Unit - 5

Auto Encoders

Syllabus

- **Autoencoders:**
- Efficient data representation,
- stacked autoencoders,
- Unsupervised pretraining using SA,
- Denoising, Sparse autoencoders,
- variational and other autoencoders.

- **Reinforcement Learning:**
- Learning to optimize rewards,
- policy search,
- Neural network polices,
- Evaluating actions,
- Policy gradients,
- Markov decision processes,
- TDL and Q-learning

What is Autoencoders?

- Autoencoders are artificial neural networks capable of learning efficient representations of the input data, called coding's, without any supervision (i.e., the training set is unlabeled).

- These coding's typically have a much lower dimensionality than the input data, making autoencoders useful for dimensionality reduction.

- More importantly, autoencoders act as powerful feature detectors, and they can be used for unsupervised pretraining of deep neural networks.

- Lastly, they are capable of randomly generating new data that looks very similar to the training data; this is called a generative model. For example, you could train an autoencoder on pictures of faces, and it would then be able to generate new faces

Efficient Data Representations

- An autoencoder is always composed of two parts:

  An encoder (or recognition network) that converts the inputs to an internal representation,

  followed by a decoder (or generative network) that converts the internal representation to the outputs.

- An autoencoder typically has the same architecture as a Multi-Layer Perceptron except that the number of neurons in the output layer must be equal to the number of inputs.
- In this example, there is just one hidden Autoencoders layer composed of two neurons (the encoder), and one output layer composed of three neurons (the decoder).
- The outputs are often called the reconstructions since the autoencoder tries to reconstruct the inputs.

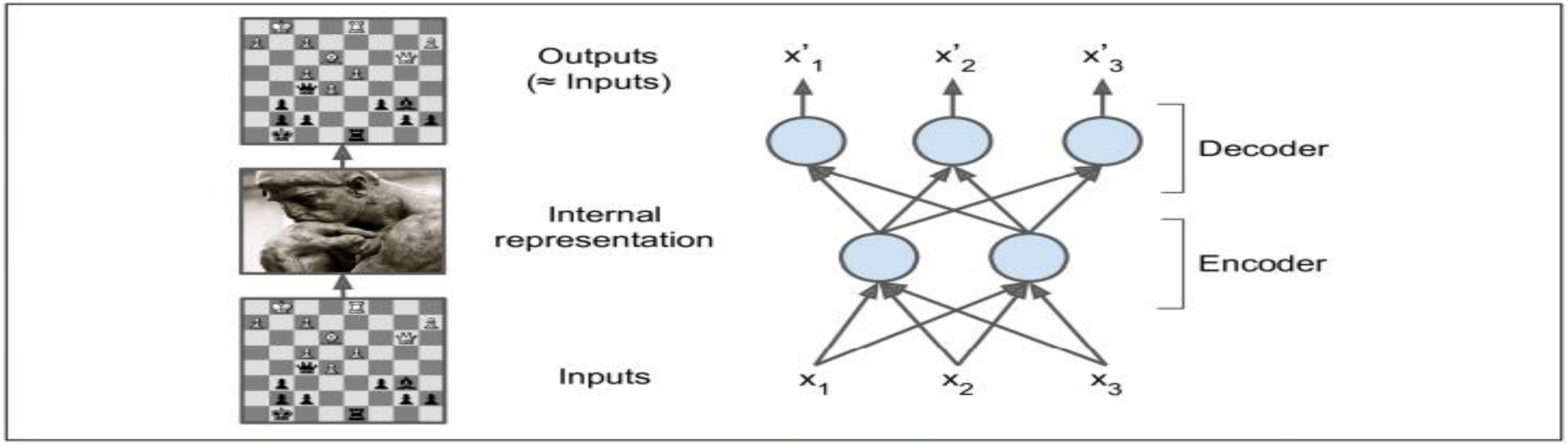
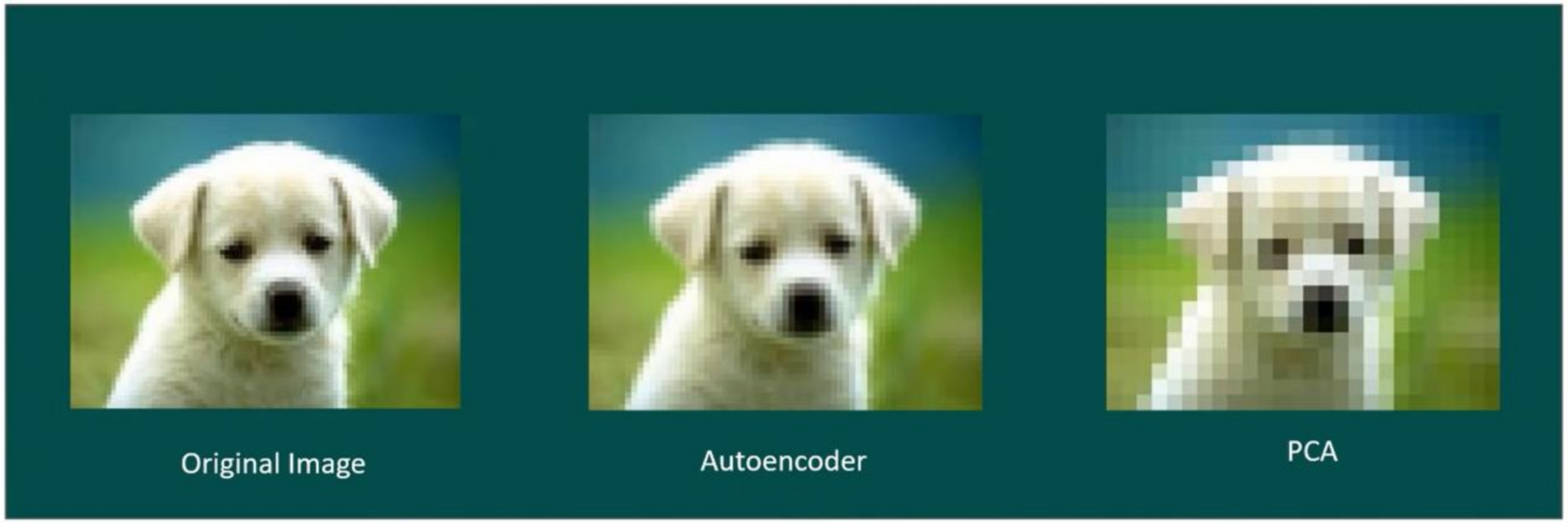


Figure 15-1. The chess memory experiment (left) and a simple autoencoder (right)

How autoencoder works?

