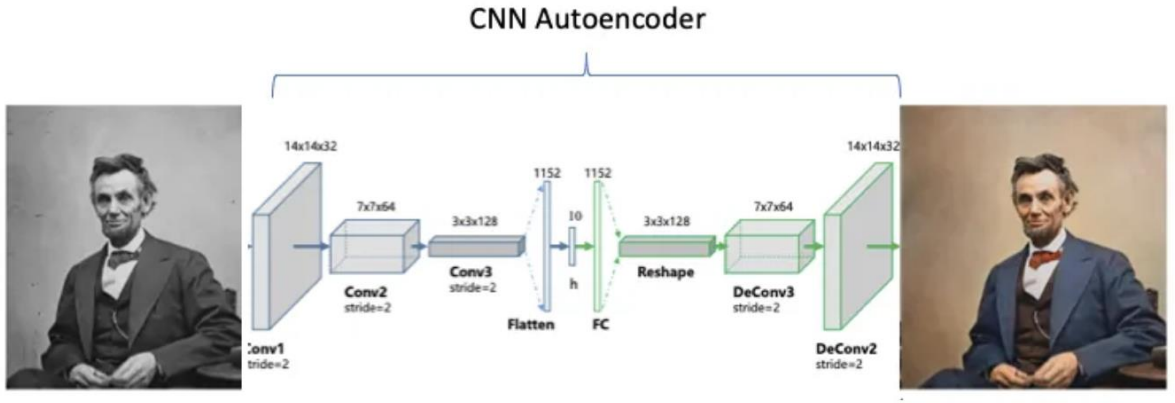
With Denoising Autoencoders, the input and output of the model are no longer the same. For example, the model could be fed some low-resolution corrupted images and work for the output to improve the quality of the images. In order to assess the performance of the model and improve it over time, we would then need to have some form of labeled clean image to compare with the model prediction.

Types of Auto-Encoder

Convolutional Autoencoder

To work with image data, Convolutional Autoencoders replace traditional feedforward neural networks with Convolutional Neural Networks for both the encoder and decoder steps. Updating type of loss function, etc., this type of Autoencoder can also be made, for example, Sparse or Denoising, depending on your use case requirements.



