CS3232 Fundamental of Deep learning

Parts of the contents are from deeplearning.ai, TensorFlow, jalFaizy and other resources on the Internet

Unsupervised Learning and Generative

Autoencoder (Deep, CNN-based),

Training Autoencoders,

Variational Autoencoders (VAE),

Introduction to GAN

GAN Architecture (Generator, Discriminator),

Applications.

Deep Unsupervised Learning Models

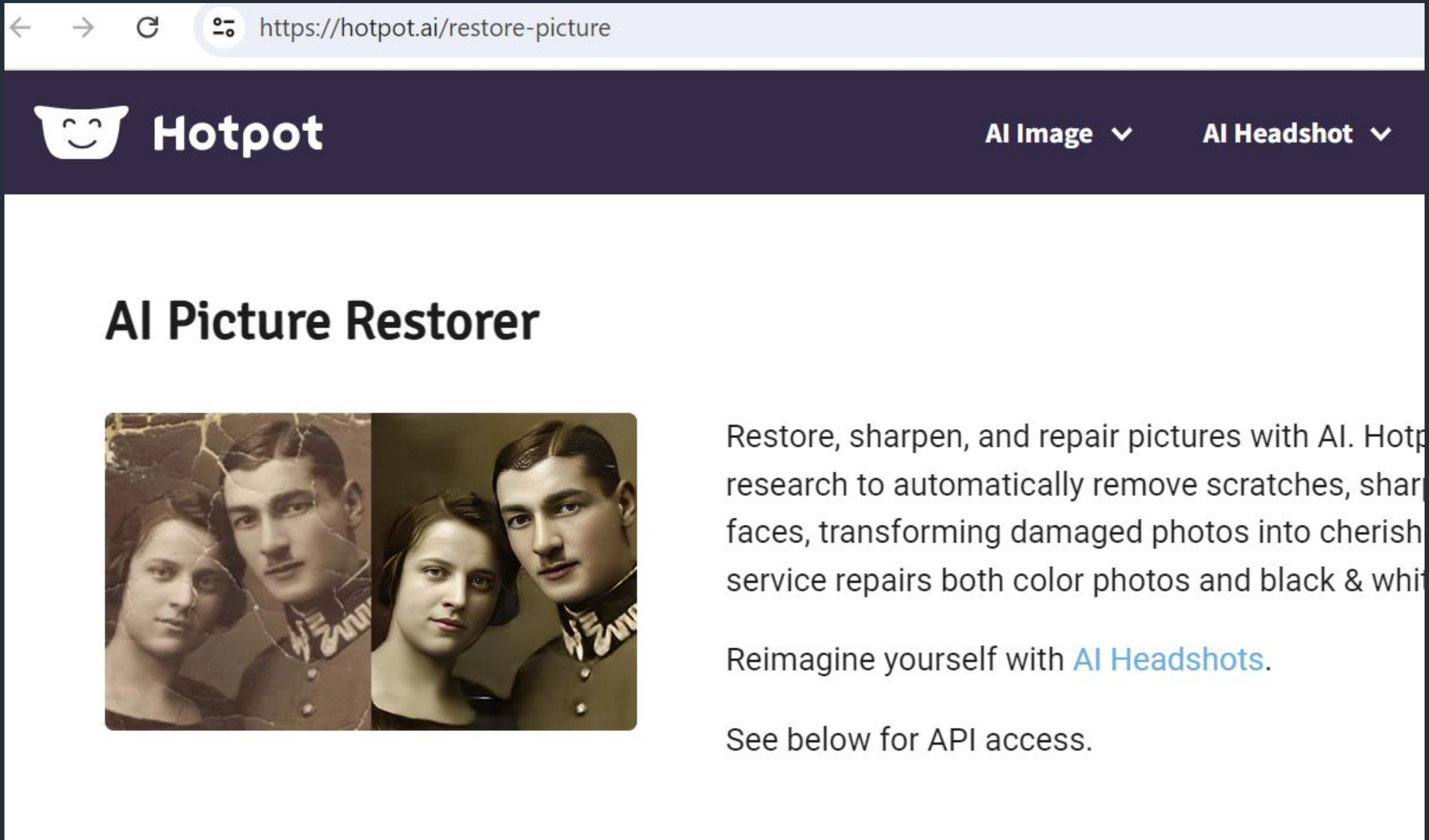
RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Deep Unsupervised Learning models are a class of deep learning techniques where the model is trained on data without explicit labels.
- These models learn to identify patterns, structures, and relationships within the data, which can be used for various tasks such as clustering, dimensionality reduction, anomaly detection, and generative modeling.

Deep Unsupervised Learning Models

- Autoencoders (AEs) – to generate new content
- Generative Adversarial Networks (GANs)
- Self-Organizing Maps (SOMs)
- Deep Belief Networks (DBNs)

Autoencoders (AEs) Example



- https://hotpot.ai/restore-picture

- Application: Music Recommendation System
- Description: Utilizes autoencoders to learn compact representations of user listening patterns and preferences, enabling personalized music recommendations.
- https://research.atspotify.com/publications/variational-user-modeling-with-slow-and-fast-features/

GAN Example

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

https://this-person-does-not-exist.com

AutoEncoders

- Neural networks capable of learning internal (dense) representations of input data without supervision
  - Training data is not labelled
  - Similar in concept to compression methodology
    - Converts a big image to a smaller image without loosing too much information
      - But not a direct compression of pixels
      - Lossy representation of the contents in the image but maintains the sense of the original image
- Useful for dimensionality reduction and for visualization (10D -> 3D)
- Can be used to generate new data that resembles input data – focus of this course
  - By learning a representation of the input data
  - Create images by the method of the internal representation so that decoding can create something new
- They copy input to output and learn efficient ways to represent data

Learn representation

Example

83 12 21 42 99 18 51

4 9 16 25 36 49 64 81 100 121 144 169

Learn representation

Example

Which set can you recall?

Learn representation

Example

Which set can you recall?

83 12 21 42 99 18 51

4 9 16 25 36 49 64 81 100 121 144 169

No need to remember if recognized the pattern

They are squares

Learn representation

Example

Which set can you recall?

83 12 21 42 99 18 51

4 9 16 25 36 49 64 81 100 121 144 169

No need to remember if recognized the pattern

They are squares

Latent representation

Autoencoders

Autoencoders are to create a latent representation

so that when it is asked to produce an output

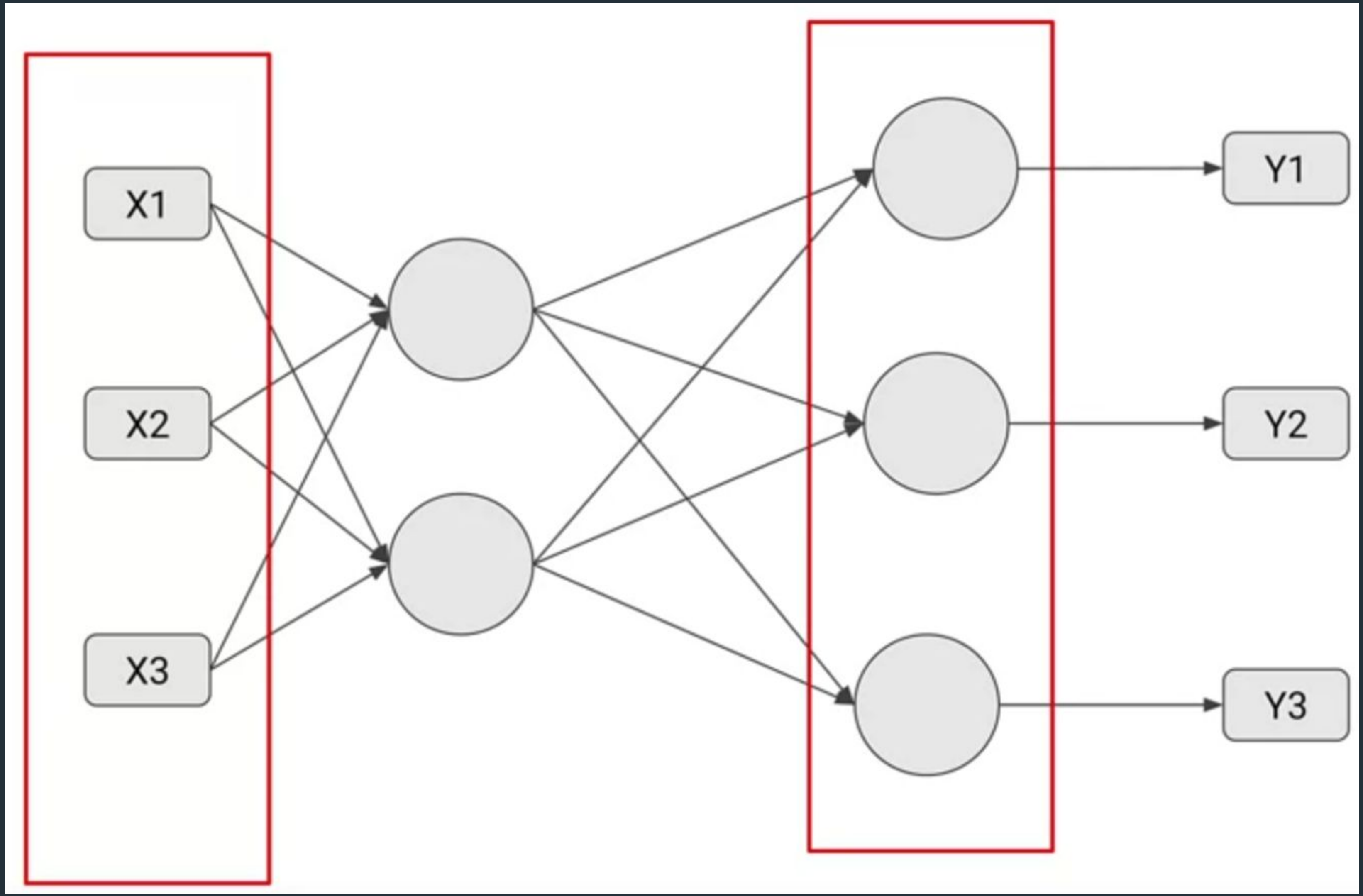
it can do so similar to how you recalled the set of numbers

(the best way to remember the numbers was to remember

they were squares so that you can output a new one too)

