UCL ARTIFICIAL
INTELLIGENCE SOCIETY

TUTORIAL #7

Session 7

Deep Neural Networks

UCL ARTIFICIAL
INTELLIGENCE SOCIETY

LECTURE OVERVIEW

01

NN Training

How do we train NN?
What are some common mistakes?

02

RNN

Showcase of RNN architecture.

03

LSTM

Showcase of LSTM architecture.

04

Interpretability

How can we better understand what is going on inside of NN?

Theory

NN TRAINING

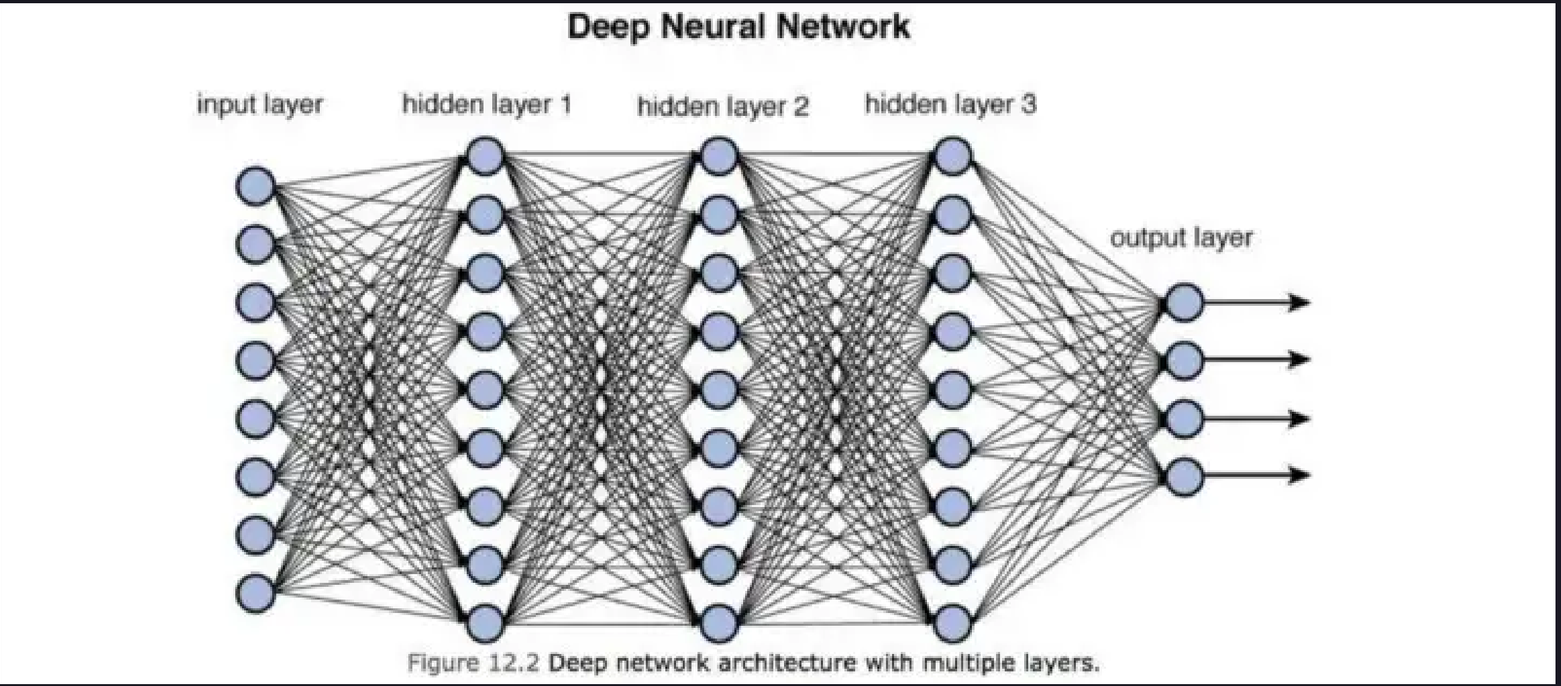


Figure 12.2 Deep network architecture with multiple layers.