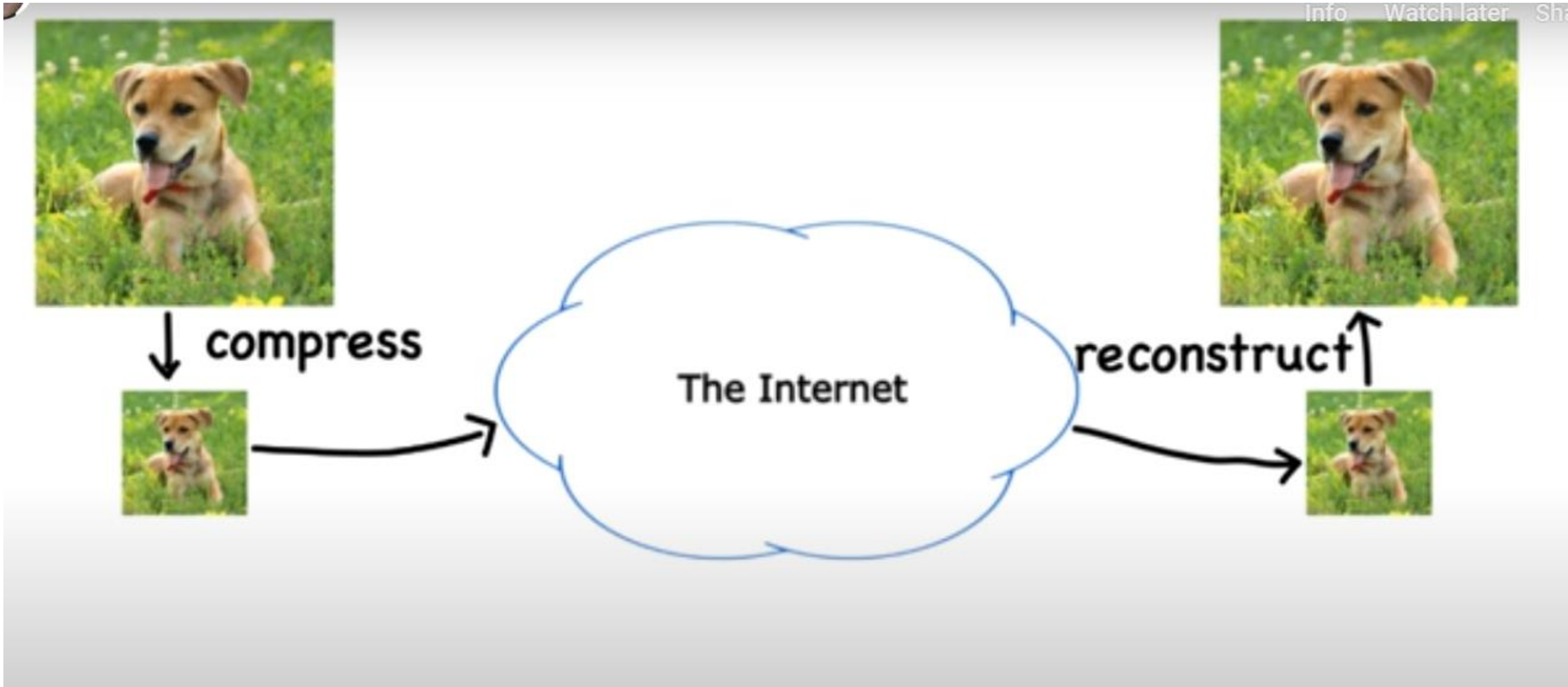
- Compression is to convert the input data into lower dimension, and later using the lower dimension data it can be reconstructed which will be close to original data.
- Autoencoders are more accurate compared to PCA



How autoencoder works?

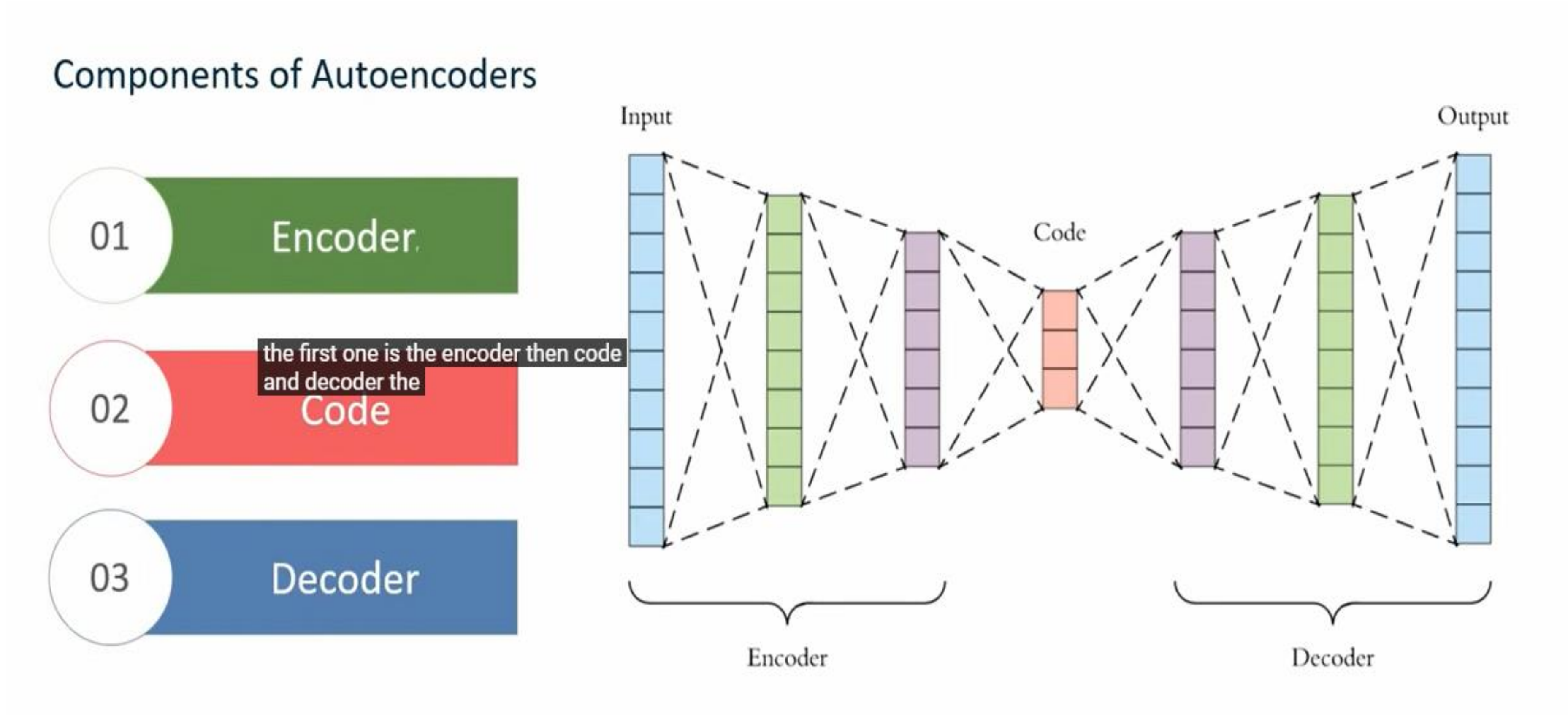
- Autoencoder are popularly used for dimensionality reduction and noise elimination.
- Autoencoders are simple network, which converts input into output with minimal possible errors.
- It is unsupervised ML algorithm that applied back propagation, which sets target value similar to input.