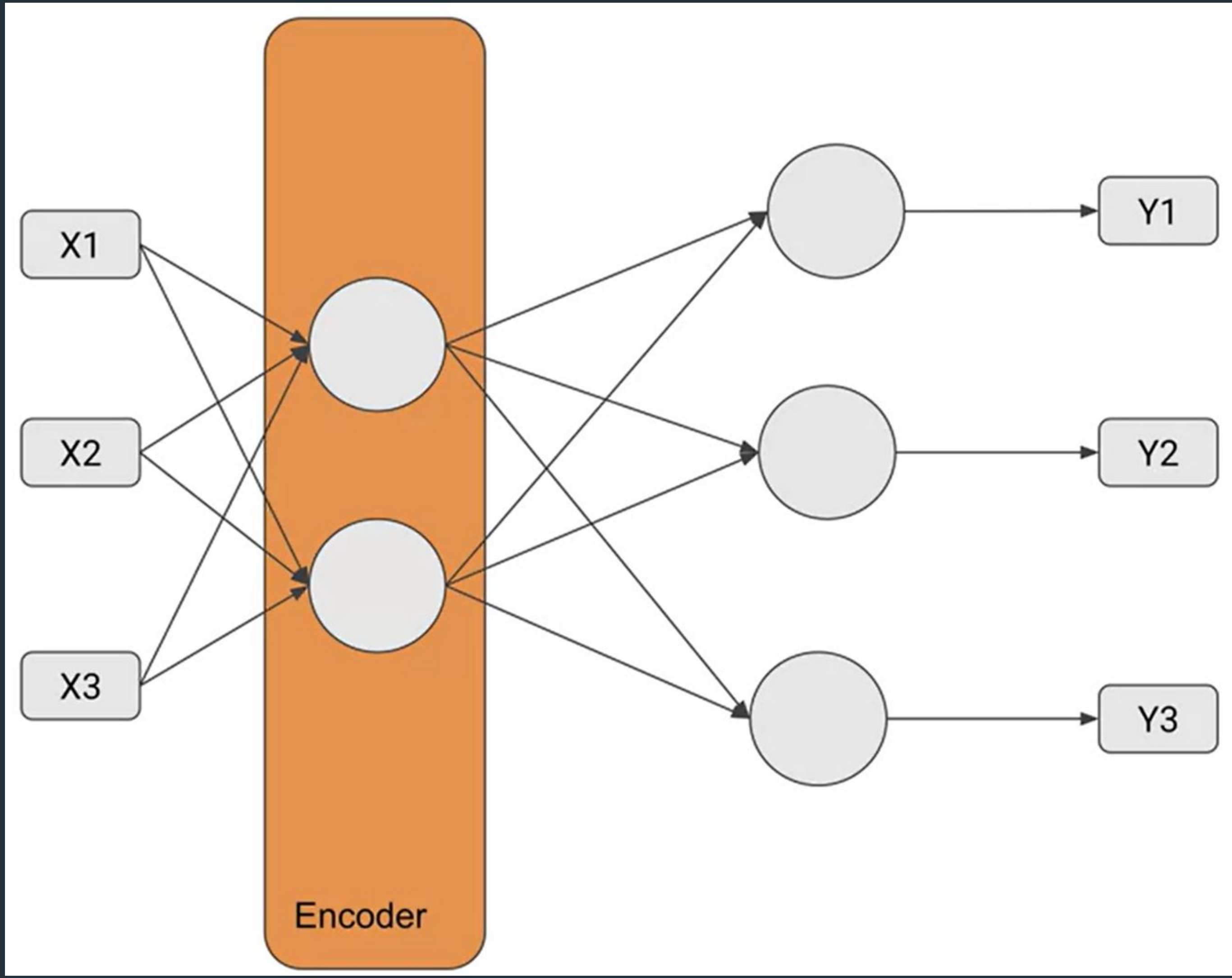
Autoencoder architecture

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS



Number of output neurons must match the number of input neurons

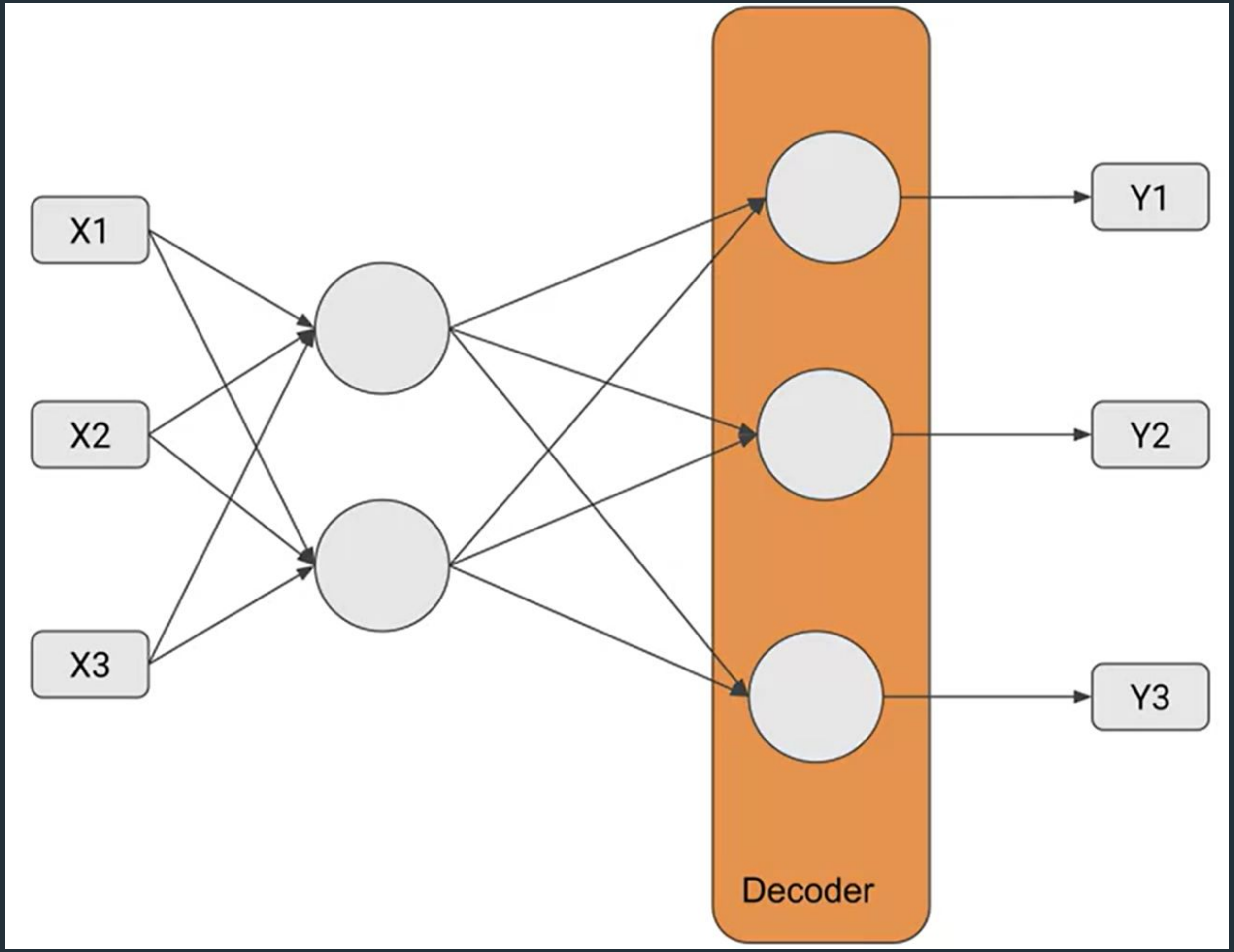
Autoencoder architecture



Encodes the latent representation of the input data

Autoencoder architecture

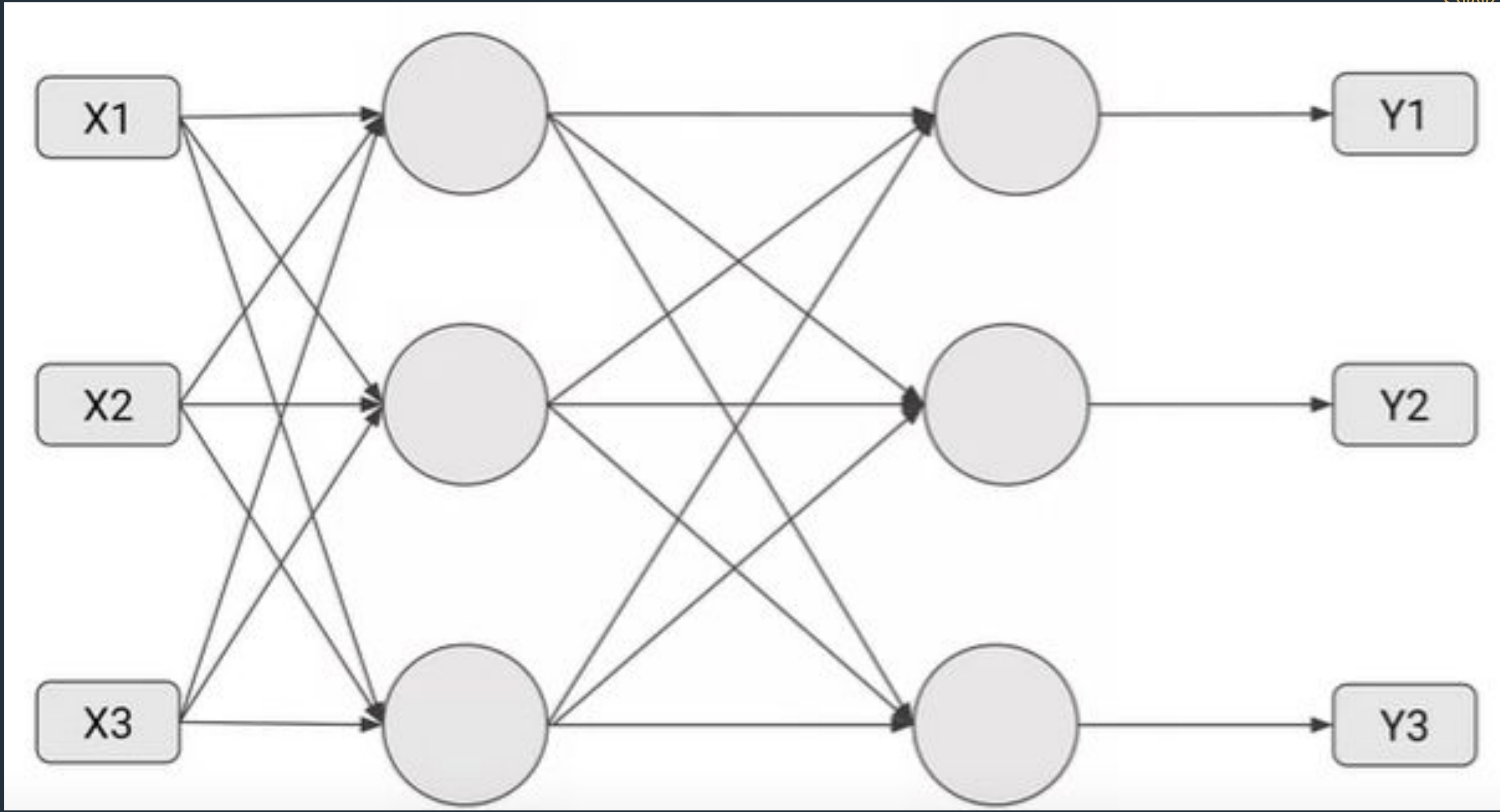
RV UNIVERSITY
प्रज्ञा: ★ धीरा:
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS



Decodes the latent representation into the output.

But because of the dimension (information) loss in the encoder layer, there will be an approximation learnt from the training data

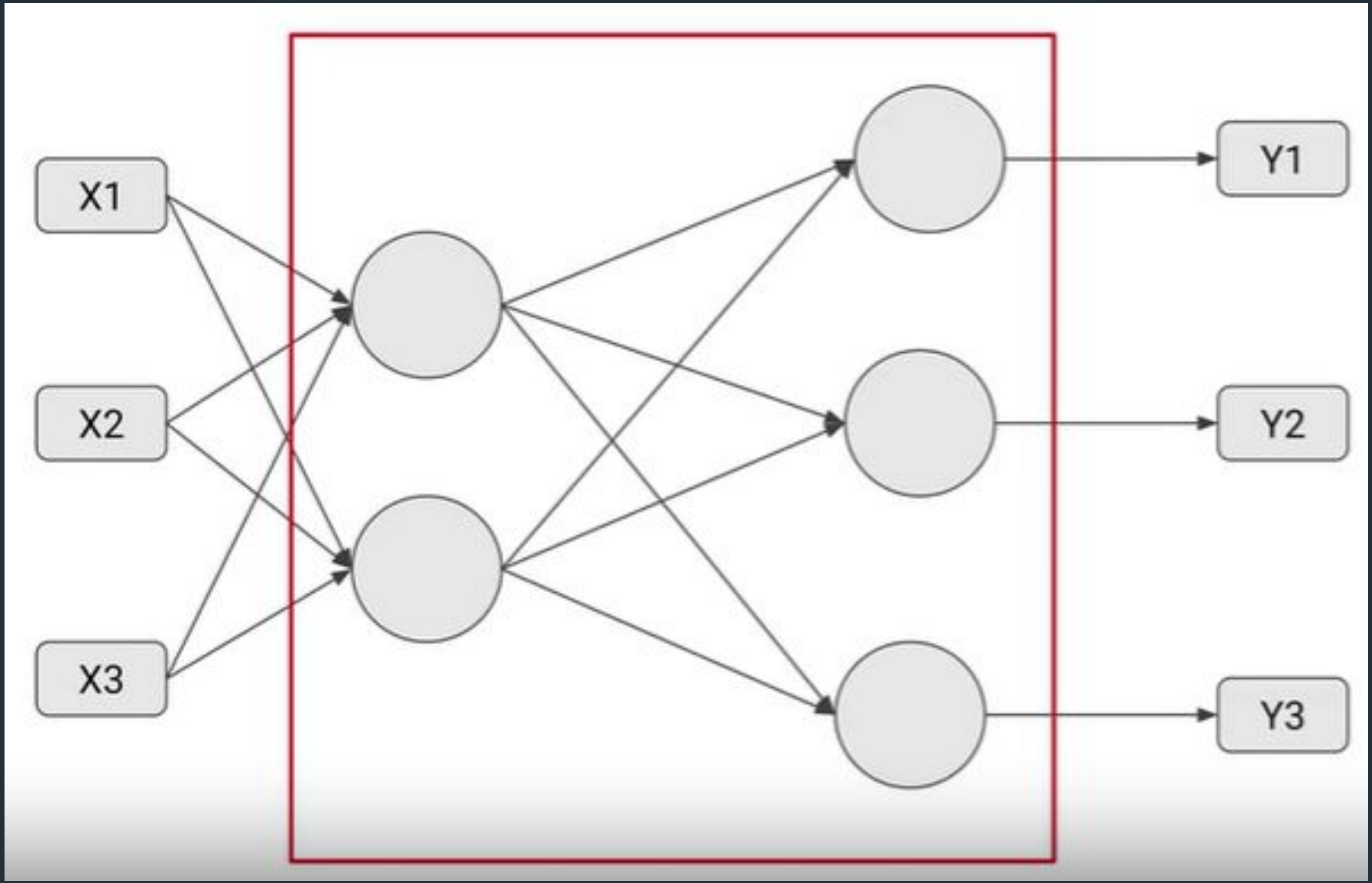
Autoencoder architecture



Why not the same number of neurons like here?