Theory

NN TRAINING

What might be some good things to do before training Deep Neural Network?

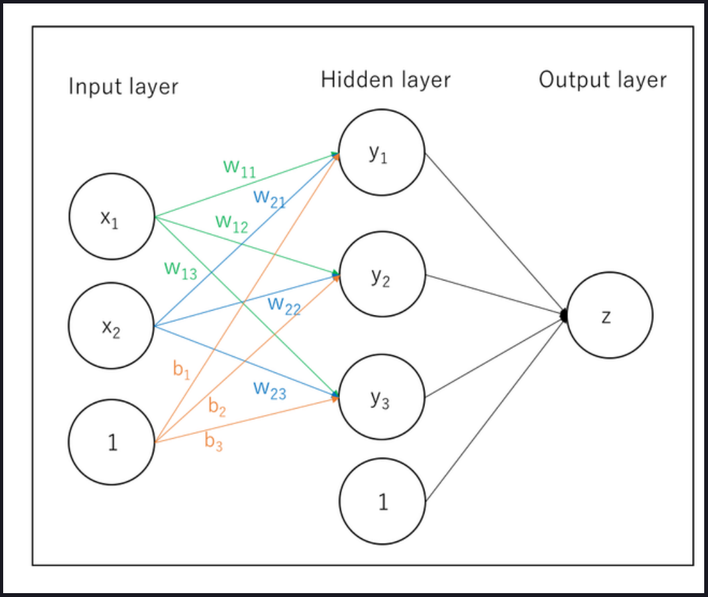
- Deep NN is a structure mainly for supervised learning (at least in this tutorial)
- We are thus mapping some input to some output

Theory

NN TRAINING

How do we initialize weights?

- Should all weights start as 0?



$$y_1 = x_1 \times w_{11} + x_2 \times w_{21} + b_1$$
$$y_2 = x_1 \times w_{12} + x_2 \times w_{22} + b_2$$
$$y_3 = x_1 \times w_{13} + x_2 \times w_{23} + b_3$$

Theory

NN TRAINING

How do we initialize weights?

NO! -> There is a danger of failure to break symmetry

In practice we use Xavier initialization.

The motivation behind Xavier initialization is to initialize the weights in such a way they do not end up in saturated regimes of tanh activation i.e initialize with values not too small or too large. To achieve that we scale by the number of inputs while randomly sampling from standard Gaussian.

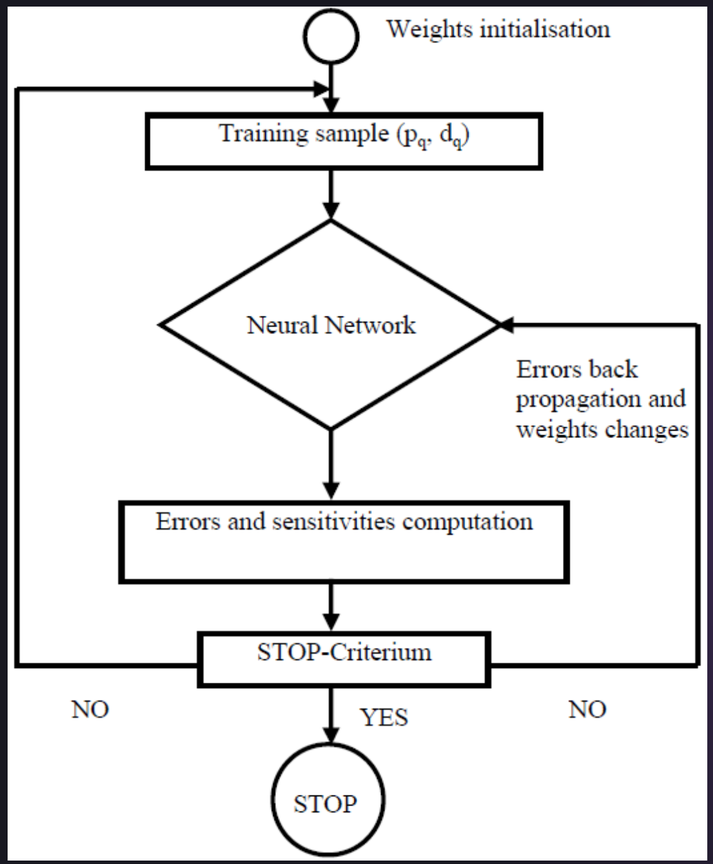
Theory

NN TRAINING

How does NN training look like?

