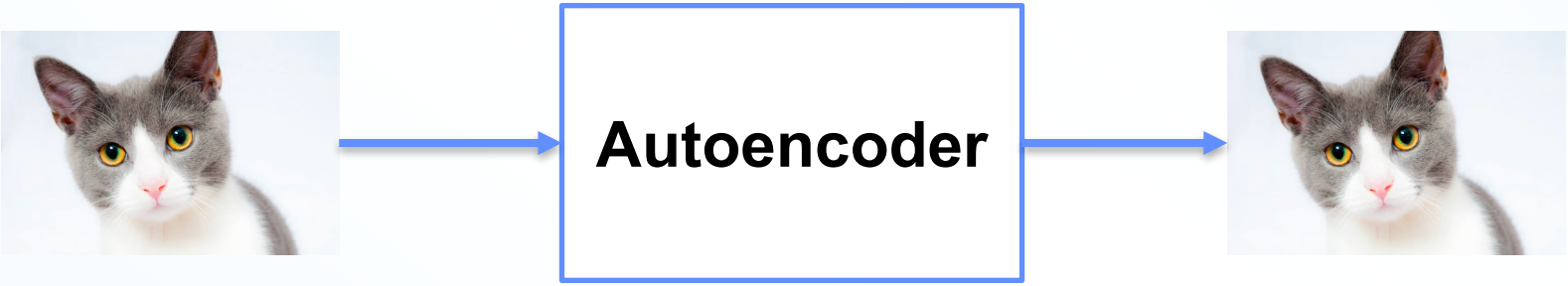
Autoencoders

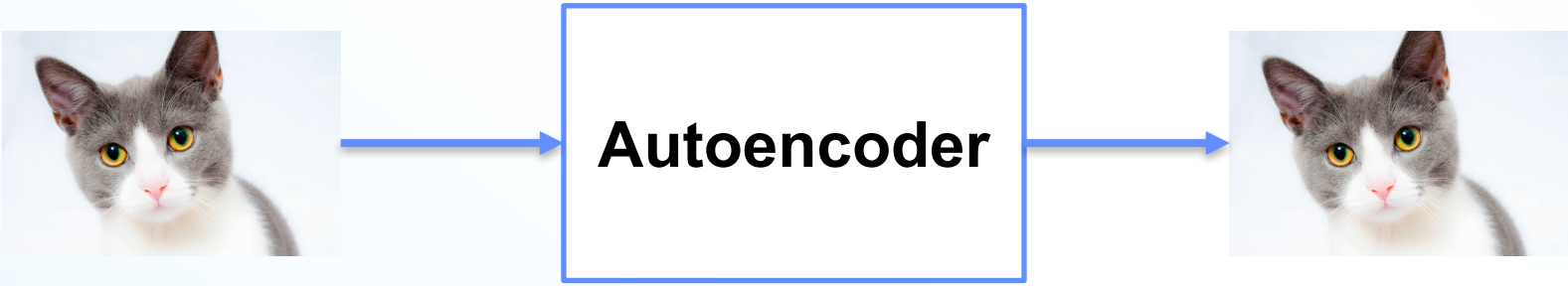
Autoencoders

- **Autoencoders are neural networks that are trained to copy their input to their output**



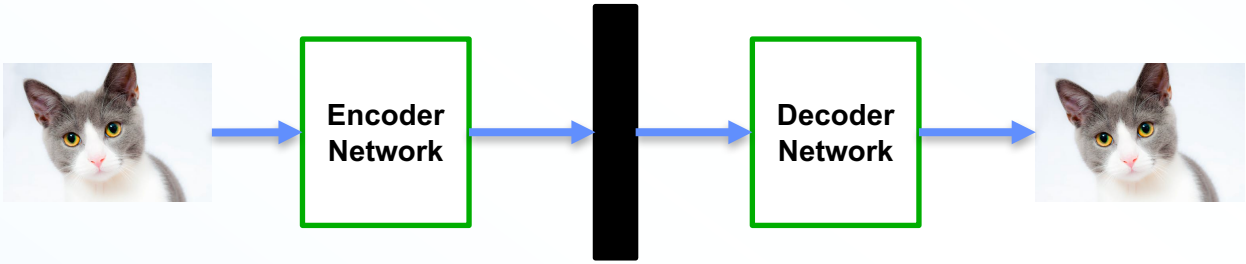
Autoencoders

- **But why do this ?**
  - **The goal is learn a hidden representation(typically lower dimensional) of the input**