Input can go straight to output and pattern of representation may not be learnt

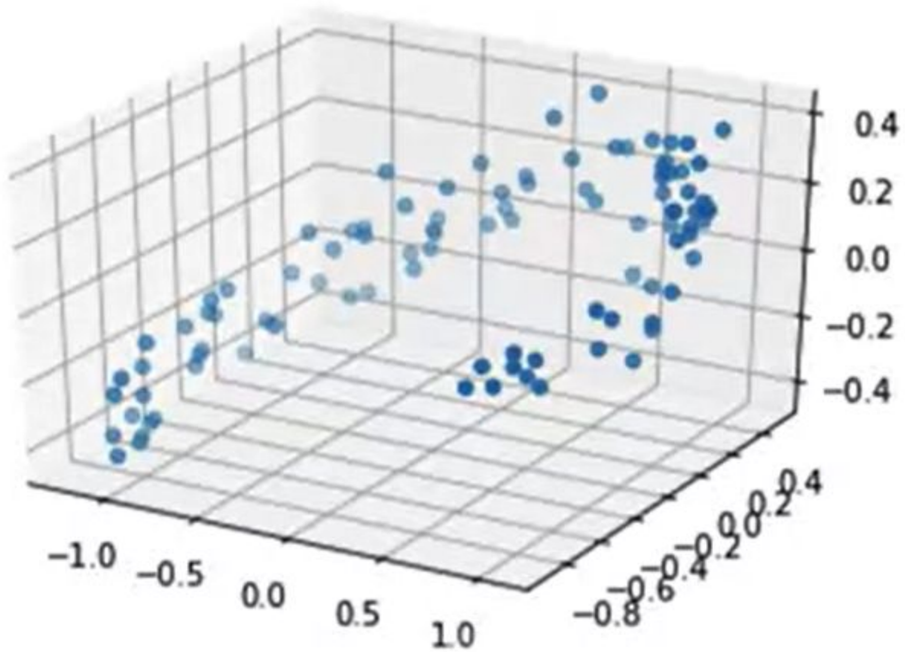
Autoencoder architecture



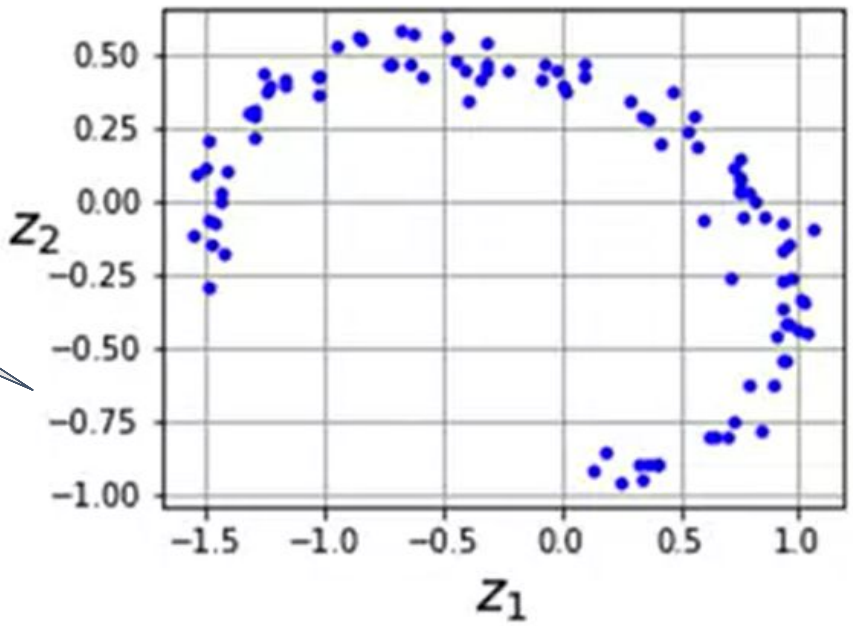
Undercomplete network design

- 2D representation of the 3D input here

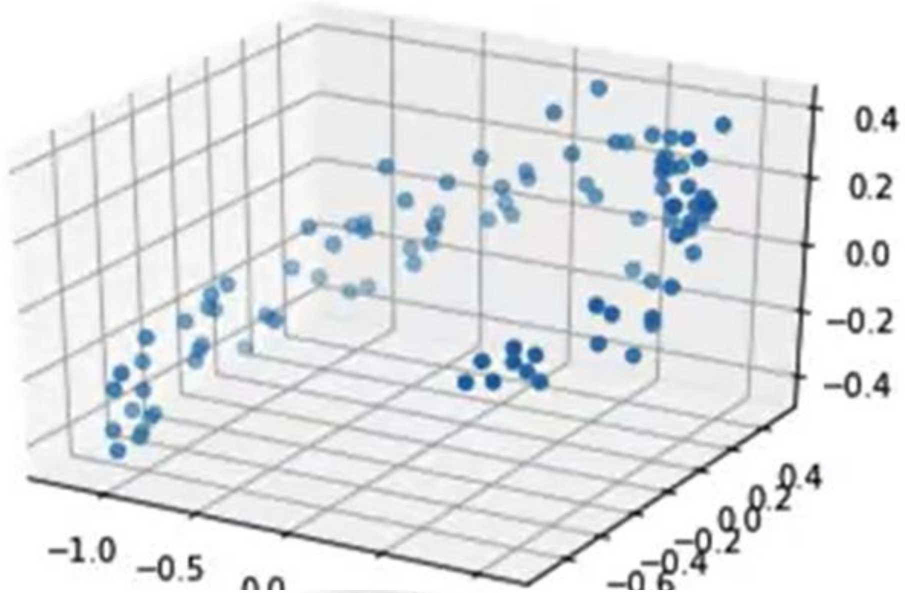
Dimensionality reduction



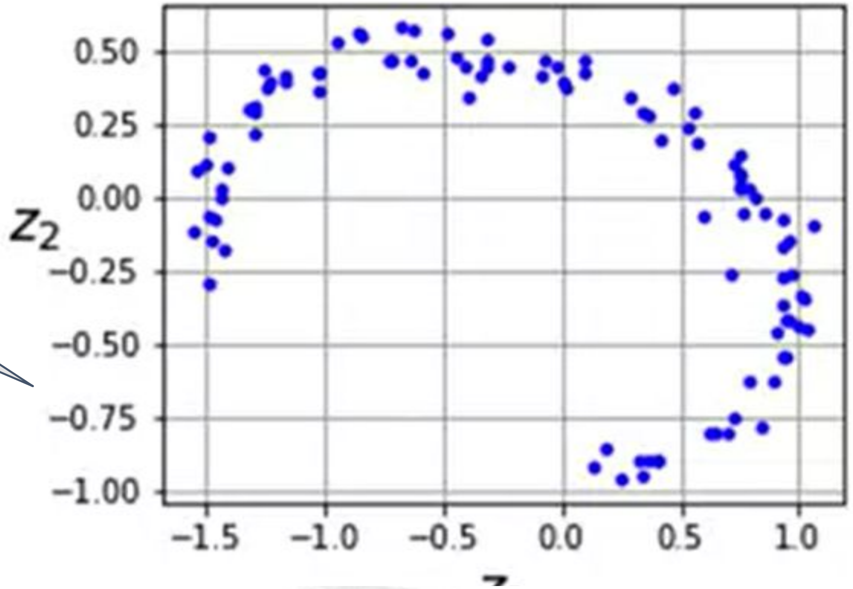
Information loss
~ noise loss

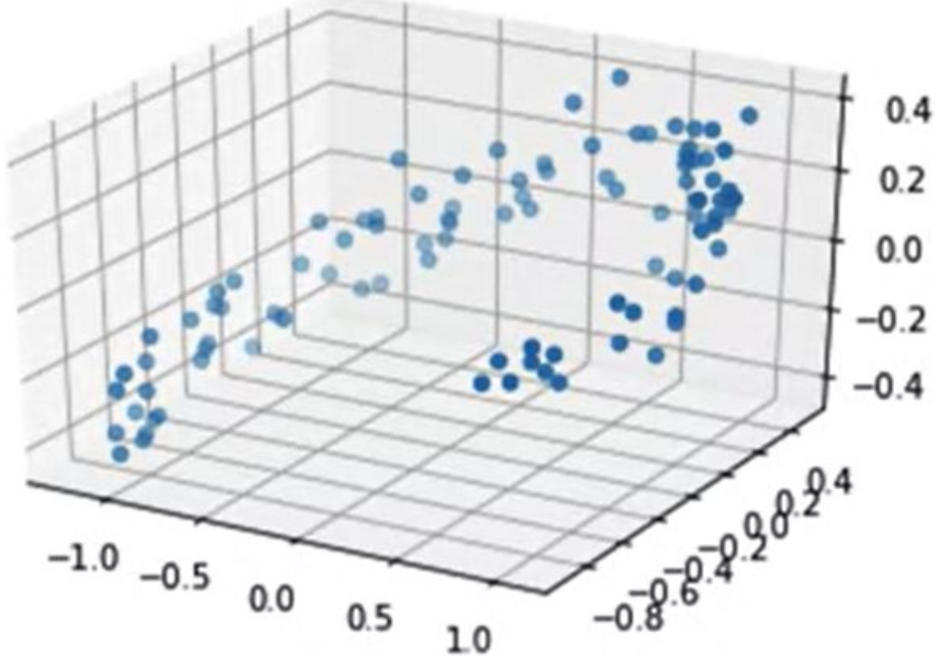


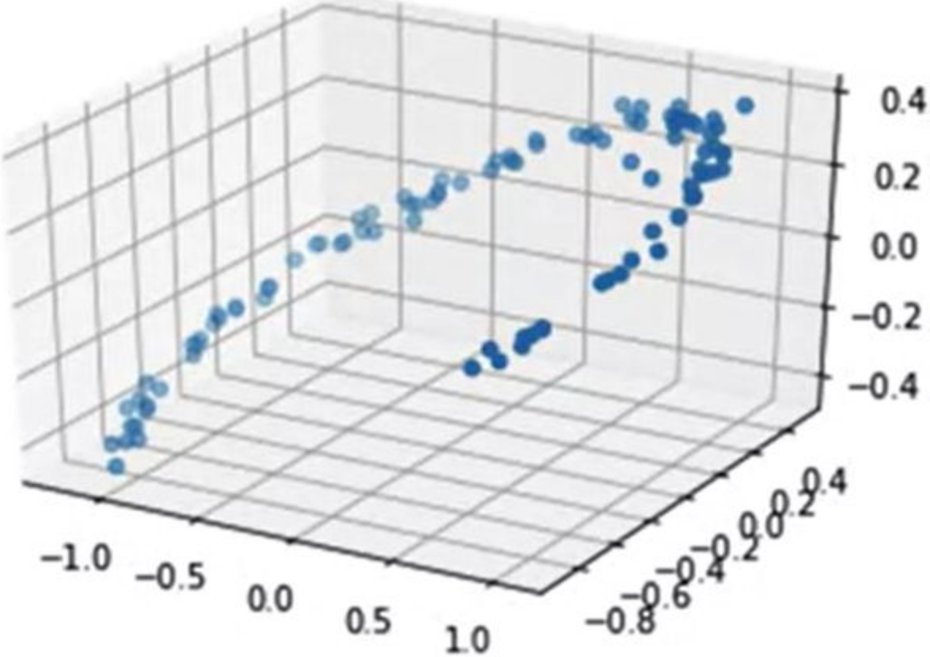
Dimensionality reduction



Information loss
~ noise loss





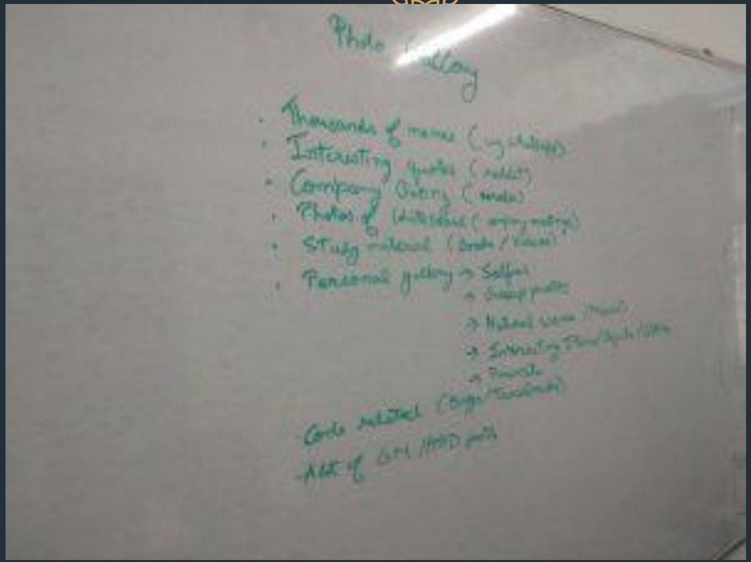


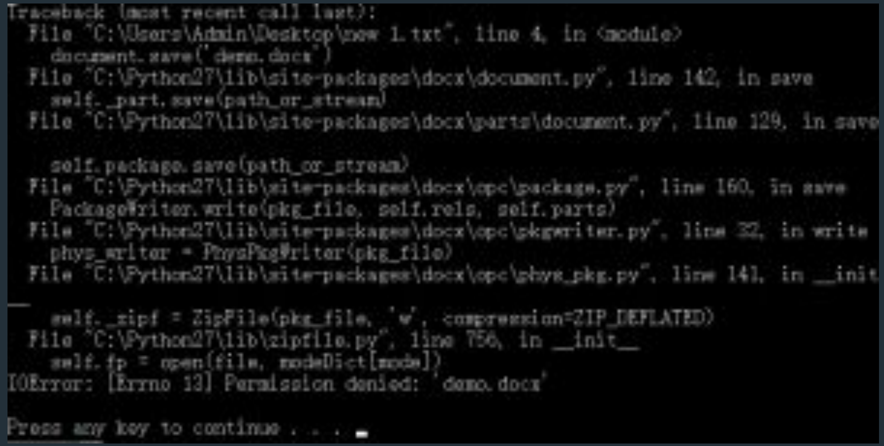
Case Study of Unsupervised Deep Learning

Say, you have 2000+ photos in your phone now. If you had been a selfie freak, the photo count would easily be 10 times more. Sifting through these photos is a nightmare, because every third photo turns out to be unnecessary and useless for you.