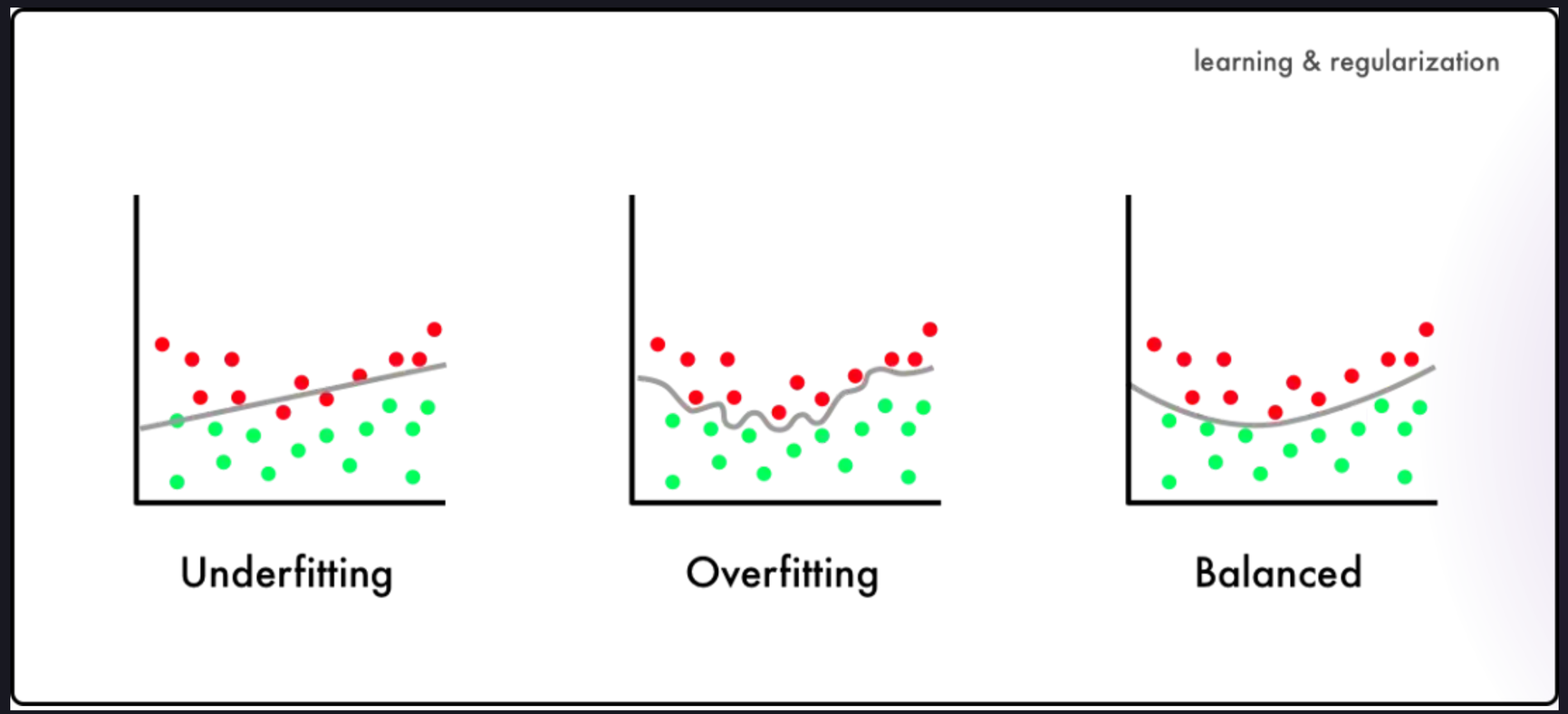
It is happening in **epochs**
- In each epoch the models is shown the whole training dataset

The **learning rate** controls how quickly the model is adapted to the problem.