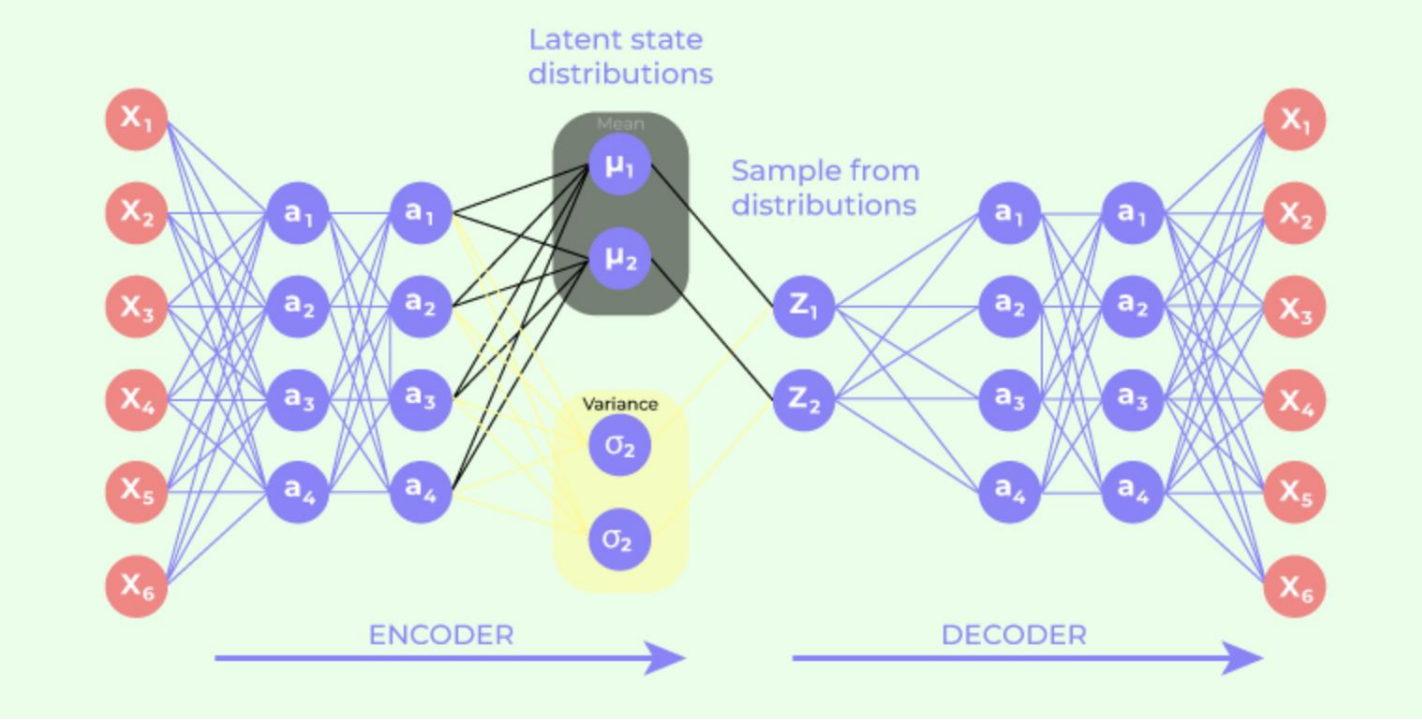


When CNN is used for image noise reduction or coloring, it is applied in an Autoencoder framework, i.e, the CNN is used in the encoding and decoding parts of an autoencoder. Each of the input image samples is an image with noises, and each of the output image samples is the corresponding image without noises. We can apply the trained model to a noisy image and then output a clear image. Likewise, it can be used to train a model for image coloring.

Types of Auto-Encoder

riational Autoencoder

every type of Autoencoder considered so far, the encoder outputs a single value for each dimension involved. With Variational Autoencoders (VAE), we make this process instead probabilistic, creating a probability distribution for each dimension. The decoder can then sample a value from each distribution describing the different dimensions and construct the input vector, which it can then be used to reconstruct the original input data. One of the main applications of Variational Autoencoders is for generative tasks. In fact, sampling the latent model from distributions can enable the decoder to create new forms of outputs that were previously not possible using a deterministic approach.



Architecture of Variational Autoencoder

- The encoder-decoder architecture lies at the heart of Variational Autoencoders (VAEs), distinguishing them from traditional autoencoders. The encoder network takes raw input data and transforms it into a probability distribution within the latent space.
- The latent code generated by the encoder is a probabilistic encoding, allowing the VAE to express not just a single point in the latent space but a distribution of potential representations.
- The decoder network, in turn, takes a sampled point from the latent distribution and reconstructs it back into data space. During training, the model refines both the encoder and decoder parameters to minimize the reconstruction loss – the disparity between the input data and the decoded output. The goal is not just to achieve accurate reconstruction but also to regularize the latent space, ensuring that it conforms to a specified distribution.
- The process involves a delicate balance between two essential components: the reconstruction loss and the regularization term, often represented by the Kullback-Leibler divergence. The reconstruction loss compels the model to accurately reconstruct the input, while the regularization term encourages the latent space to adhere to the chosen distribution, preventing overfitting and promoting generalization.
- By iteratively adjusting these parameters during training, the VAE learns to encode input data into a meaningful latent space representation. This optimized latent code encapsulates the underlying features and structures of the data, facilitating precise reconstruction. The probabilistic nature of the latent space also enables the generation of novel samples by drawing random points from the learned distribution.

Please refer to Codes on Canvas


THANK YOU
Any Questions?