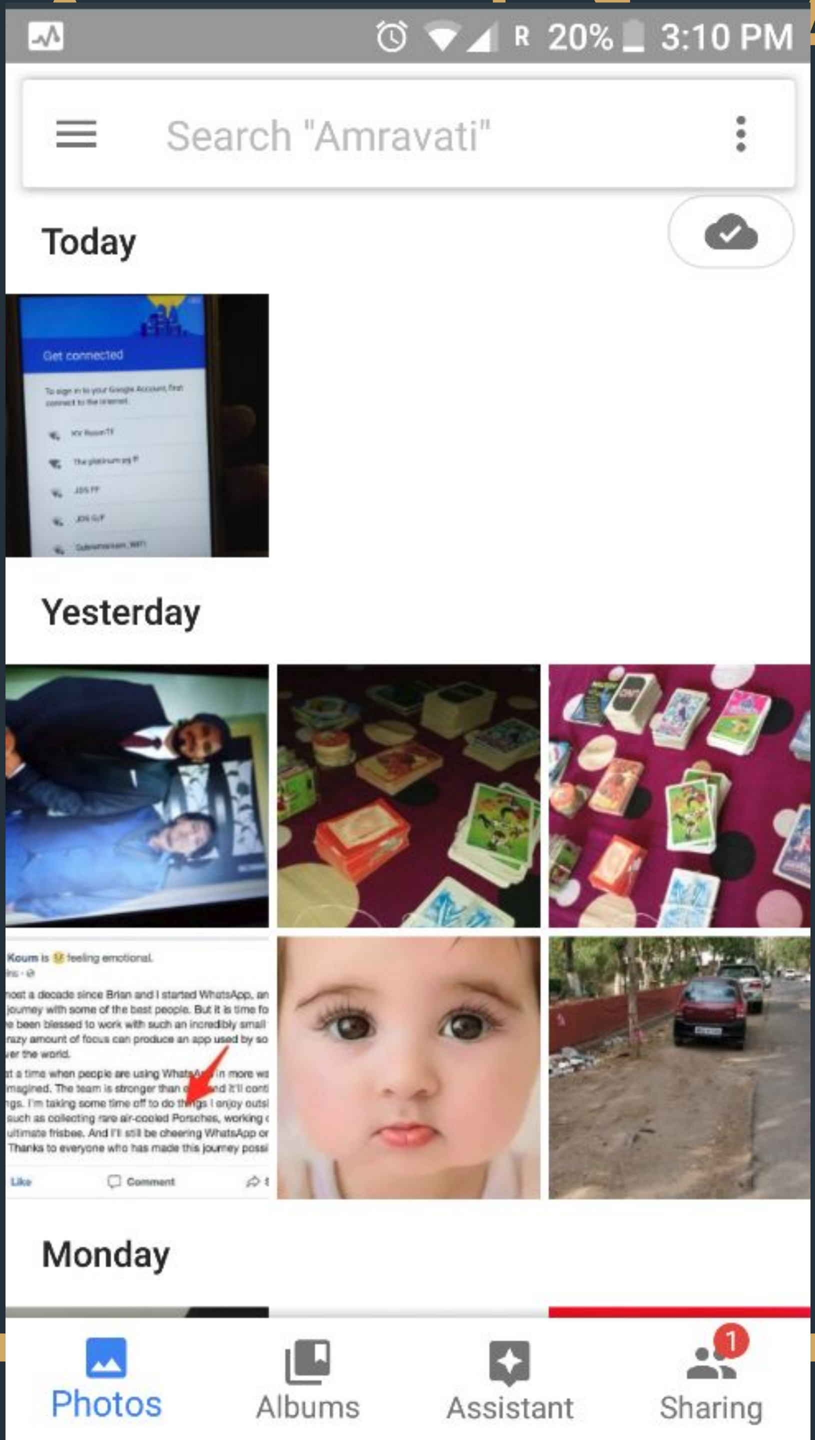
Ideally, what you want is an app which organizes the photos in such a manner that you can go through most of the photos and have a peek at it if you want. This would actually give you context as such of the different kinds of photos you have right currently.

Categories of photos





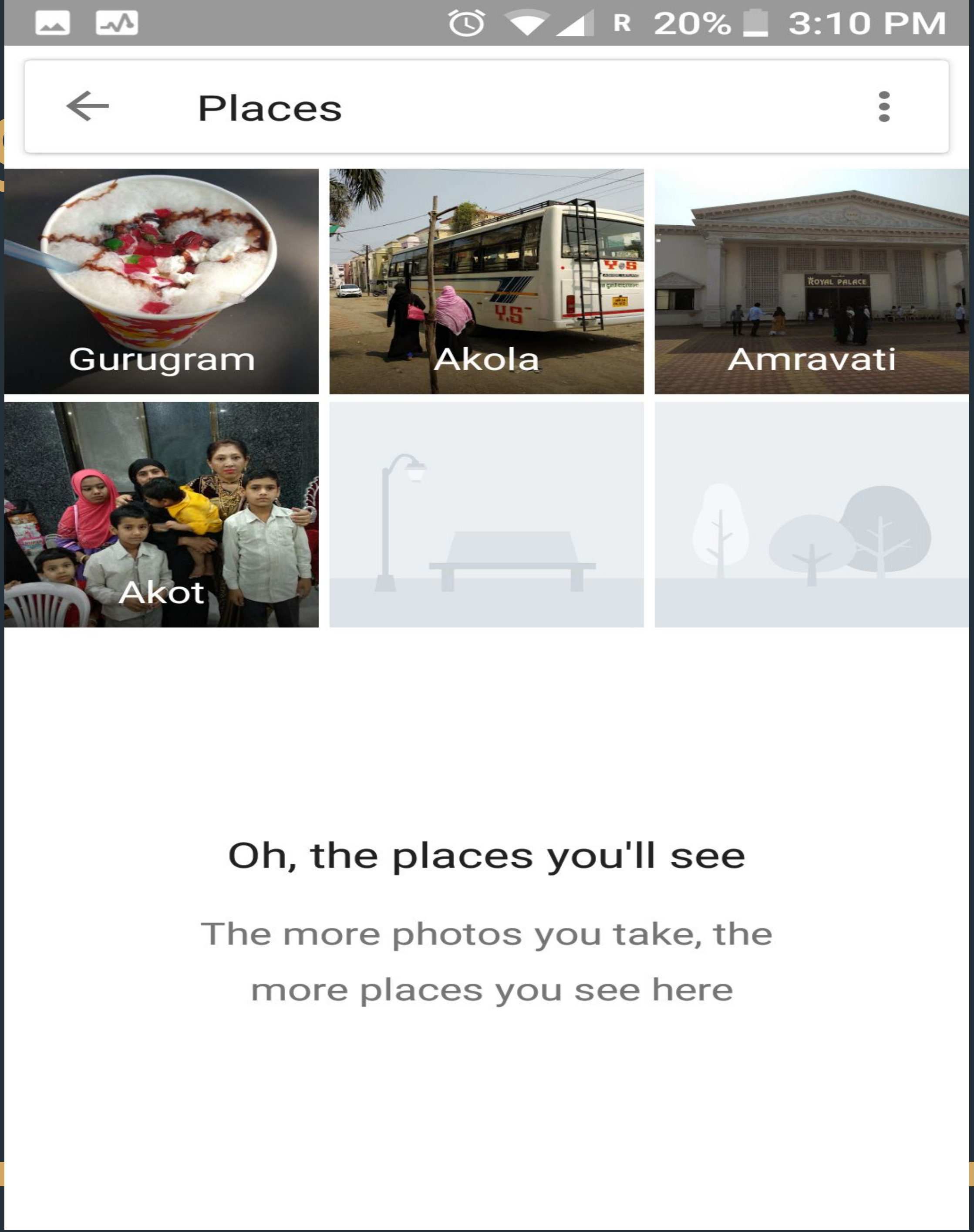
Arrange on the



Events of the day are together but is not group according to "good morning" versus "personal"

Approach 2 – Arrang
basis of location

Not possible to define a location for a meme etc.

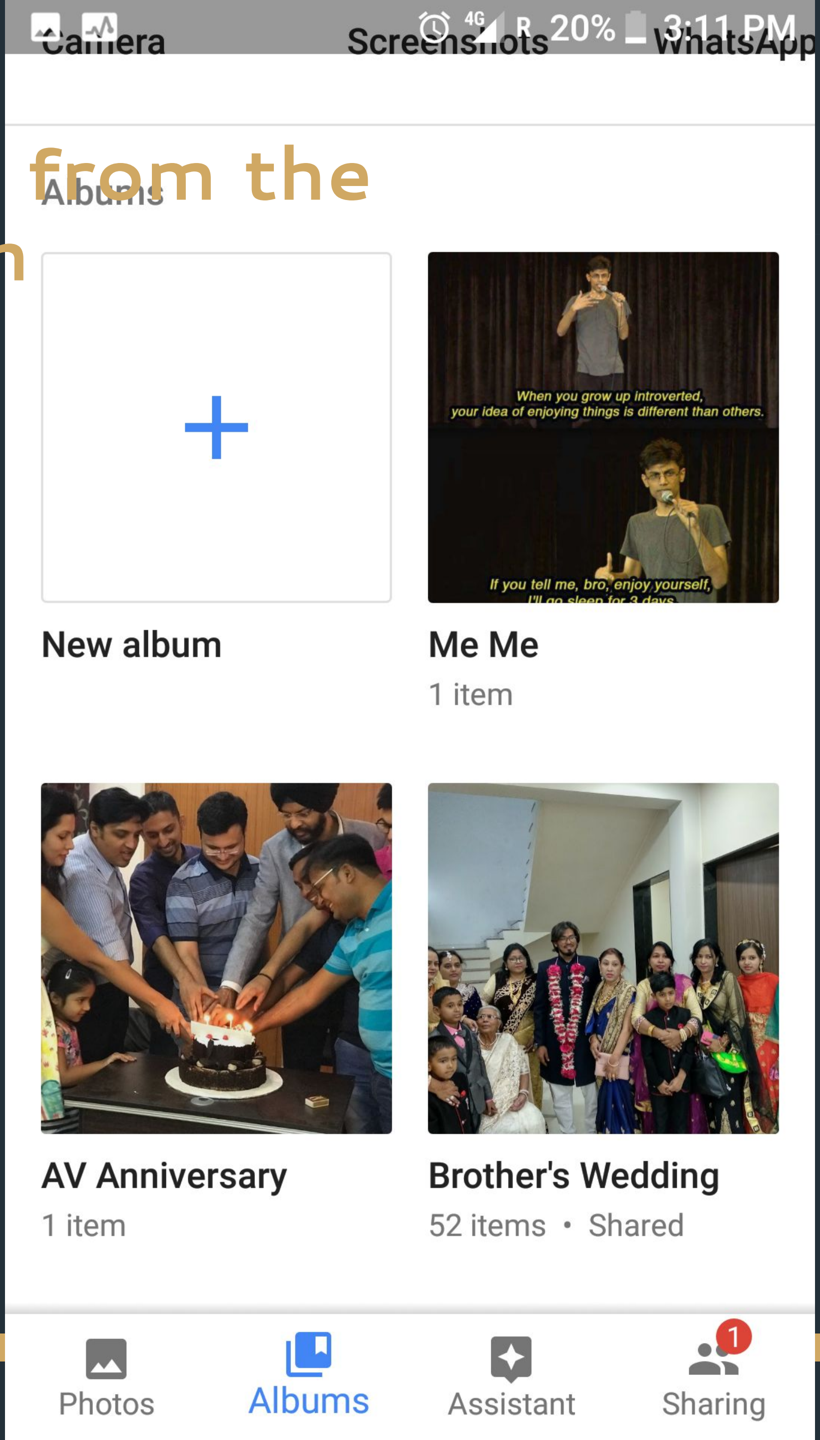


Approach 3 – Extract semantic meaning from the image and use it to define your collection

Previous approaches were based on metadata captured with images.