Theory

NN TRAINING



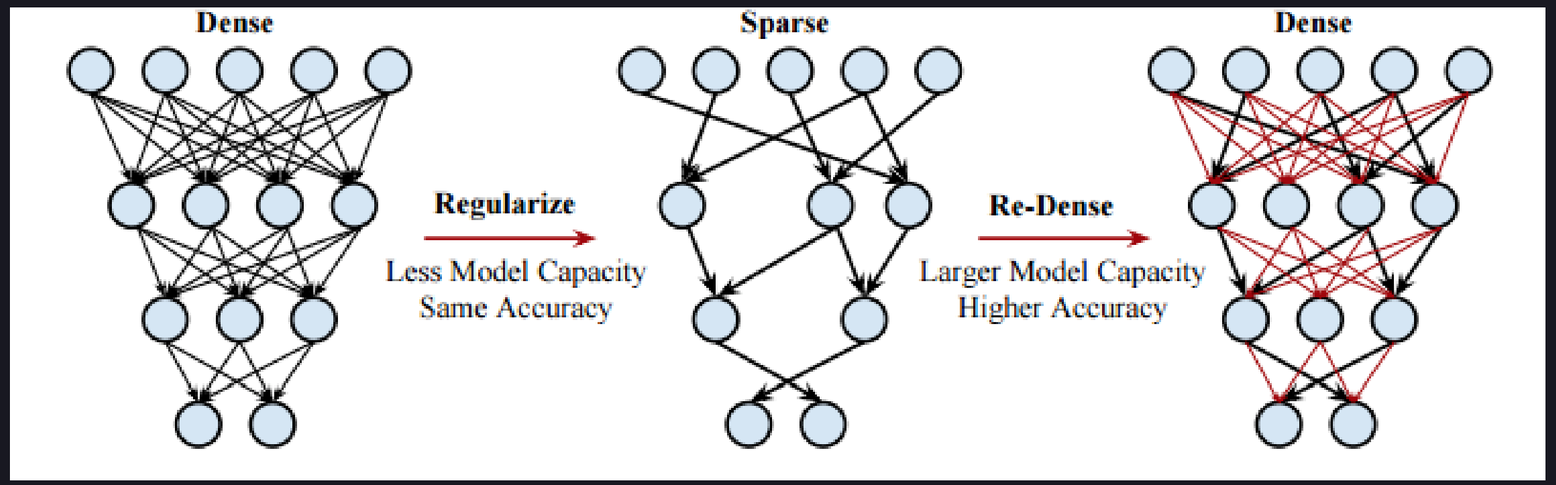
Theory

NN TRAINING

Regularization

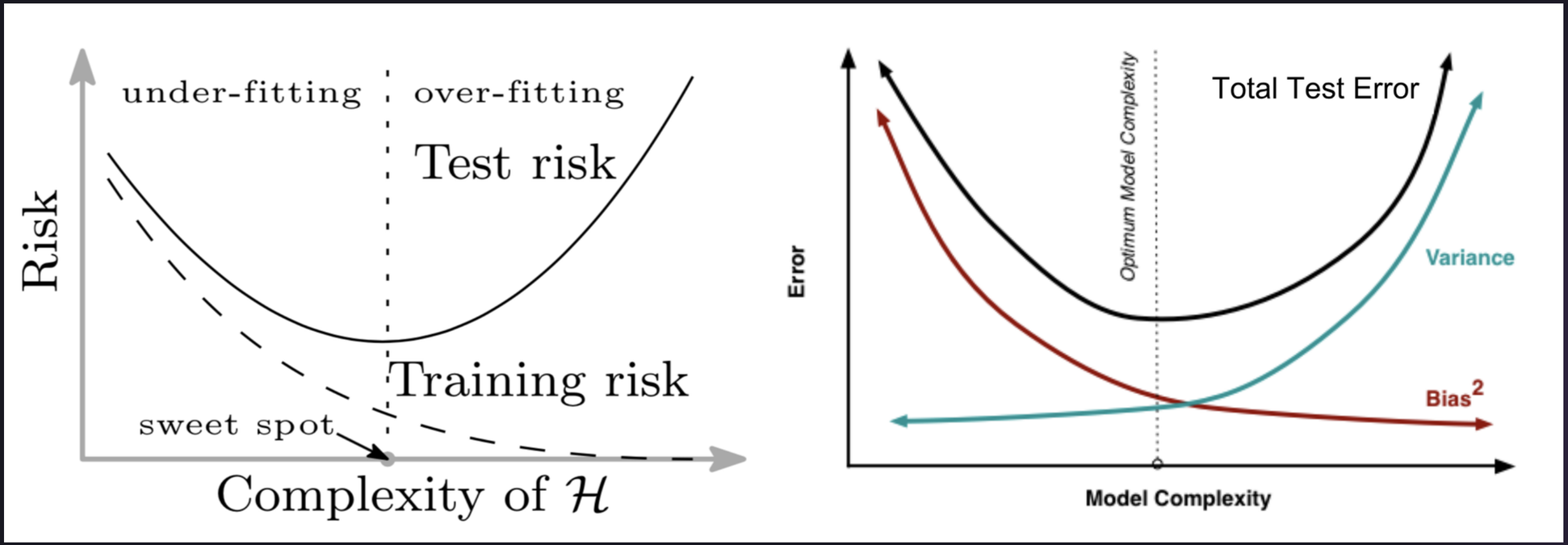
Technique designed to prevent over-fitting.

At each iteration we are randomly dropping some units from the network. And consequently, we are forcing each unit to not rely (not give high weights) on any specific set of units from previous layer as any of them could go off randomly.