Image transmission over internet

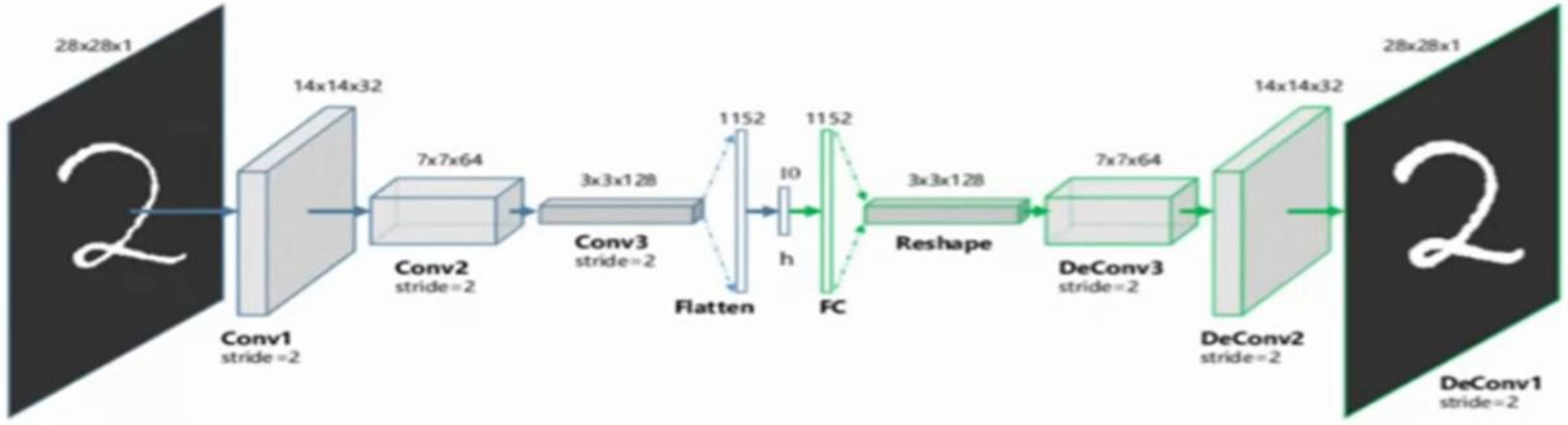
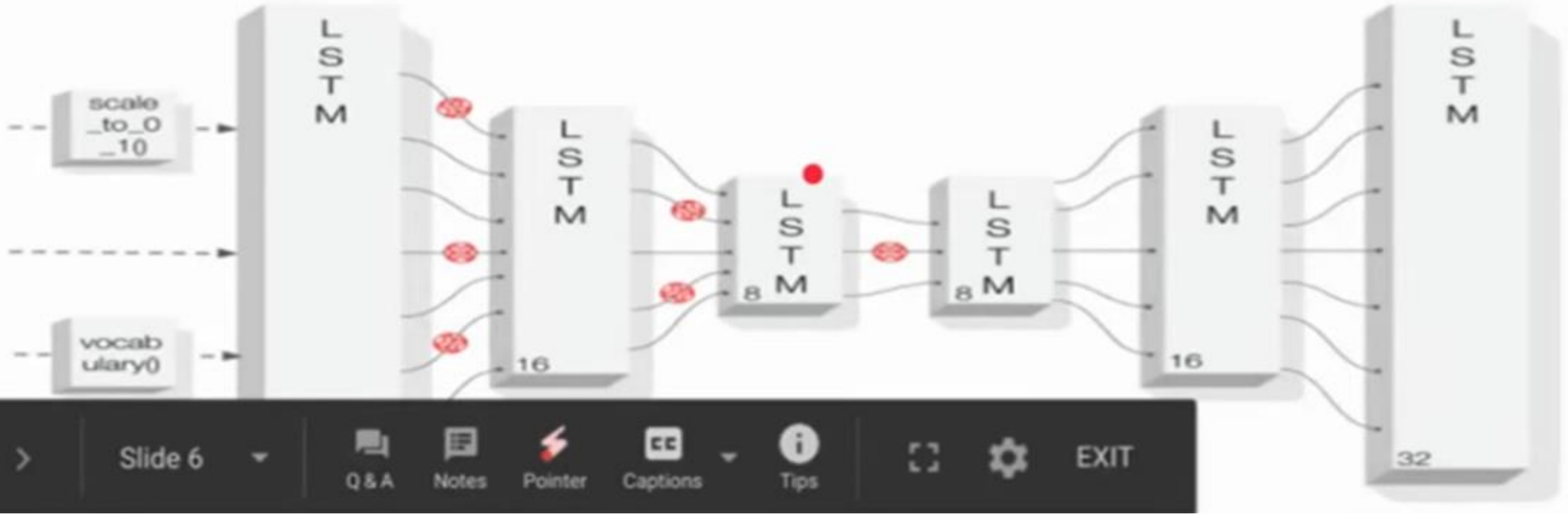
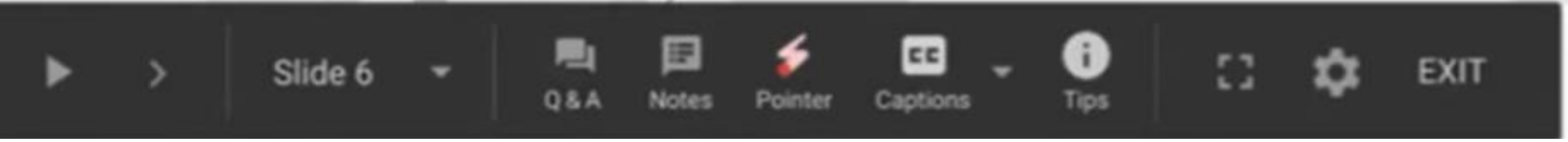


Components of Autoencoders

- Three components of Autoencoders are:
- 1. Encoder
- 2. Code
- 3. Decoder.



the first one is the encoder then code and decoder the


28x28x1
14x14x32
7x7x64
3x3x128
1152
10
h
1152
3x3x128
7x7x64
14x14x32
28x28x1
Conv1
stride=2
Conv2
stride=2
Conv3
stride=2
Flatten
FC
Reshape
DeConv3
stride=2
DeConv2
stride=2
DeConv1
stride=2

scale
_to_0
_10
vocab
ulary0
LSTM
LSTM
16
LSTM
8
LSTM
8
LSTM
16
LSTM
32

Slide 6
Q & A
Notes
Pointer
Captions
Tips
EXIT

Components

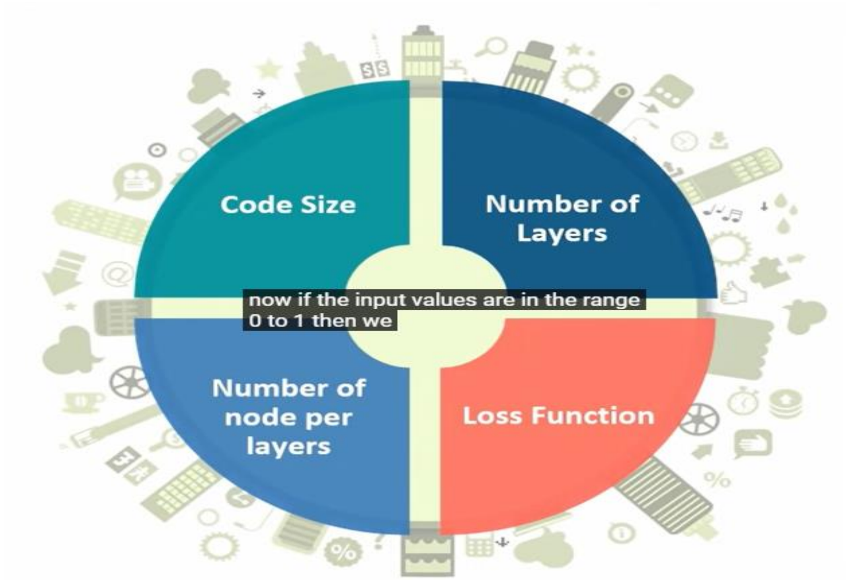
- Encoder layer compresses the input into a different representation, which is of reduced dimension.
- The next component is the code which is the compressed data, which is called latent space data.
- Decoder, decodes the code and reconstructs the original data.

Properties of Autoencoders

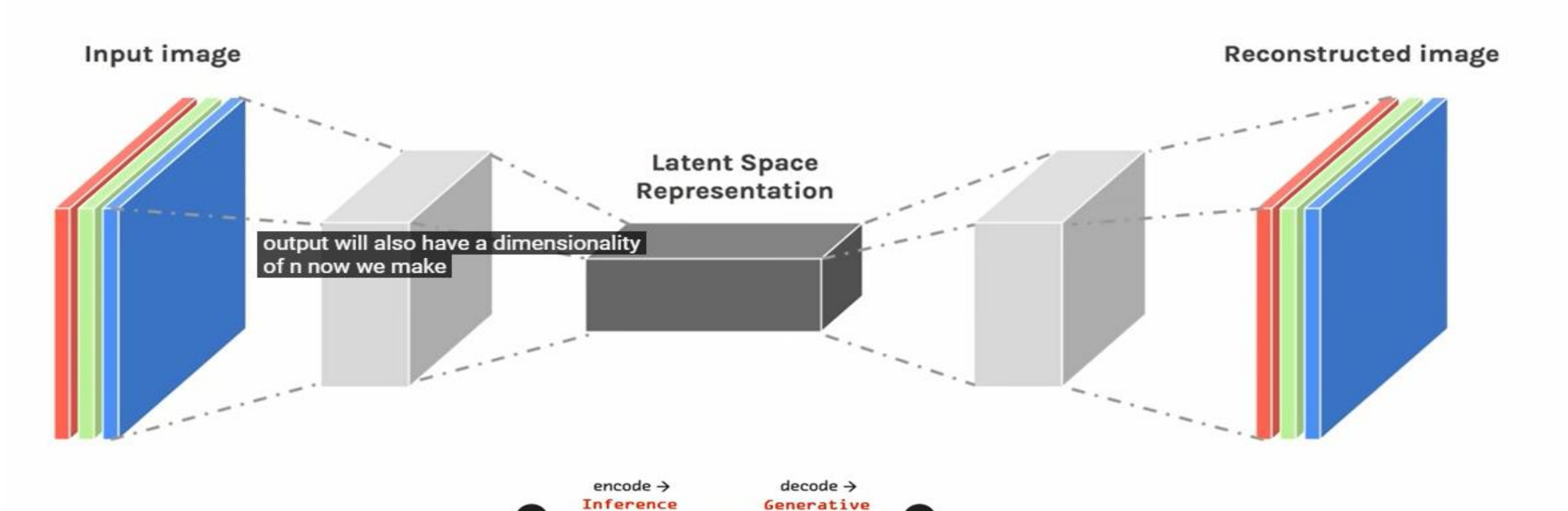
- Autoencoders are basically used for compression.
- They are unsupervised as they don’t require explicit labels to train
- Autoencoders are lossy, which means that the decompressed data will be degraded when compared to original data.