Autoencoders

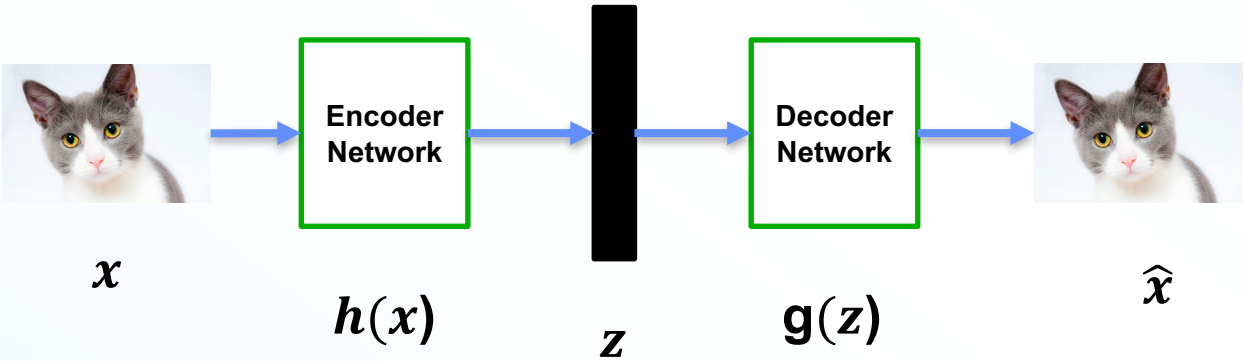
- But Why do this ?
  - The goal is to learn a hidden representation(typically lower dimensional) of the input
  - This hidden representation could act a feature or lower dimensional representation of input data



Low dimensional
hidden representation

Autoencoders

- They consist of an encoder function $h(x)$ which takes an input x to the hidden representation z and a decoder function $g(z)$ that maps the hidden representation z to the input x.