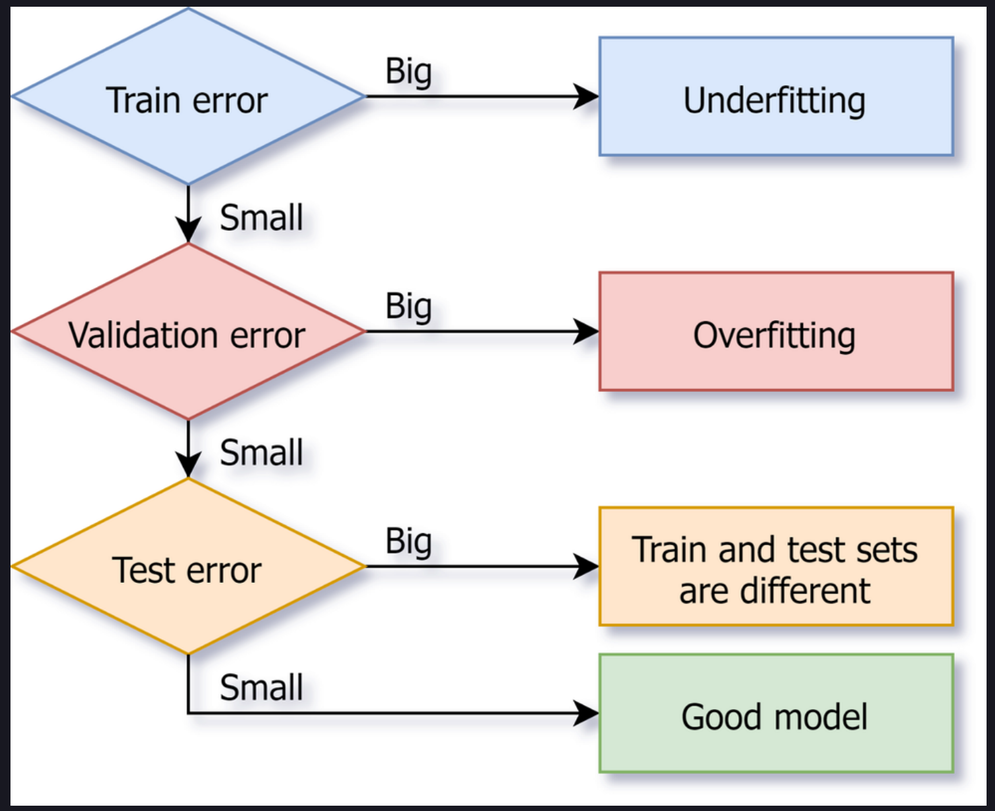
Theory

NN TRAINING



Theory

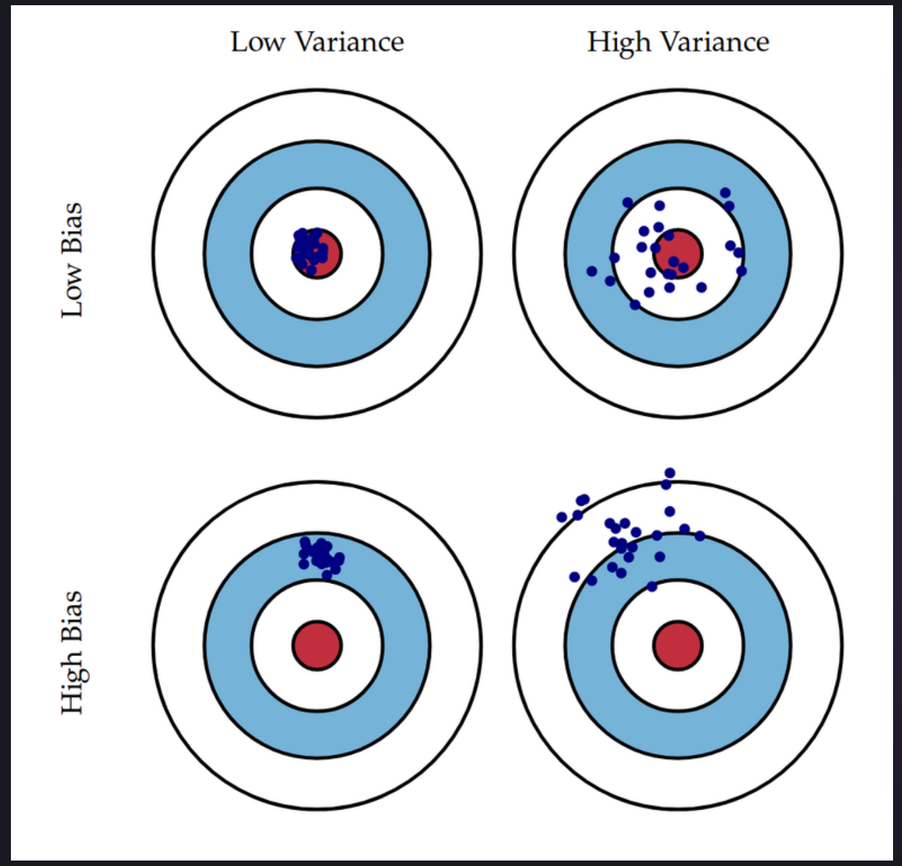
NN TRAINING



Theory

NN TRAINING

Variance and Bias



Theory

NN TRAINING

What are some signs and solutions to under-fitting

Reasons for uner-fitting:

- High bias and low variance
- The size of the training dataset used is not enough
- The model is too simple
- Training data is not cleaned and also contains noise in it

Techniques to reduce under-fitting:

- Increase model complexity
- Increase the number of features, performing feature engineering
- Remove noise from the data.
- Increase the number of epochs or increase the duration of training to get better results.