Training the autoencoders

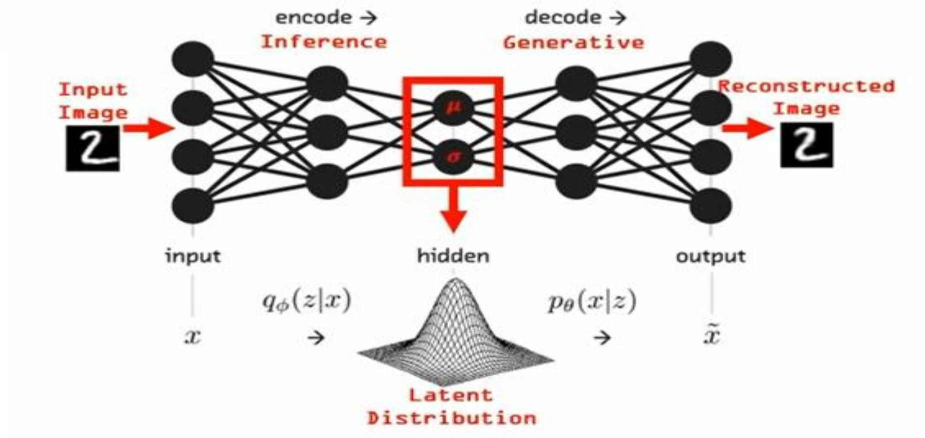
- 4 parameters are required while training the auto encoders.
  - Code Size : Data in the middle of the auto encoder after compression.
  - Number of layers : Number of neural layers used
  - Number of nodes per layer : Number of neurons per layer
  - Loss function : It can be mean squared error or binary entropy


Code Size
Number of Layers
now if the input values are in the range 0 to 1 then we
Number of node per layers
Loss Function

Architecture of Autoencoders



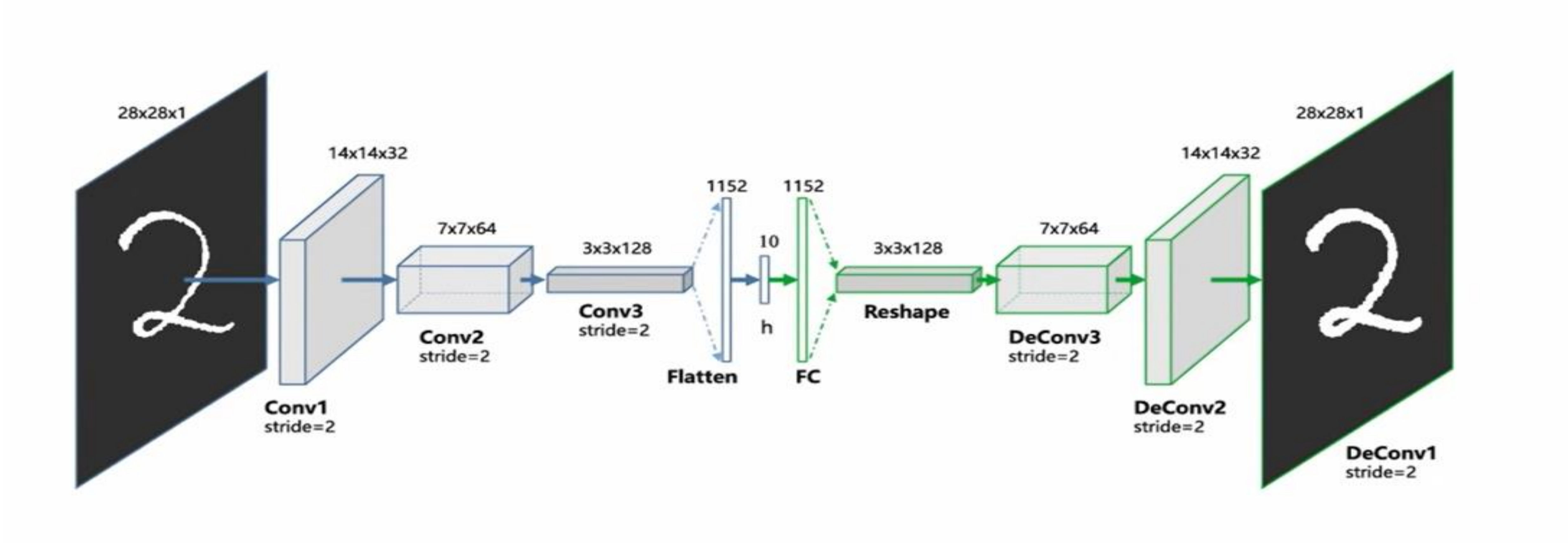
output will also have a dimensionality of n now we make



Encoder

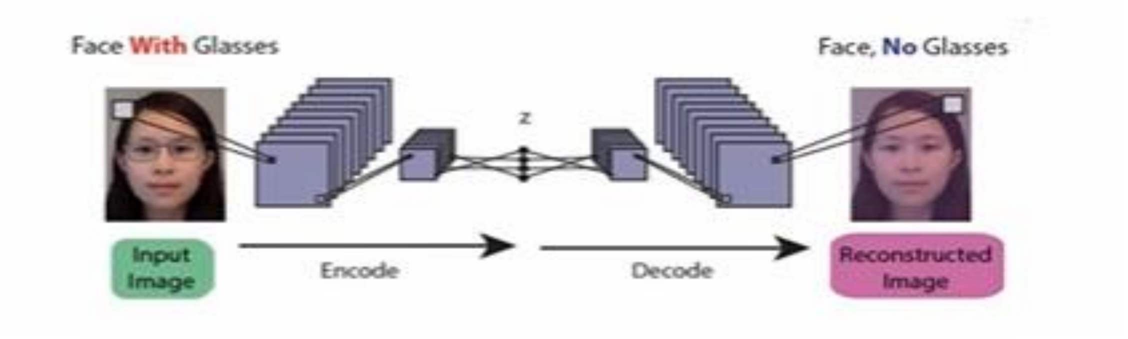
- The layers between input and output of autoencoders are of lesser dimension.
- Input if it is of 28x28 image= 784 pixels, let us say it as X.
- Its output is Z which is much smaller compared to X

Convolution Autoencoders: uses convolution operator to learn to encode the input in a set of simple singles and then try to reconstruct the input in a set of simple signals and then try to reconstruct the input from them