Semantic information-based:

- Natural vs artificially generated image
- Text in the image – can we identify what it is?
- Kinds of objects in the image – do they combine to define the aesthetics of the image?
- People? Can we recognize them?
- Similar image on the Internet that can help us identify the context of the image?



Approach 3 – Extract semantic meaning from the image and use it to define your collection

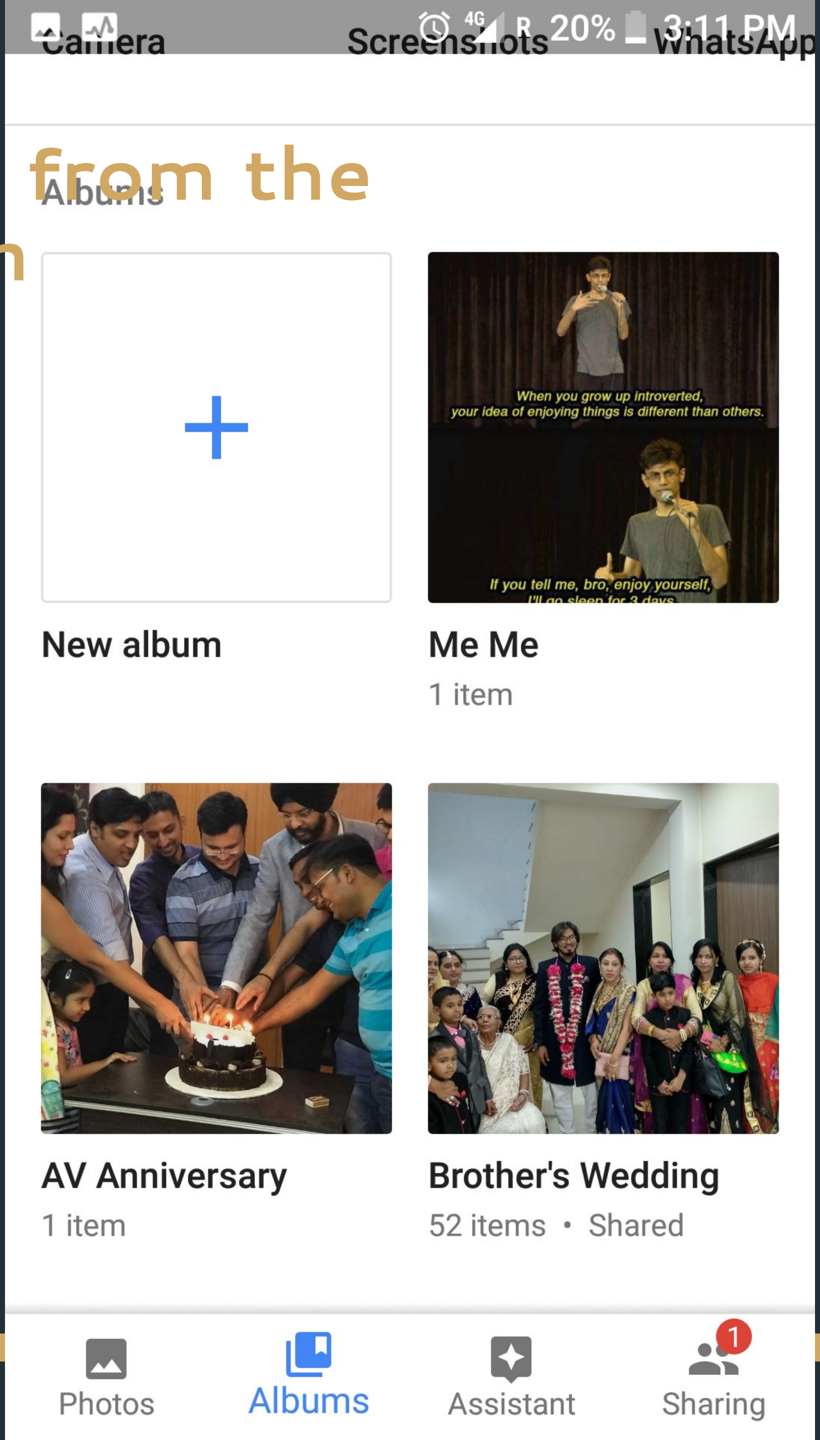
Previous approaches were based on metadata captured with images.

Semantic information-based:

- Natural vs artificially generated image
- Text in the image – can we identify what it is?
- Kinds of objects in the image – do they combine to define the aesthetics of the image?
- People? Can we recognize them?
- Similar image on the Internet that can help us identify the context of the image?

Capture these <u>without</u> explicitly tagging what is there and not there.

Unsupervised way



Approach 3 – Extract semantic meaning from the image and use it to define your collection

Previous approaches were based on metadata captured with images.

Semantic information-based:

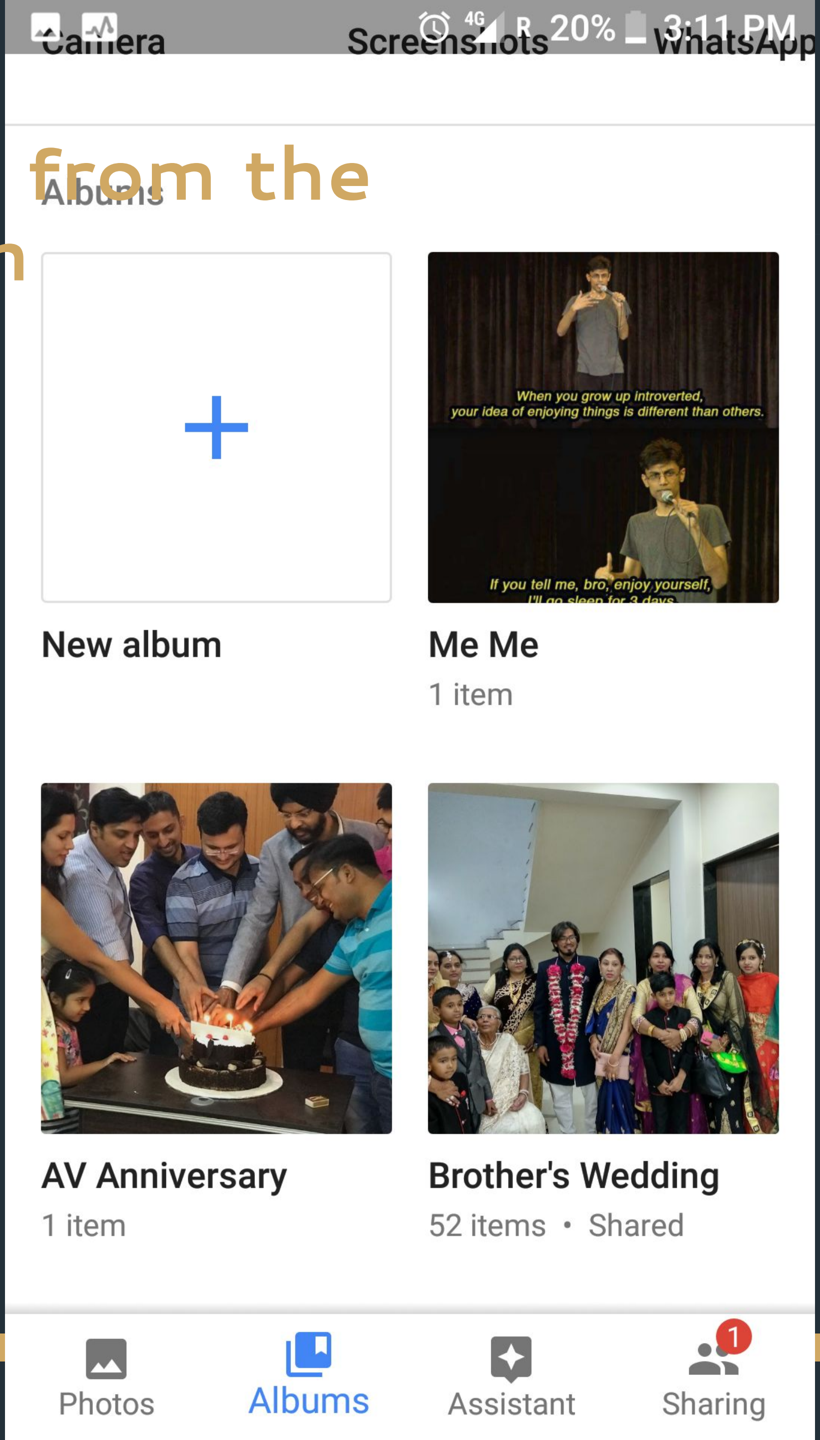
- Natural vs artificially generated image
- Text in the image – can we identify what it is?
- Kinds of objects in the image – do they combine to define the aesthetics of the image?
- People? Can we recognize them?
- Similar image on the Internet that can help us identify the context of the image?

Capture these without explicitly tagging what is there and not there.

Unsupervised way

Reduce the dimensions of an image so that we can reconstruct the image back from these encoded representations