Theory

NN TRAINING

What are some signs and solutions to over-fitting

Reasons for oer-fitting:

- High variance and low bias
- The model is too complex
- The data is too simplistic for the problem

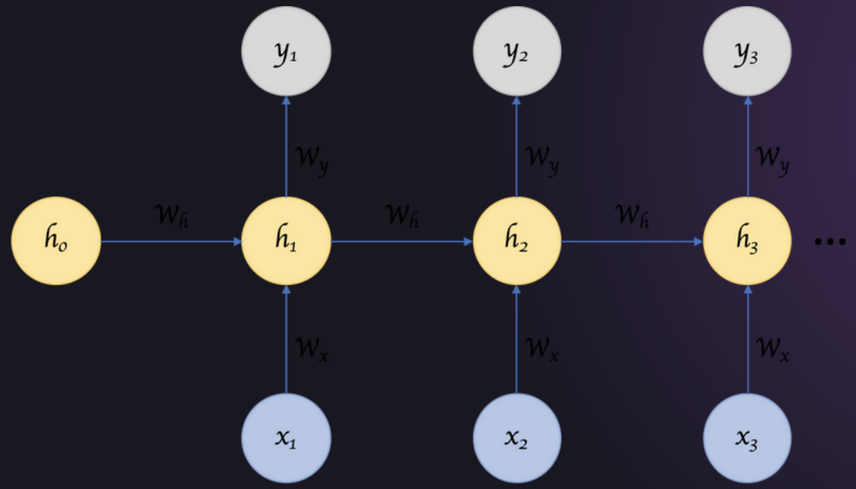
Techniques to reduce over-fitting:

- Increase training data
- Reduce model complexity
- Use special techniques (Early stopping, Regularization, Dropout)

Theory

RNN

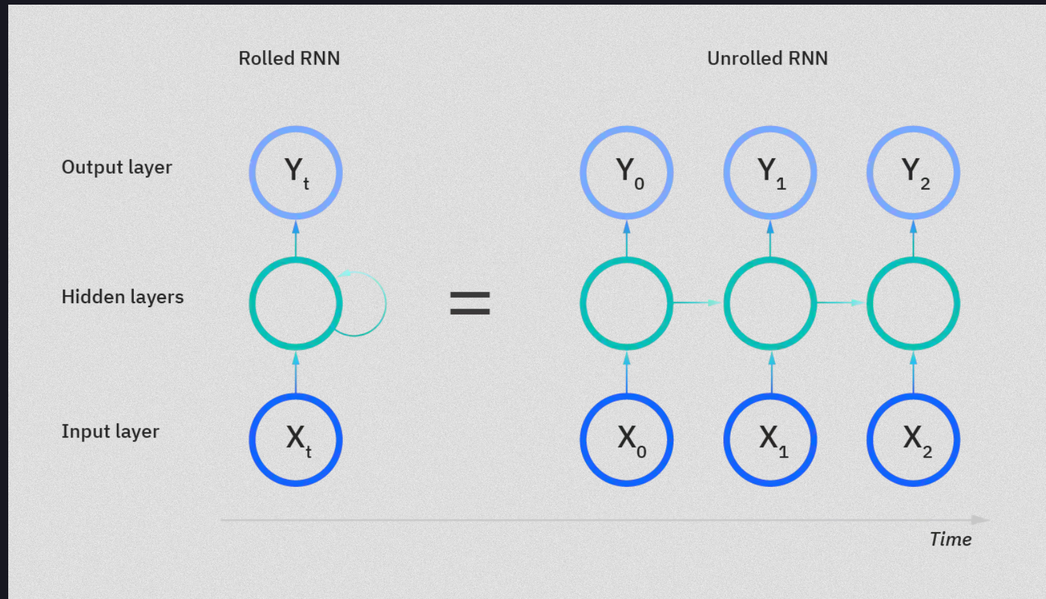
- Recurrent Neural Networks process sequential data
- Examples include: text, speech, stock data
- RNNs can even be used for machine vision