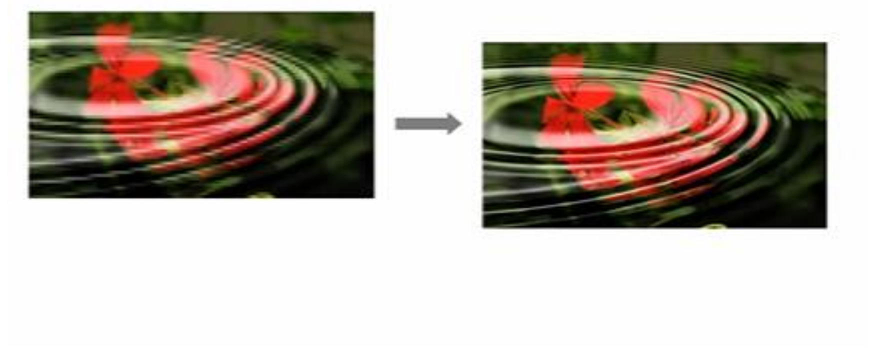
Uses of Convolution autoencoders

- 1. Image reconstruction



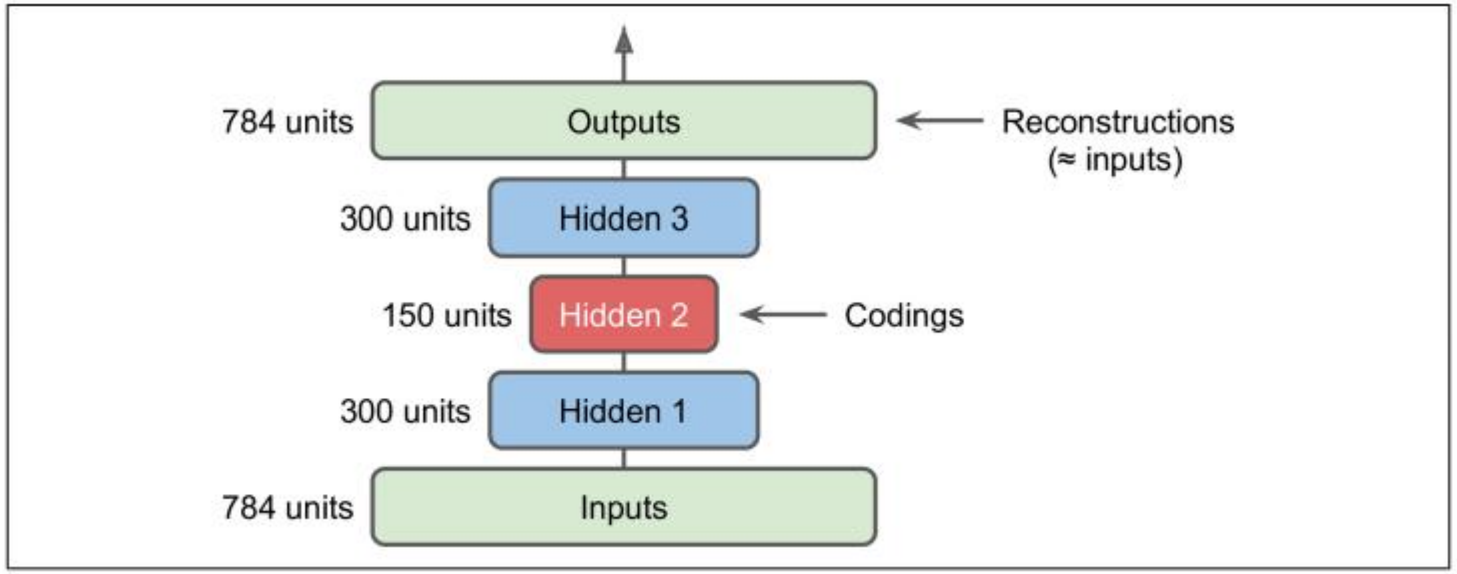
- 2. Image Colorization

- 3. Advanced application like
  generating high resolution image



Stacked Autoencoders

- Just like other neural networks we have discussed, autoencoders can have multiple hidden layers.
- In this case they are called stacked autoencoders (or deep autoencoders).
- Adding more layers helps the autoencoder learn more complex coding's.



Unsupervised Pretraining using Stacked Autoencoders

- Tackling a complex supervised task when do not have a lot of labeled training data, one solution is to find a neural network that performs a similar task, and then reuse its lower layers.

- This makes it possible to train a high-performance model using only little training data because your neural network won’t have to learn all the low-level features; it will just reuse the feature detectors learned by the existing net.

unsupervised Pretraining Using Stacked Autoencoders

- Similarly, if you have a large dataset but most of it is unlabeled, you can first train a stacked autoencoder using all the data, then reuse the lower layers to create a neural network for your actual task, and train it using the labeled data.

Unsupervised pretraining using autoencoders

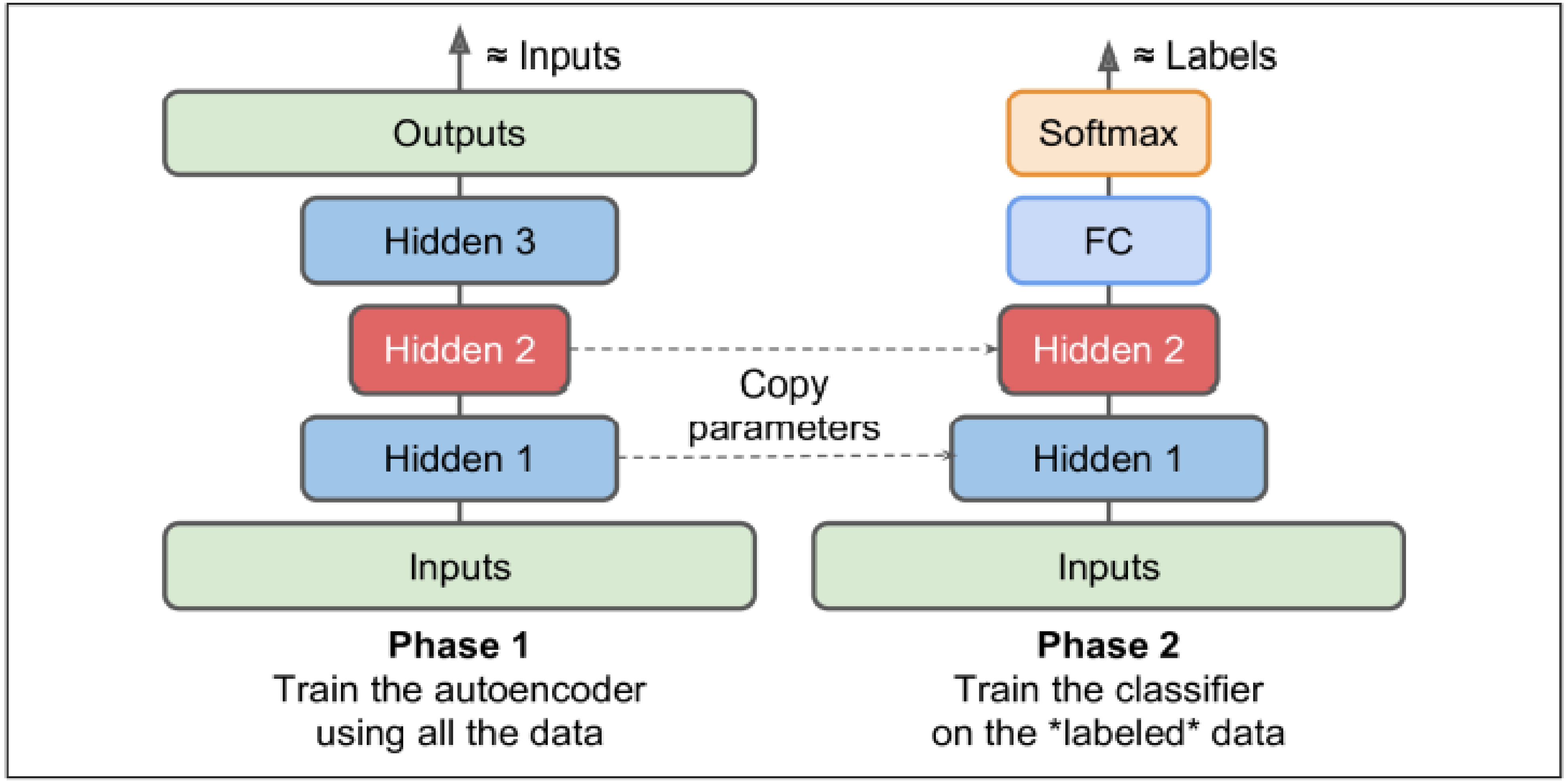


Figure 15-8. Unsupervised pretraining using autoencoders

Denoising Autoencoders

- Another way to force the autoencoder to learn useful features is to add noise to its inputs, training it to recover the original, noise-free inputs. This prevents the autoencoder from trivially copying its inputs to its outputs, so it ends up having to find patterns in the data.