$$\min_{h,g} \sum (\hat{x} - x)^2$$

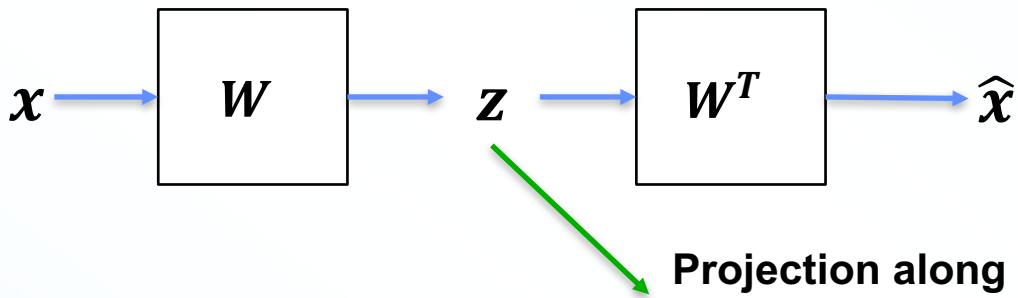
Autoencoders

- **Autoencoders can be thought of as a non linear PCA**

PCA

$x \epsilon R^n$

$z \in R^d$



Projection along d (d < n) principal components

$$\min_{W} \sum (\hat{x} - x)^2$$

$$W^T W = I$$

$$\min_{W} \sum (W^T W x - x)^2$$

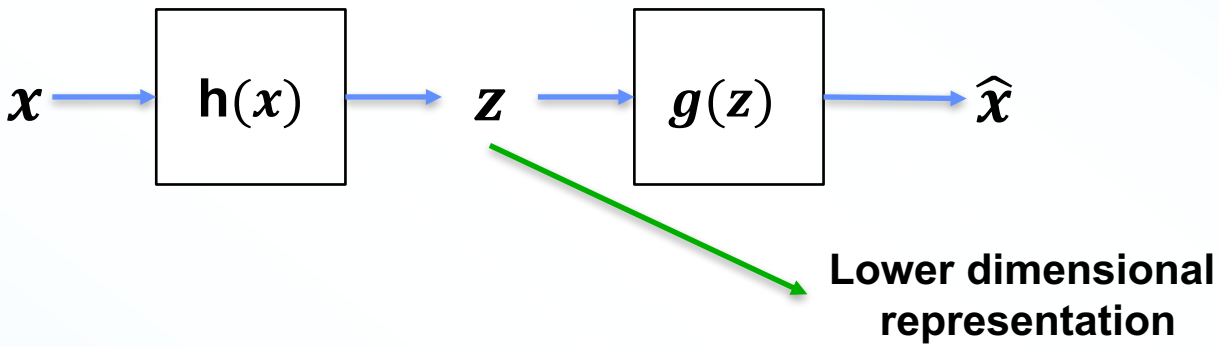
Autoencoders

- **Autoencoders can be thought of as a non linear PCA**

Non linear PCA

$x \epsilon R^n$

$z \in R^d$



$$\min_{h,g} \sum (\hat{x} - x)^2$$

$$\min_{h,g} \sum (\, g(h(x)) \, - x)^{\,2}$$

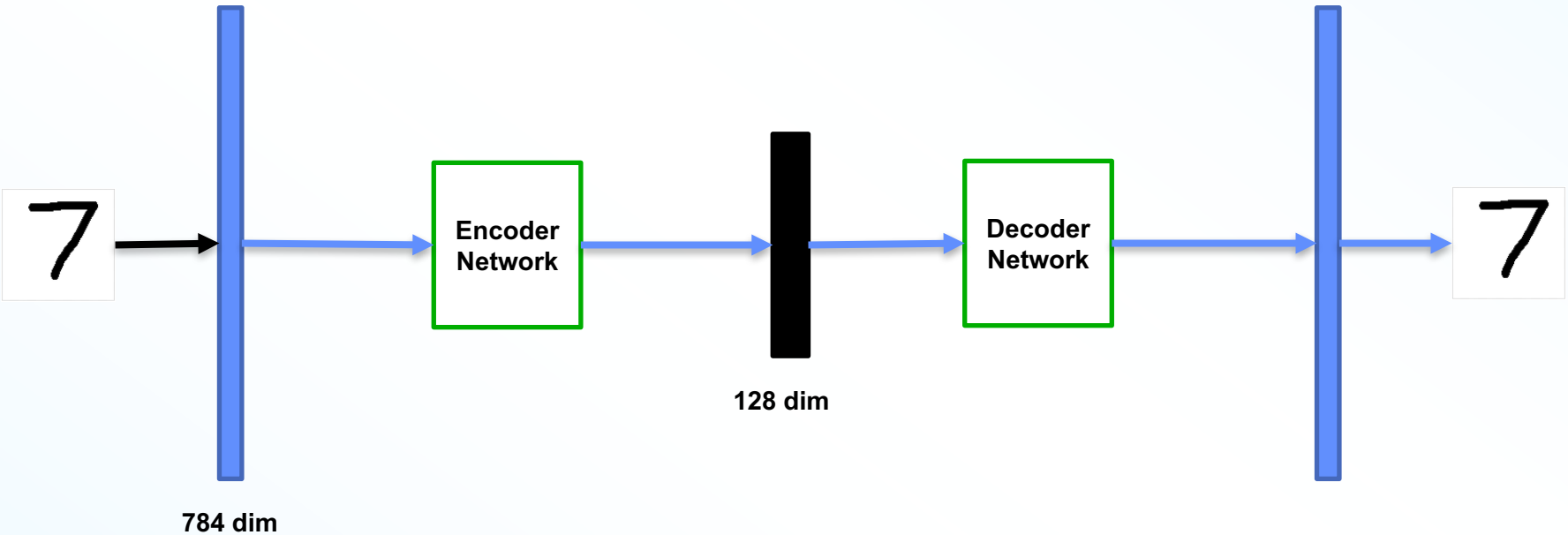
Autoencoders

- **Autoencoders are trained using gradient descent (or some other optimization algorithm) on the loss function through backprop just like any other cnn**