using autoencoders

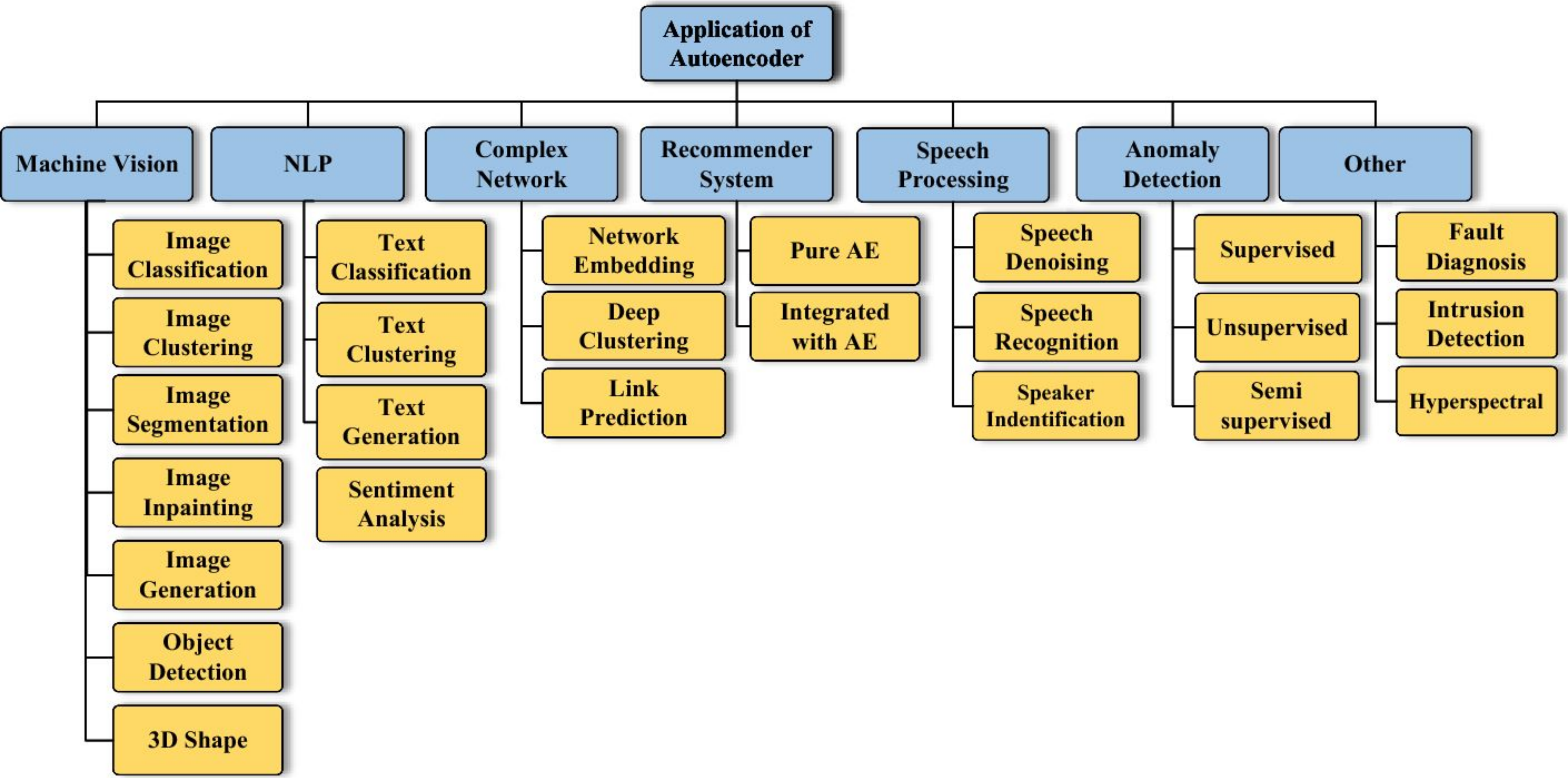


Autoencoder

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

An autoencoder neural network is an Unsupervised Machine learning algorithm that applies backpropagation, setting the target values to be equal to the inputs.

Autoencoders are used to reduce the size of our inputs into a smaller representation. If anyone needs the original data, they can reconstruct it from the compressed data.

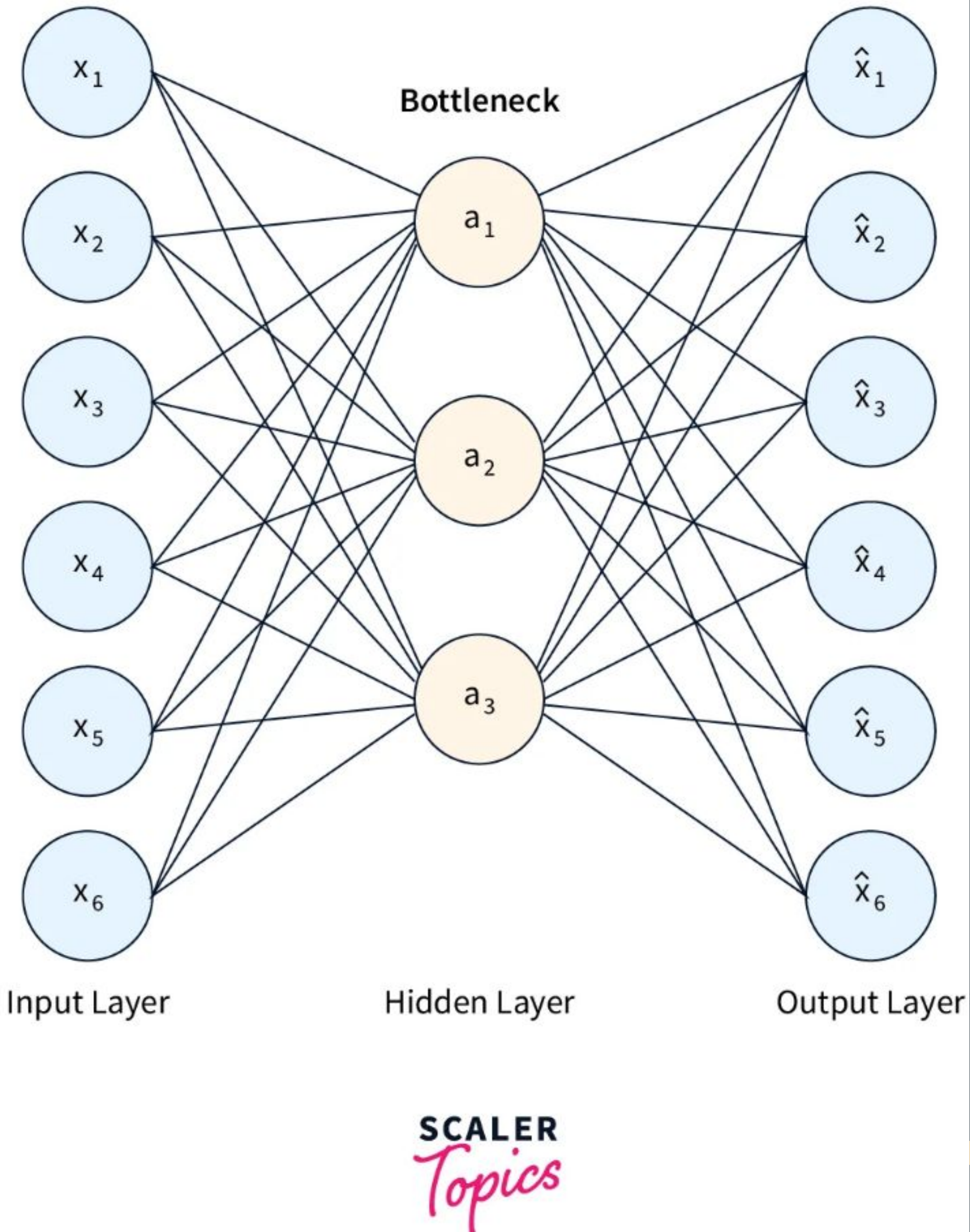

Application of Autoencoder
Machine Vision
NLP
Complex Network
Recommender System
Speech Processing
Anomaly Detection
Other
Image Classification
Image Clustering
Image Segmentation
Image Inpainting
Image Generation
Object Detection
3D Shape
Text Classification
Text Clustering
Text Generation
Sentiment Analysis
Network Embedding
Deep Clustering
Link Prediction
Pure AE
Integrated with AE
Speech Denoising
Speech Recognition
Speaker Indentification
Supervised
Unsupervised
Semi supervised
Fault Diagnosis
Intrusion Detection
Hyperspectral

Autoencoder

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- An autoencoder can learn non-linear transformations with a non-linear activation function and multiple layers.
- It doesn't have to learn using dense layers. It can use convolutional layers to learn which is better for video, image and series data.
- It is more efficient to learn with several layers using an autoencoder rather than learn one huge transformation with PCA.
- An autoencoder provides a representation of each layer as the output.
- It can make use of pre-trained layers from another model to apply transfer learning to enhance the encoder/decoder

Autoencoder



Encoder

2.1 Encoder

The encoder is a function f that maps the input $\mathbf{x}$ to the latent representation $\mathbf{z}$:

$$\mathbf{z} = f(\mathbf{x}; \mathbf{W}_e, \mathbf{b}_e)$$

where $\mathbf{W}_e$ and $\mathbf{b}_e$ are the weights and biases of the encoder, respectively. For a simple autoencoder, the encoder is often a linear transformation followed by a non-linear activation function:

$$\mathbf{z} = \sigma(\mathbf{W}_e\mathbf{x} + \mathbf{b}_e)$$

where σ is a non-linear activation function such as ReLU or Sigmoid.

Decoder

2.2 Decoder

The decoder is a function g that maps the latent representation $\mathbf{z}$ back to the original input space:

$$\hat{\mathbf{x}} = g(\mathbf{z}; \mathbf{W}_d, \mathbf{b}_d)$$

where $\mathbf{W}_d$ and $\mathbf{b}_d$ are the weights and biases of the decoder, respectively. Similar to the encoder, the decoder often uses a linear transformation followed by a non-linear activation function:

$$\hat{\mathbf{x}} = \sigma(\mathbf{W}_d\mathbf{z} + \mathbf{b}_d)$$

Loss Function

2.3 Loss Function

The loss function measures the difference between the input $\mathbf{x}$ and the reconstructed output $\hat{\mathbf{x}}$. A common choice for the loss function is the Mean Squared Error (MSE):

$$L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{x}_i)^2$$

where n is the number of features in the input vector. The objective is to minimize this loss function:

$$\min_{\mathbf{W}_e, \mathbf{b}_e, \mathbf{W}_d, \mathbf{b}_d} L(\mathbf{x}, \hat{\mathbf{x}})$$

Train the Autoencoder

Feed Data: Pass the training data through the model.

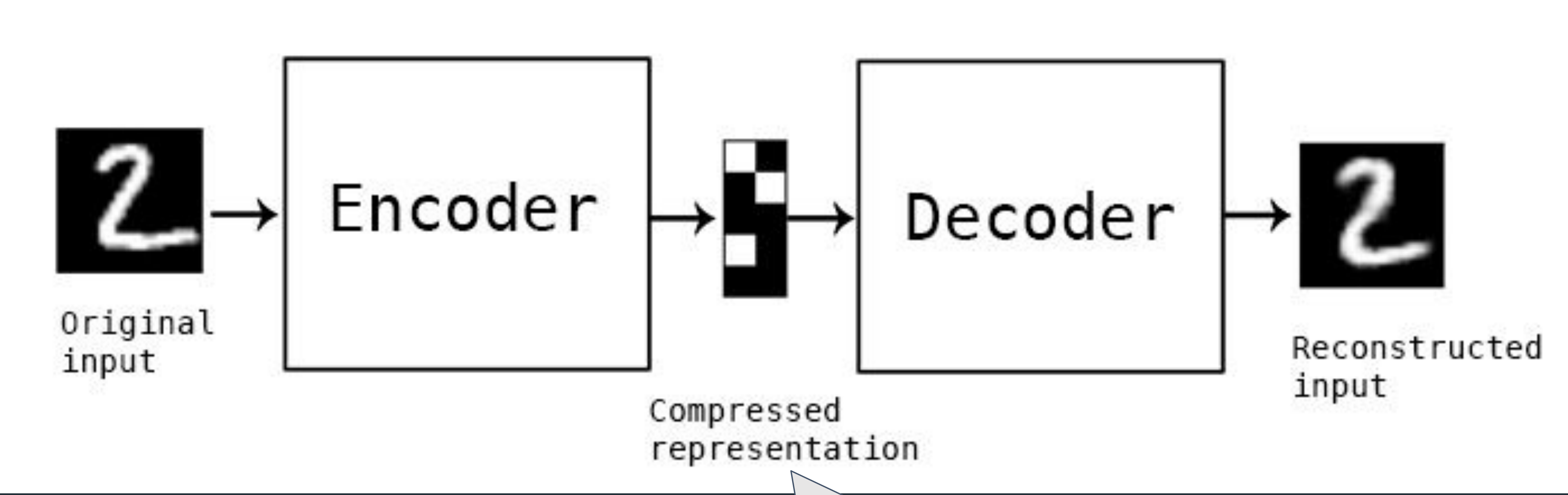
Forward Pass: Compute the output of the encoder and decoder.

Compute Loss: Measure the reconstruction loss between the original input and the reconstructed output.

Backward Pass: Compute gradients and update model parameters using the optimizer.

Epochs: Repeat the process for multiple epochs.