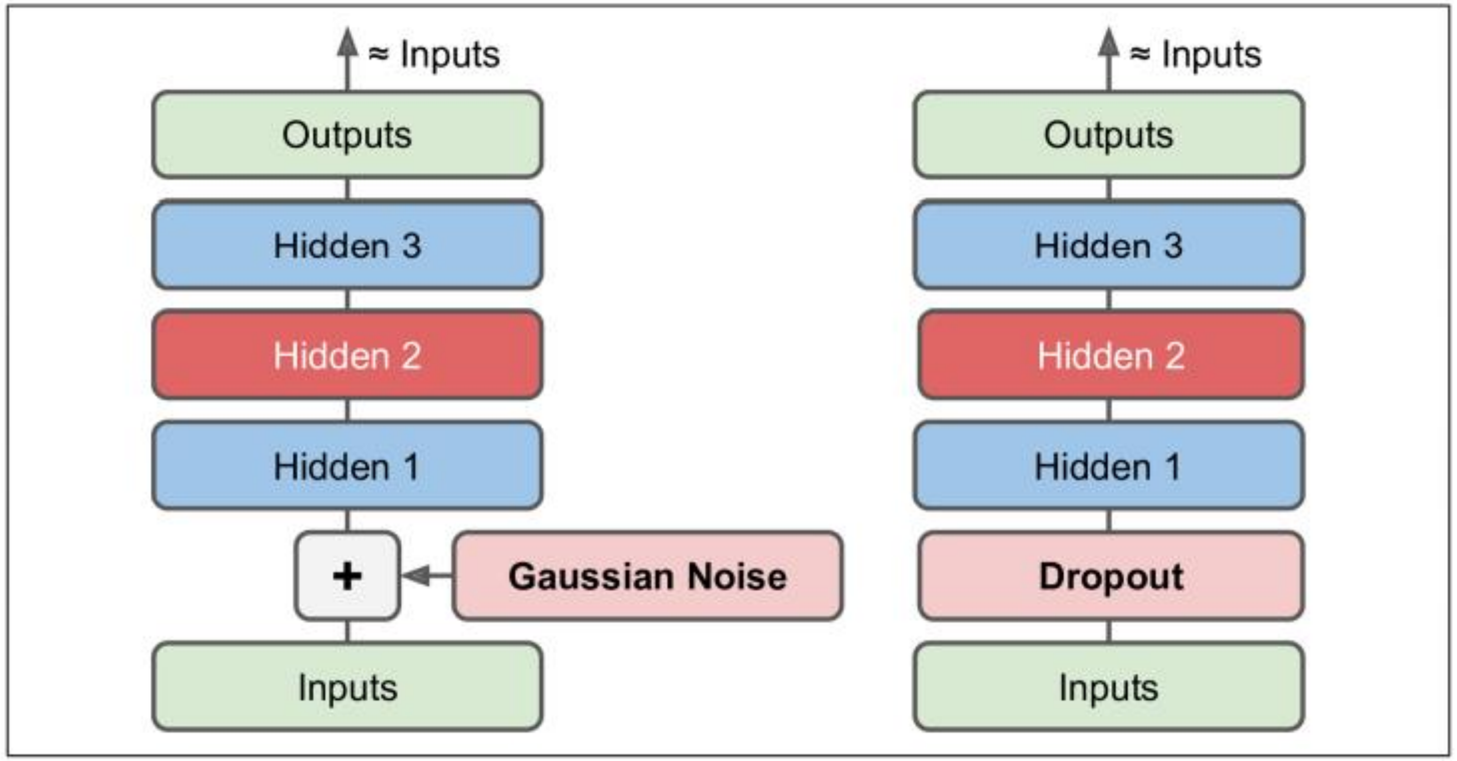


Figure 15-9. Denoising autoencoders, with Gaussian noise (left) or dropout (right)

Variational Autoencoders

- Another important category of autoencoders variational autoencoders.

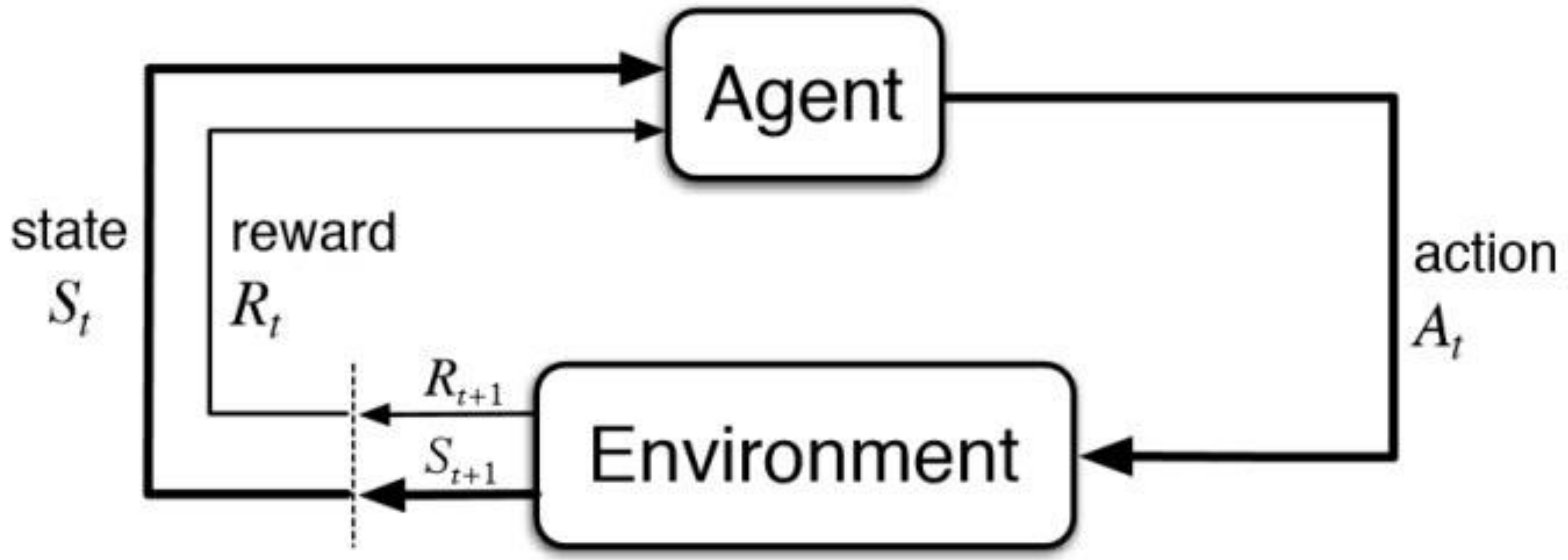
- They are probabilistic autoencoders, meaning that their outputs are partly determined by chance, even after training

- Most importantly, they are generative autoencoders, meaning that they can generate new instances that look like they were sampled from the training set.

Reinforcement Learning

- What is reinforcement Learning?
- Reinforcement learning is **a machine learning training method based on rewarding desired behaviors and/or punishing undesired ones**.
- In general, a reinforcement learning agent is able to perceive and interpret its environment, take actions and learn through trial and error.
- Reinforcement Learning (RL) is **the science of decision making**. It is about learning the optimal behavior in an environment to obtain maximum reward.

Reinforcement Learning Architecture



Some of the important terms used

- **Agent:** It is an assumed entity which performs actions in an environment to gain some reward.
- **Environment (e):** A scenario that an agent has to face.
- **Reward (R):** An immediate return given to an agent when he or she performs specific action or task. (can be award or punishment)
- **State (s):** State refers to the current situation returned by the environment.

Example:

- The agent can be the program controlling a walking robot.
- In this case, the environment is the real world, the agent observes the environment through a set of sensors such as cameras and touch sensors.
- Its actions consist of sending signals to activate motors.
- It may be programmed to get positive rewards whenever it approaches the target destination, and negative rewards whenever it wastes time, goes in the wrong direction, or falls down.

Policy Search

- The algorithm used by the software agent to determine its actions is called its policy.

- For example, the policy could be a neural network taking observations as inputs and outputting the action to take (see Figure 16-2).