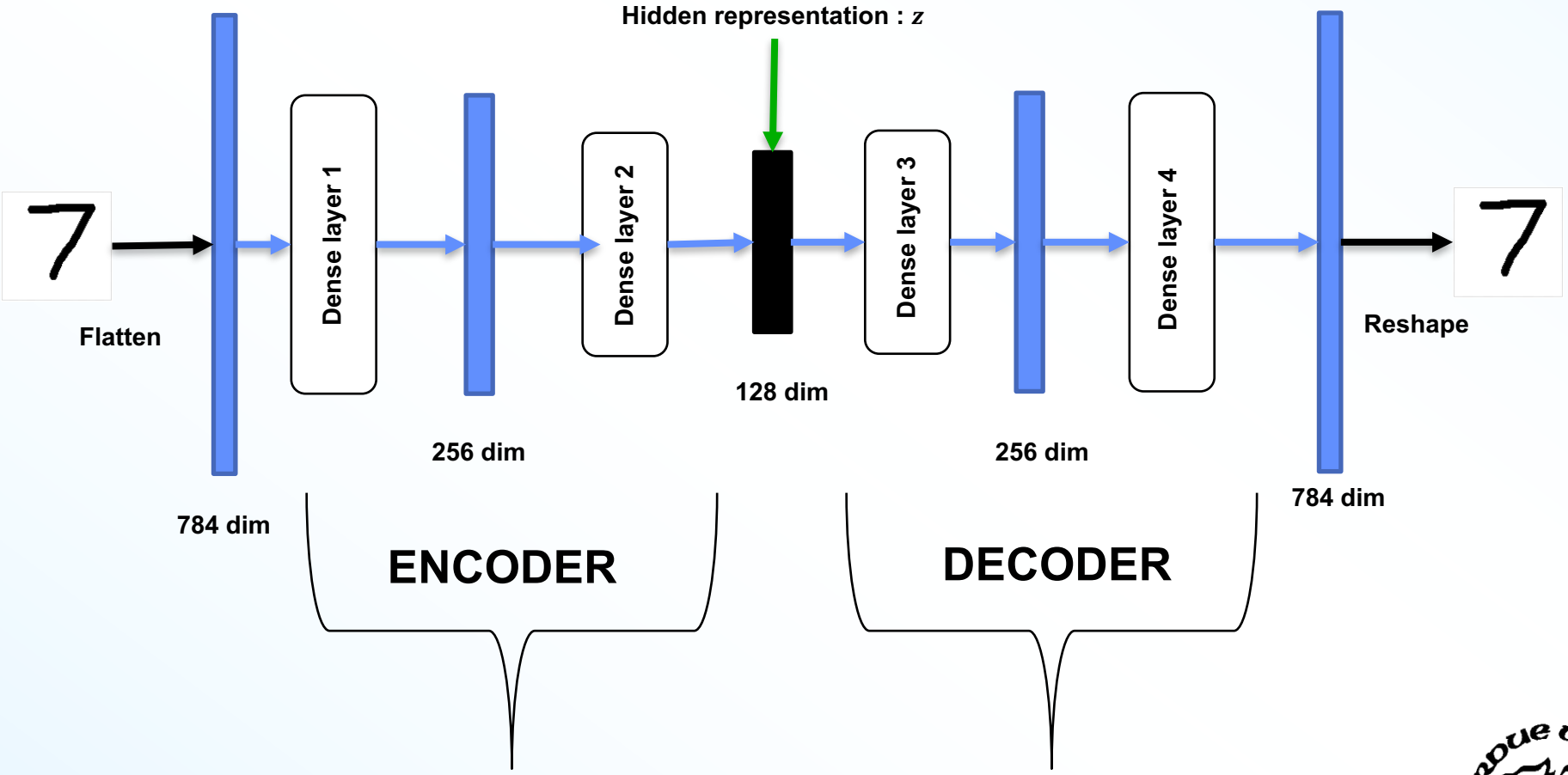
Example

- **In this session we will build an autoencoder that learns to learn a lower dimensional representation of mnist digits.**