Autoencoder



Lower resolution
(latent representation)

MNIST dataset

Loss function

- This is unsupervised learning
  - Not matching values to labels as done in classification
  - Here, labels are the pixel outputs from the decoder
    - We want them to match the input labels
    - Probability of the pixel representing the value of white
      - Value 0 is black; 1 is white; in-between values are greyscale and so are probabilities

Handson Session 1

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

MNIST Autoencoder

Types of Autoencoder

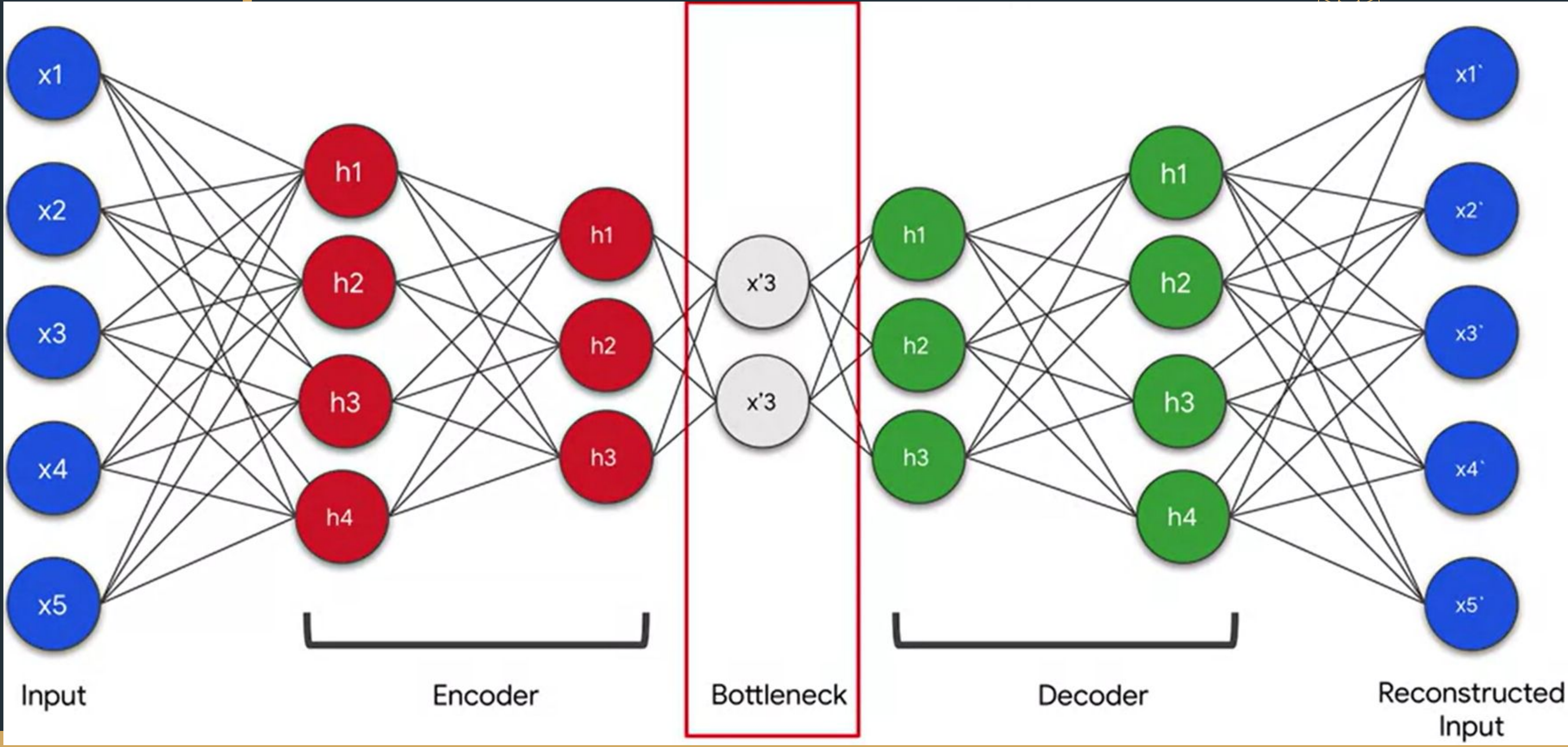
RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Vanilla Autoencoder(simple autoencoder)
- Deep (Stacked) Autoencoder
- CNN Autoencoder
- Denoising Autoencoder
- Variational Autoencoders (VAE)
- Adversarial Autoencoder
- Recurrent Autoencoder
- Stacked Autoencoder

Deep Autoencoder

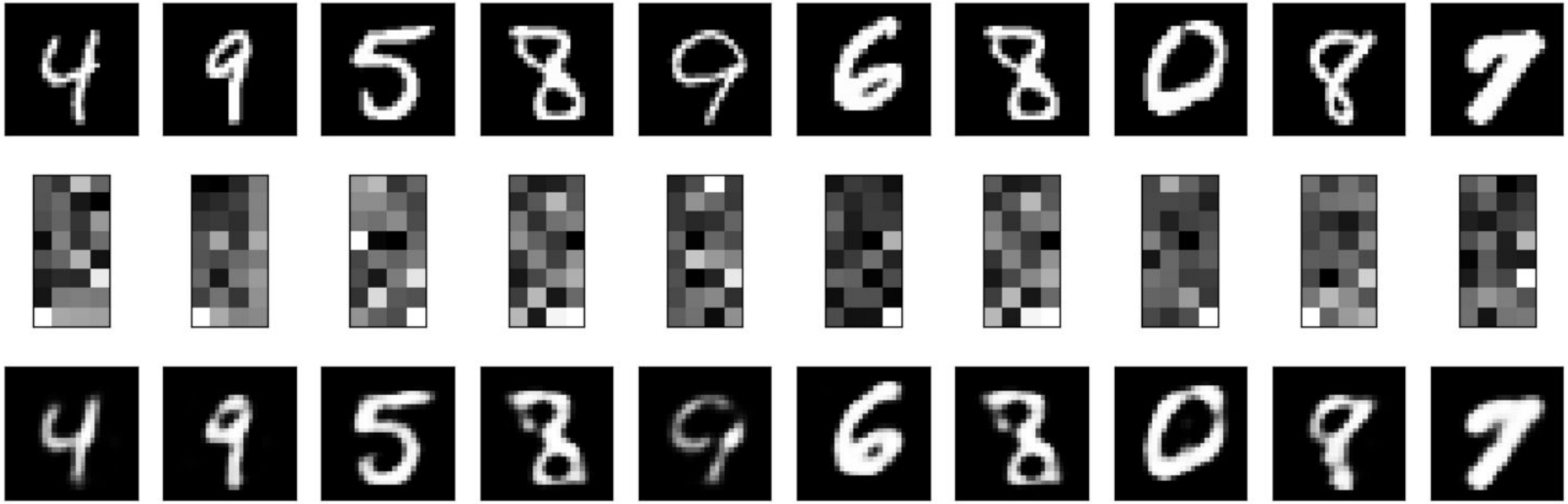
- A deep autoencoder is a single, deep neural network with multiple hidden layers that is trained end-to-end.
- Potentially harder to train due to vanishing gradient problems, especially with many layers.
- Suitable for dimensionality reduction, data denoising, and anomaly detection.

Deep Autoencoder

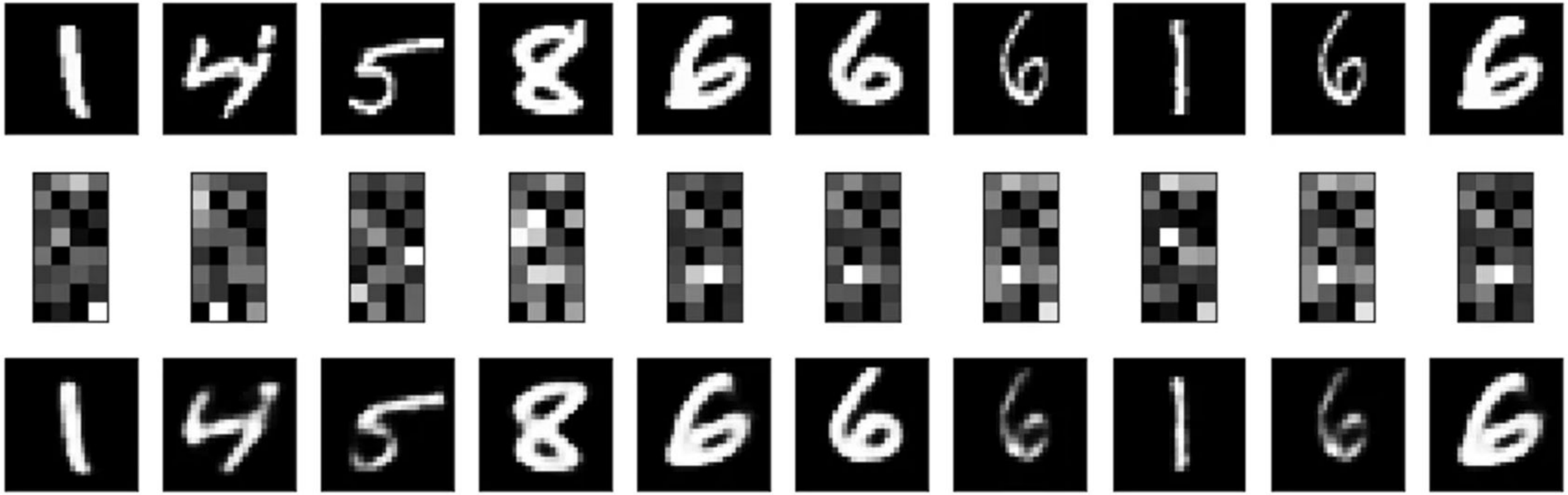


```python
def simple_autoencoder(inputs):
  encoder = tf.keras.layers.Dense(units=32, activation='relu')(inputs)
  decoder = tf.keras.layers.Dense(units=784, activation='sigmoid')(encoder)
  return encoder, decoder
```

```python
def deep_autoencoder():
 encoder = tf.keras.layers.Dense(units=128, activation='relu')(inputs)
 encoder = tf.keras.layers.Dense(units=64, activation='relu')(encoder)
 encoder = tf.keras.layers.Dense(units=32, activation='relu')(encoder)

 decoder = tf.keras.layers.Dense(units=64, activation='relu')(encoder)
 decoder = tf.keras.layers.Dense(units=128, activation='relu')(decoder)
 decoder = tf.keras.layers.Dense(units=784, activation='sigmoid')(decoder)

 return encoder, decoder

deep_encoder_output, deep_autoencoder_output = deep_autoencoder()

deep_encoder_model = tf.keras.Model(inputs=inputs, outputs=deep_encoder_output)
deep_autoencoder_model = tf.keras.Model(inputs=inputs, outputs=deep_autoencoder_output)
```



Simple encoder



Deep encoder

Deep autoencoder

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Probably hitting law of diminishing returns on the MNIST dataset at this point
- With more complex images, this technique becomes essential

Handson Session-2

MNIST_DeepAutoencoder

Convolutional Autoencoder

Convolutional Autoencoder

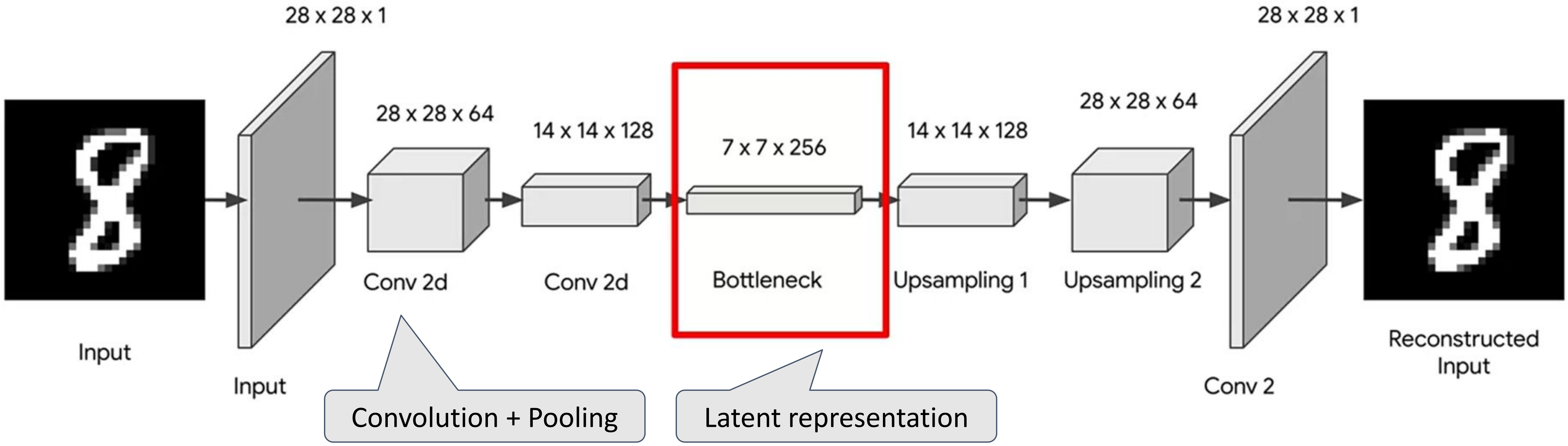
RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- A Convolutional Autoencoder (CNN Autoencoder) is a type of autoencoder that uses convolutional layers to encode and decode the input data.
- This is particularly useful for image data, where spatial hierarchies and patterns are important.
- It combines the structure of a convolutional neural network (CNN) with the autoencoder architecture, making it particularly effective for tasks like image compression, denoising, and feature extraction.