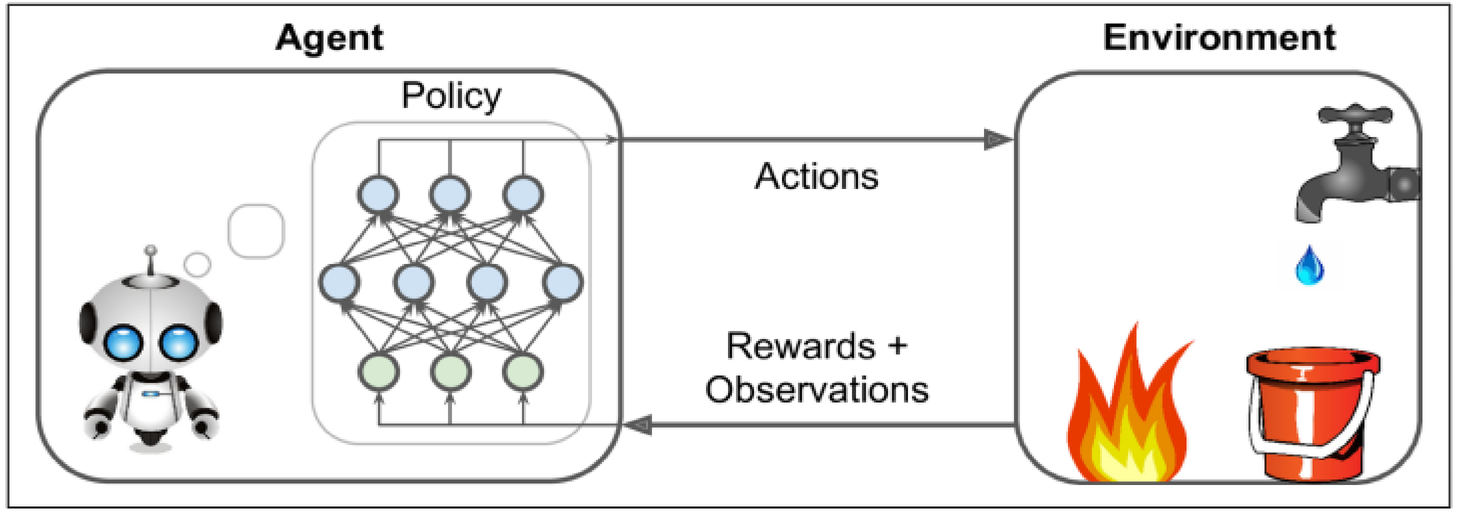


Figure 16-2. Reinforcement Learning using a neural network policy

Neural Network Policy

- Neural network will take an observation as input, and it will output the action to be executed.
- More precisely, it will estimate a probability for each action, and then we will select an action randomly according to the estimated probabilities.

Neural Network Policy

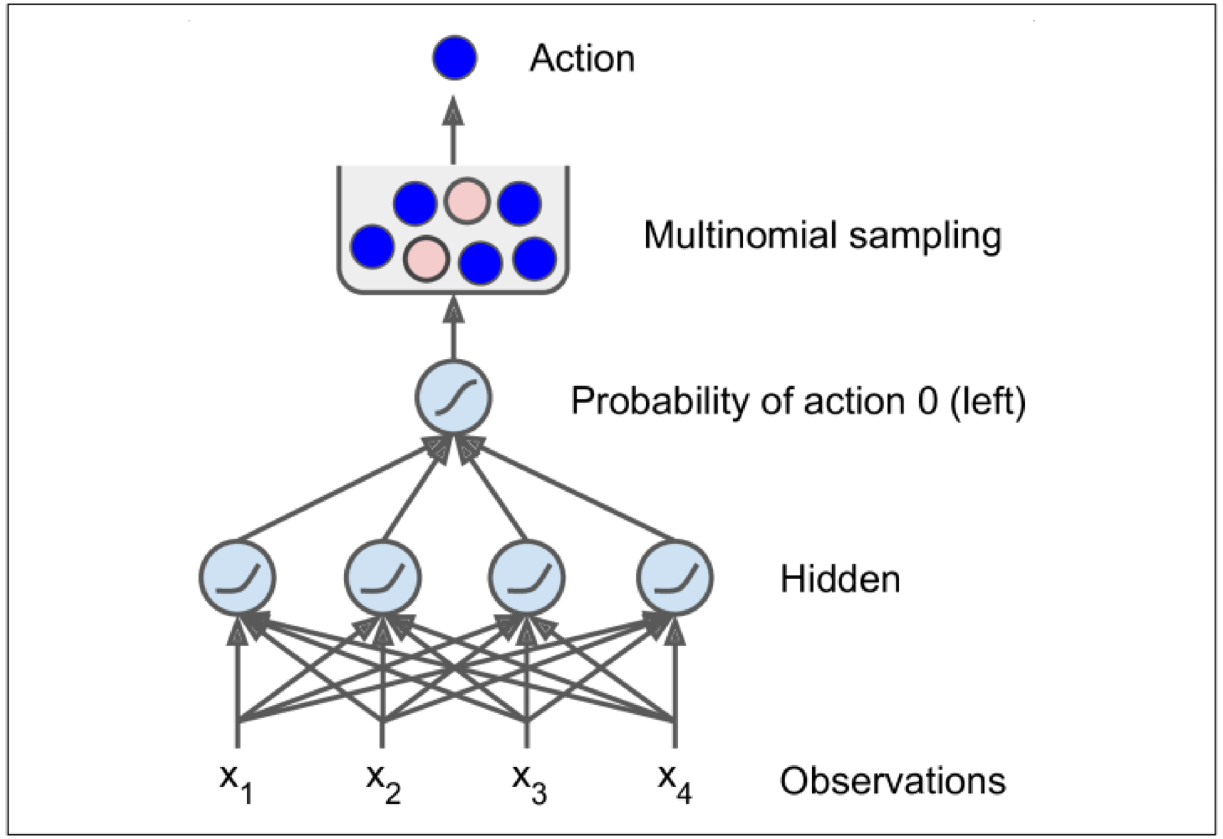


Figure 16-5. Neural network policy

Evaluating Actions:

- If we knew what the best action was at each step, we could train the neural network as usual, by minimizing the cross entropy between the estimated probability and the target probability.

- It would just be regular supervised learning.

- However, in Reinforcement Learning the only guidance the agent gets is through rewards, and rewards are typically sparse and delayed.

- To tackle this problem, a common strategy is to evaluate an action based on the sum of all the rewards that come after it, usually applying a discount rate r at each step.

- if an agent decides to go right three times in a row and gets +10 reward after the first step, 0 after the second step, and finally –50 after the third step, then assuming we use a discount rate r = 0.8, the first action will have a total score of 10 + r × 0 + r 2 × (–50) = –22.

Policy Gradients (PG)

- PG algorithms optimize the parameters of a policy by following the grad
- First, let the neural network policy play the game several times and at each step compute the gradients that would make the chosen action even more likely, but don’t apply these gradients yet.
- Once you have run several episodes, compute each action’s score.
- If an action’s score is positive, it means that the action was good and you want to apply the gradients computed earlier to make the action even more likely to be chosen in the future.
- However, if the score is negative, it means the action was bad and you want to apply the opposite gradients to make this action slightly less likely in the future. The solution is simply to multiply each gradient vector by the corresponding action’s score.
- Finally, compute the mean of all the resulting gradient vectors, and use it to per- form a Gradient Descent step

Markov Decision Process (MDP)

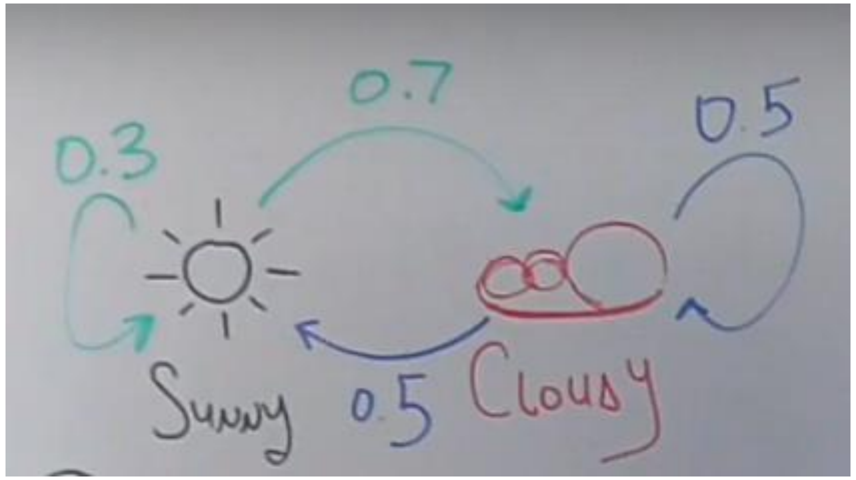
- If the information provided by the state is sufficient to determine the future states of the environment given any action, then the state is deemed to have the Markov property (or in some literature, the state is deemed as Markovian).

- This comes from the fact that the environments we deal with while doing Reinforcement Learning are modeled as Markov Decision Processes.

Markovian Decision Process continued..