Example



Autoencoder Structure



Autoencoder Structure

- **Dense layer 1 : transforms input of dim 784 to 256 so it will have a weight matrix of (256x784) and bias vector of length 256**

- **Dense layer 3 : transforms input of dim 128 to 256 so it will have a weight matrix of (256x128) and bias vector of length 256**

- **Assumption : Every dense layer has a sigmoid activation function**

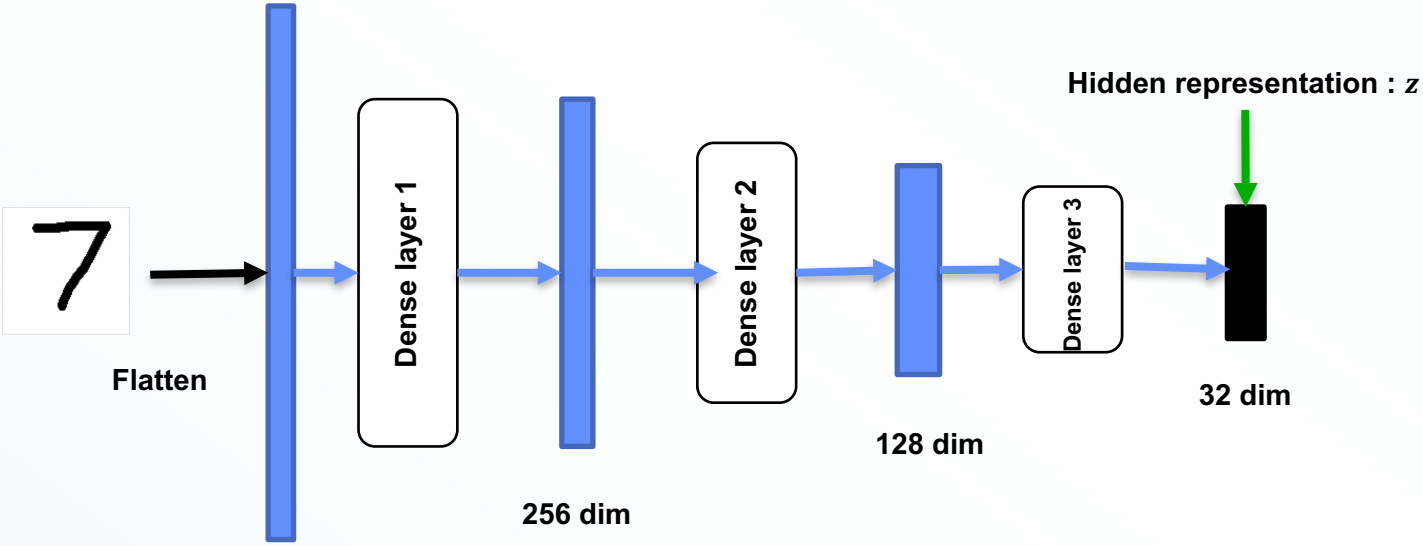
Autoencoder Architecture

- **For the demo we will use an l2 loss function and rmsprop as optimizer.**

Assignment

- **In the demo we used an l2 loss. Try running the code with an l1 loss and see the results.**

- **Once you are done with this construct an autoencoder of the with the following encoder architecture**



- **After constructing the autoencoder train it with l2 and l1 loss and visualize the results. If the results are not good try changing the hyperparameters like learning rate, optimizer etc to see if the results improve**