DNN Autoencoder

CNN Autoencoder

Convolutional Auto-Encoders



Encoder

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- The Encoder part of a Convolutional Autoencoder transforms the input image into a lower-dimensional representation, or latent space, by applying a series of convolutional and pooling layers.
- Input Layer: This is the input image, which in the case of the MNIST dataset, is a 28x28 grayscale image. We add a channel dimension to make it (28, 28, 1).
- Convolutional Layers: These layers apply convolutional filters to the input image, capturing spatial features. Each convolutional layer is followed by an activation function (usually ReLU).
- Pooling Layers: These layers reduce the spatial dimensions (height and width) of the image, effectively downs sampling the feature maps. Max Pooling is a common choice.
- Latent Space Representation: The output of the final convolutional layer in the encoder serves as the compressed representation (latent space) of the input image.

```
def encoder(inputs):
  conv_1 = tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3),
                                  activation='relu', padding='same')(inputs)

  max_pool_1 = tf.keras.layers.MaxPooling2D(pool_size=(2,2))(conv_1)

  conv_2 = tf.keras.layers.Conv2D(filters=128, kernel_size=(3,3),
                                  activation='relu', padding='same')(max_pool_1)

  max_pool_2 = tf.keras.layers.MaxPooling2D(pool_size=(2,2))(conv_2)

  return max_pool_2
```

```python
def bottle_neck(inputs):
  bottle_neck = tf.keras.layers.Conv2D(filters=256, kernel_size=(3,3),
                                       activation='relu', padding='same')(inputs)


  encoder_visualization = tf.keras.layers.Conv2D(filters=1, kernel_size=(3,3),
                                                 activation='sigmoid',
                                                 padding='same')(bottle_neck)

  return bottle_neck, encoder_visualization
```

To view the internal representation

```python
def decoder(inputs):
  conv_1 = tf.keras.layers.Conv2D(filters=128, kernel_size=(3,3),
                                  activation='relu', padding='same')(inputs)
  up_sample_1 = tf.keras.layers.UpSampling2D(size=(2,2))(conv_1)

  conv_2 = tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3),
                                  activation='relu', padding='same')(up_sample_1)
  up_sample_2 = tf.keras.layers.UpSampling2D(size=(2,2))(conv_2)

  conv_3 = tf.keras.layers.Conv2D(filters=1, kernel_size=(3,3),
                                  activation='sigmoid',
                                  padding='same')(up_sample_2)


  return conv_3
```

Decoder

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Up sampling Layers: These layers increase the spatial dimensions of the feature maps. Upsampling can be done using various techniques such as nearest neighbor or bilinear interpolation.

Deconvolutional (Transpose Convolution) Layers: These layers apply convolutional filters in a way that increases the spatial dimensions of the input, effectively reversing the convolution operation performed in the encoder.

Output Layer: The final layer of the decoder reconstructs the image. For grayscale images, a sigmoid activation function is often used to ensure the pixel values are between 0 and 1

```python
def convolutional_auto_encoder():
  inputs = tf.keras.layers.Input(shape=(28, 28, 1,))

  encoder_output = encoder(inputs)

  bottleneck_output, encoder_visualization = bottle_neck(encoder_output)

  decoder_output = decoder(bottleneck_output)

  model = tf.keras.Model(inputs =inputs, outputs=decoder_output)
  encoder_model = tf.keras.Model(inputs=inputs, outputs=encoder_visualization)

  return model, encoder_model
```

Handson Session-3

CNN Autoencoder

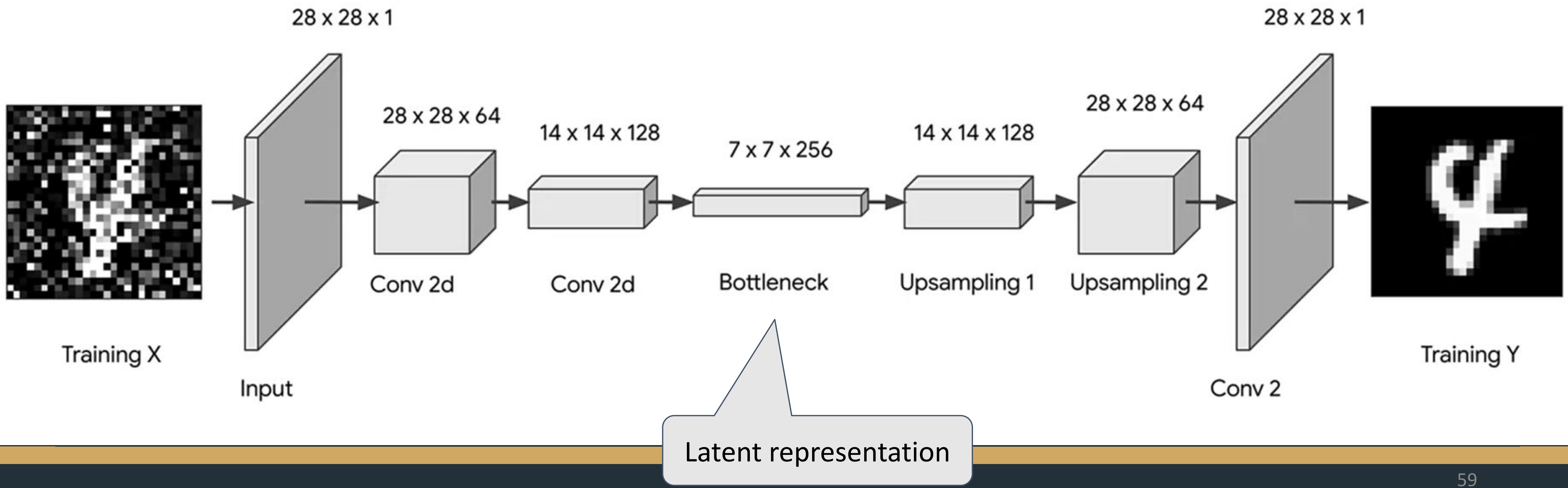
Denoising Autoencoder

A Denoising Autoencoder (DAE) is a type of autoencoder that is trained to remove noise from its input.

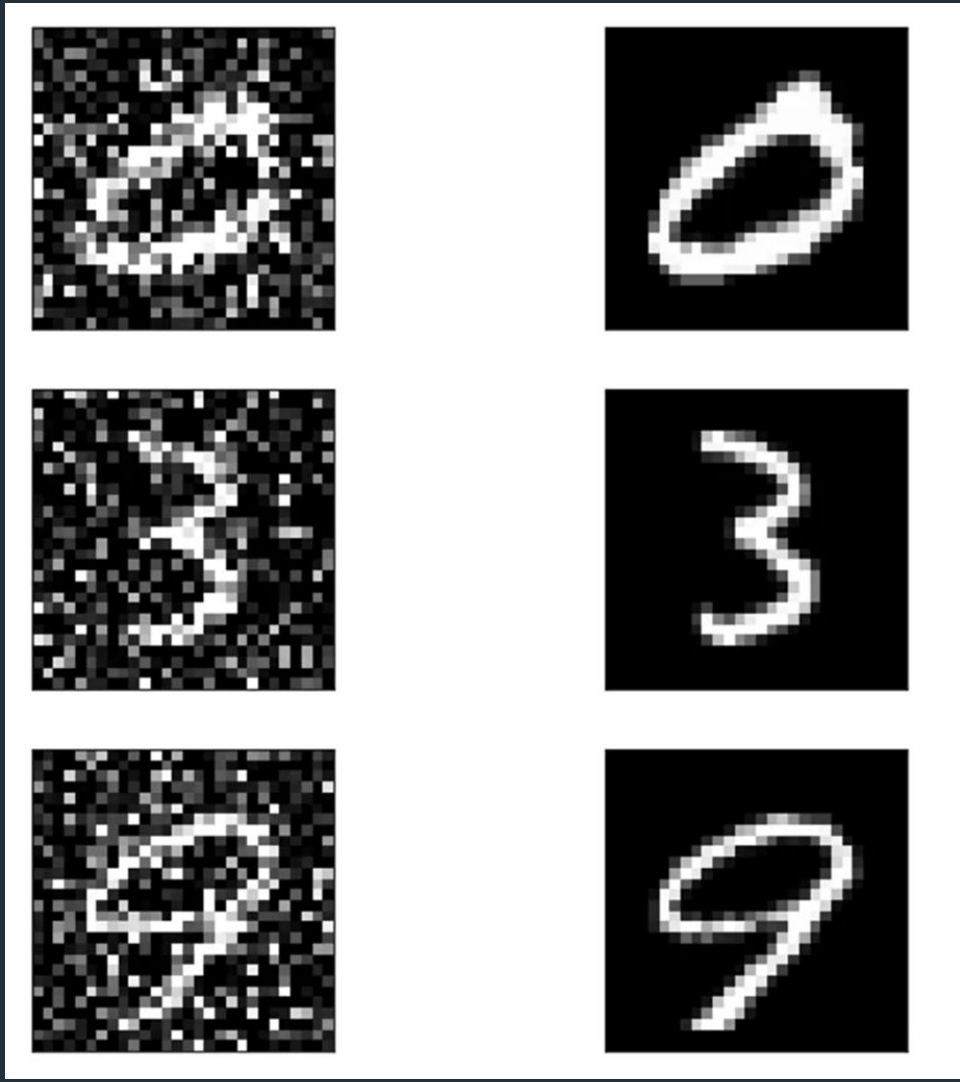
It is a variant of the traditional autoencoder, but instead of learning to reconstruct the exact input, it learns to reconstruct the original input from a corrupted version of it.

Denoising Autoencoder

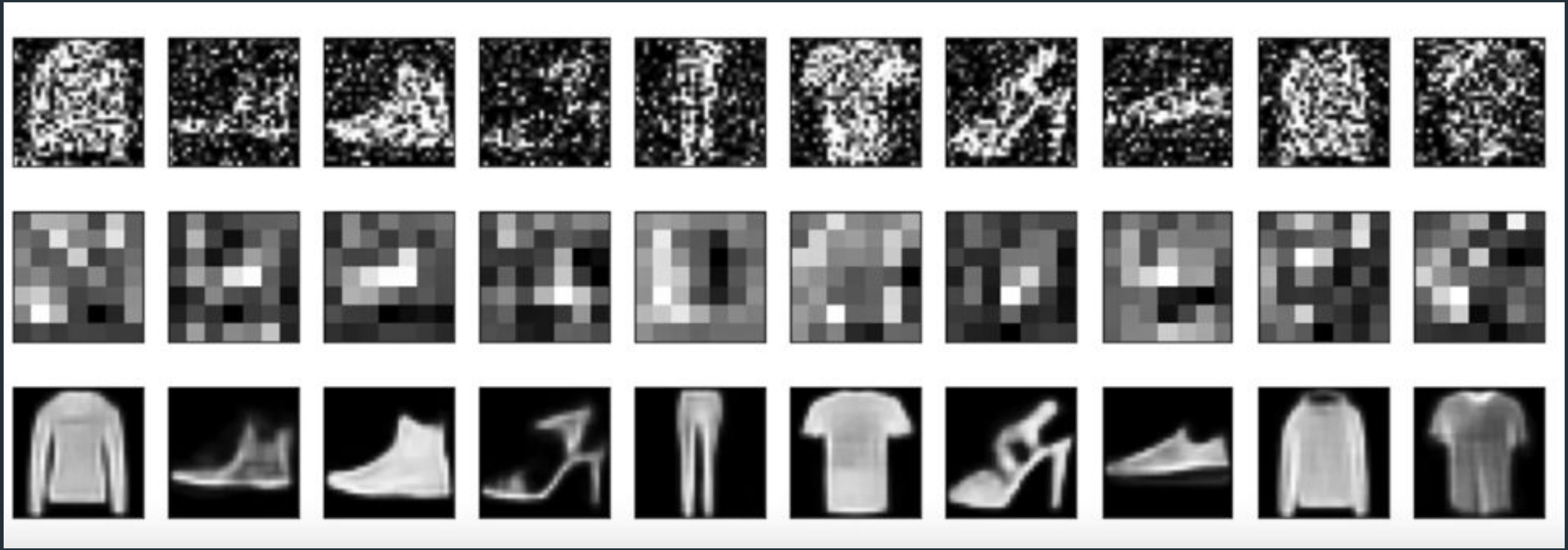
Convolutional Auto-Encoders



Denoising Autoencoder



Denoising Autoencoder



Denoising Autoencoder


```
def map_image_with_noise(image, label):
  noise_factor = 0.5
  image = tf.cast(image, dtype=tf.float32)
  image = image / 255.0

  factor = noise_factor * tf.random.normal(shape=image.shape)
  image_noisy = image + factor
  image_noisy = tf.clip_by_value(image_noisy, 0.0, 1.0)

  return image_noisy, image
```


The range

Handson Session 4

Denoising Autoencoder