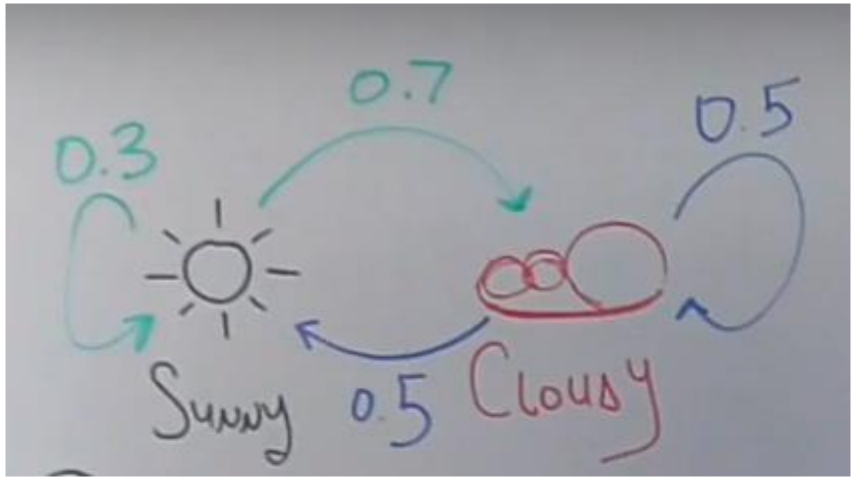
- Reinforcement Learning has:
  - Agent
  - Actions
  - Environment
  - Rewards

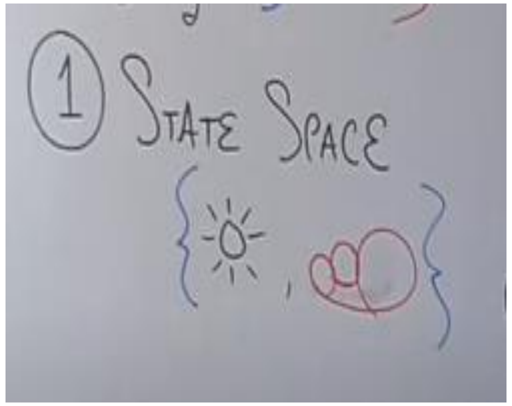
  - Consider the following to explain MDP

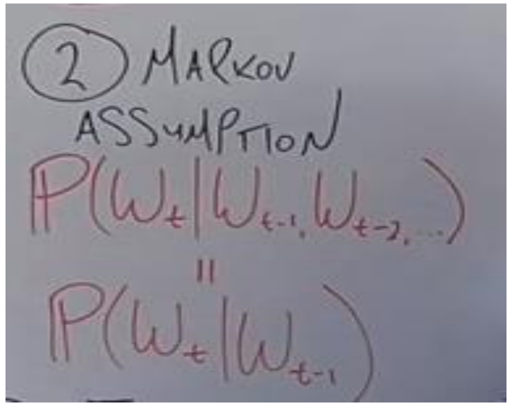


MDP

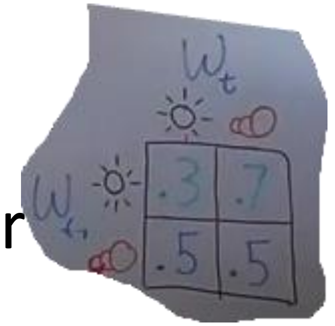
- Consider the state of the climate can be Sunny or Cloudy.







- The state change is shown in the diagram above
- The state space has only two states ..Ref (1)
- Markov assumption finally says that the current state depends only on the previous state, not on any other state earlier to it.
- In Reinforcement Learning, state change decision
- Depends only on the previous state not on any other earlier



TDL

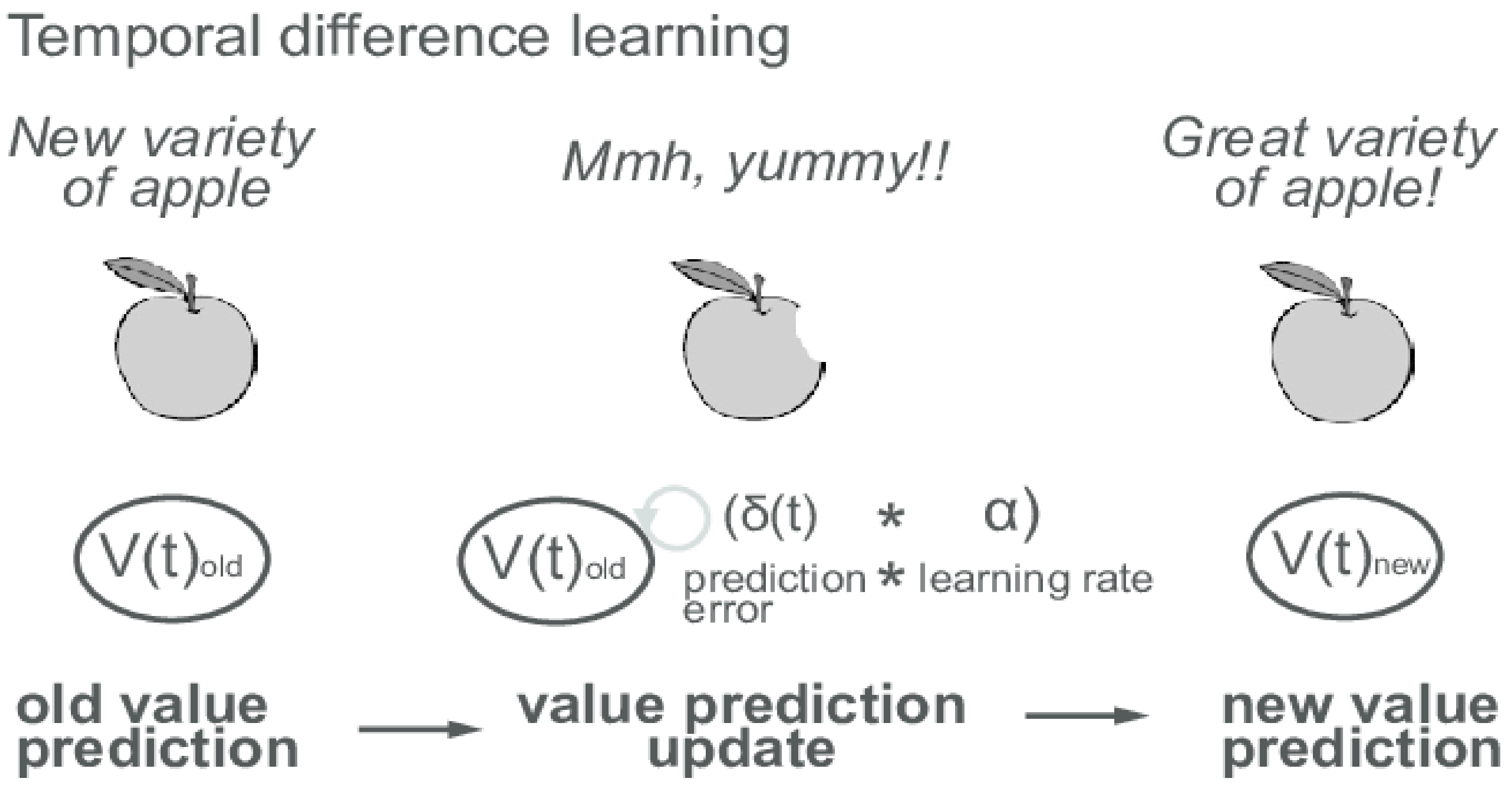
- TDL – Temporal Difference Learning

- Temporal difference (TD) learning is **an approach to learning how to predict a quantity that depends on future values of a given signal**. The name TD derives from its use of changes, or differences, in predictions over successive time steps to drive the learning process.

- Temporal Difference Learning is an unsupervised learning technique that is very commonly used in reinforcement learning for the purpose of predicting the total reward expected over the future.

TDL example

- TDL estimates the quantity, which depends on the feature signals



Q-Learning (Quality Learning)

- Q-learning is a variant of reinforcement learning algorithm that seeks to find the best action to take given the current state.
- It uses Monte Carlo Policy.
- It observes reward for all the steps in the episode, where as TDL observes only one step.

End of Unit - 5

End of the Course
Deep Learning Architecture

Wishing You
All the best!

Dr.Srinath.S
Associate Professor and Head
Dept. of CS&E
SJCE, JSS S&TU
Mysuru- 570006