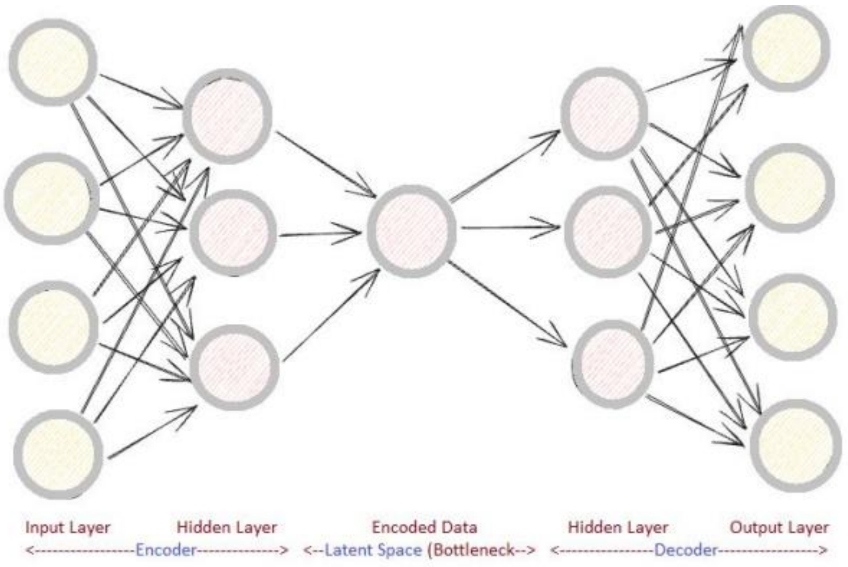
Autoencoder architecture

- Both the encoder and decoder are fully-connected feedforward NN.
- Autoencoders are trained the same way as ANNs via backpropagation.
- Code is a single layer of an ANN with the dimensionality of our choice.
- The number of nodes in the code layer (code size) is a hyperparameter that we set before training the autoencoder.


Input Layer
Hidden Layer
Encoded Data
Hidden Layer
Output Layer
<-----------------Encoder-------------->
<--Latent Space (Bottleneck-->
<----------------Decoder---------------->

- First the input passes through the encoder, which is a fully-connected ANN, to produce the code.
- The decoder, which has the similar ANN structure, then produces the output only using the code.
- The goal is to get an output identical with the input.
- Note that the decoder architecture is the mirror image of the encoder.
- This is not a requirement but it's typically the case.
- The only requirement is the dimensionality of the input and output needs to be the same.\
- Anything in the middle can be played with.

Hyperparameters:

- Code size: number of nodes in the middle layer. Smaller size results in more compression.

- Number of layers: the autoencoder can be as deep as we like. In the figure above we have 2 layers in both the encoder and decoder, without considering the input and output.

- Number of nodes per layer: The autoencoder architecture is called a stacked autoencoder since the layers are stacked one after another.

- Loss function: We either use mean squared error (mse) or binary crossentropy.
  - If the input values are in the range [0, 1] then we typically use crossentropy,
  - Otherwise we use the mean squared error.

Assumptions:

- We can make autoencoder very powerful by increasing # layers, nodes per layer and most importantly the code size
- Increasing these hyperparameters will let the autoencoder to learn more complex codings.
- Be careful while making it too powerful. Then it will
  - Simply learn to copy its inputs to the output, without learning any meaningful representation.
  - Just mimic the identity function.
  - Reconstruct training data perfectly, but result in overfitting without being able to generalize to new instances.
- This is why we prefer a “sandwitch” architecture, and deliberately keep the code size small.
- Since the coding layer has a lower dimensionality than the input data, the autoencoder is said to be undercomplete.
- It won’t be able to directly copy its inputs to the output, and will be forced to learn intelligent features.
- For example:
  --Digit “1” is somewhat straight line & digit “0” is circular, it will learn fact and encode it in compact form.
  If the input data was completely random without any internal correlation or dependency,
  --Then an undercomplete autoencoder won’t be able to recover it perfectly.