Theory

RNN

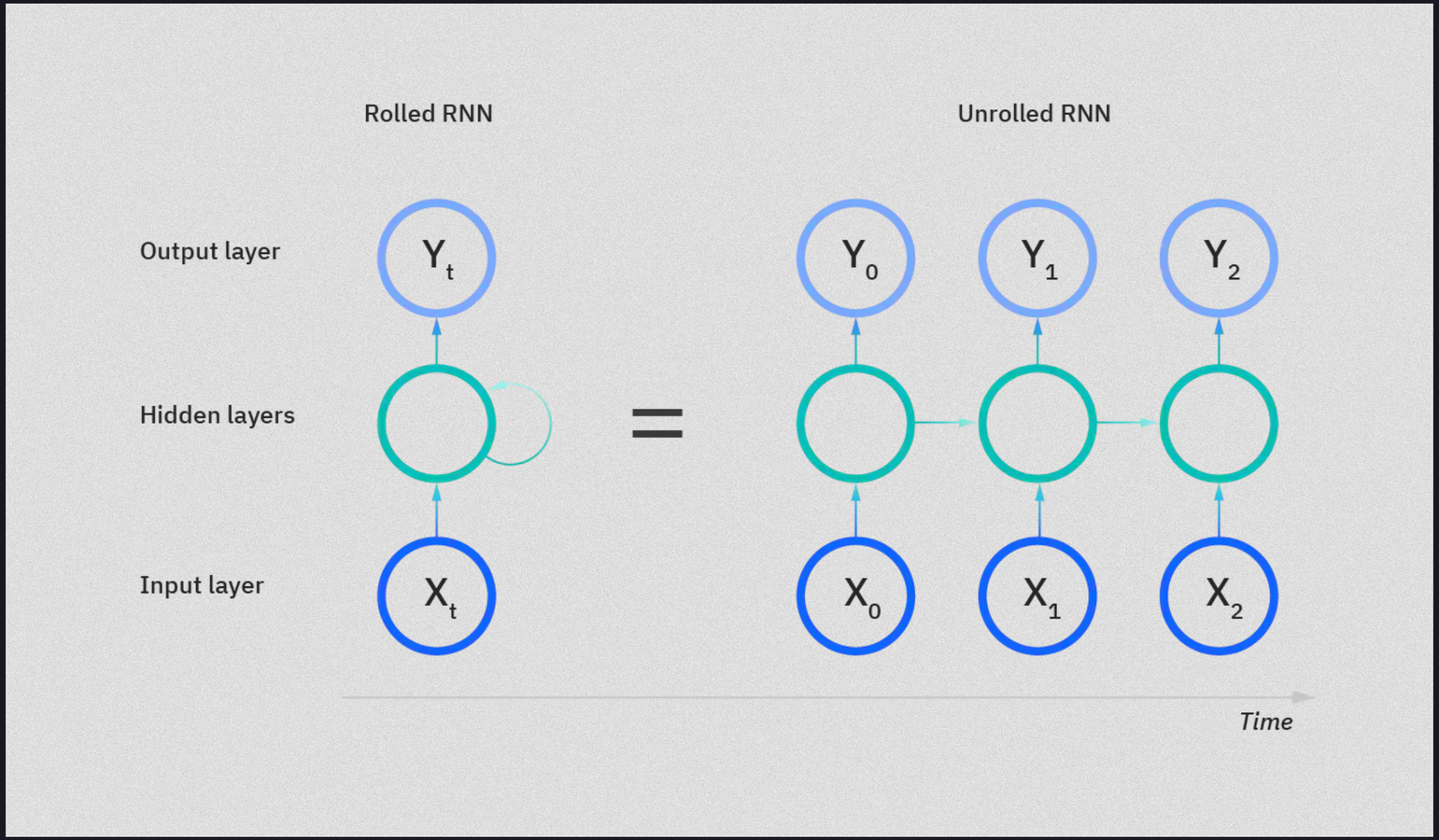
- Recurrent Neural Networks use the ideal of a regular feed forward neural net
- However, it also has "sequential memory"
- This means, for every time step/step in sequence, information from the previous step is passed on.



Theory

RNN

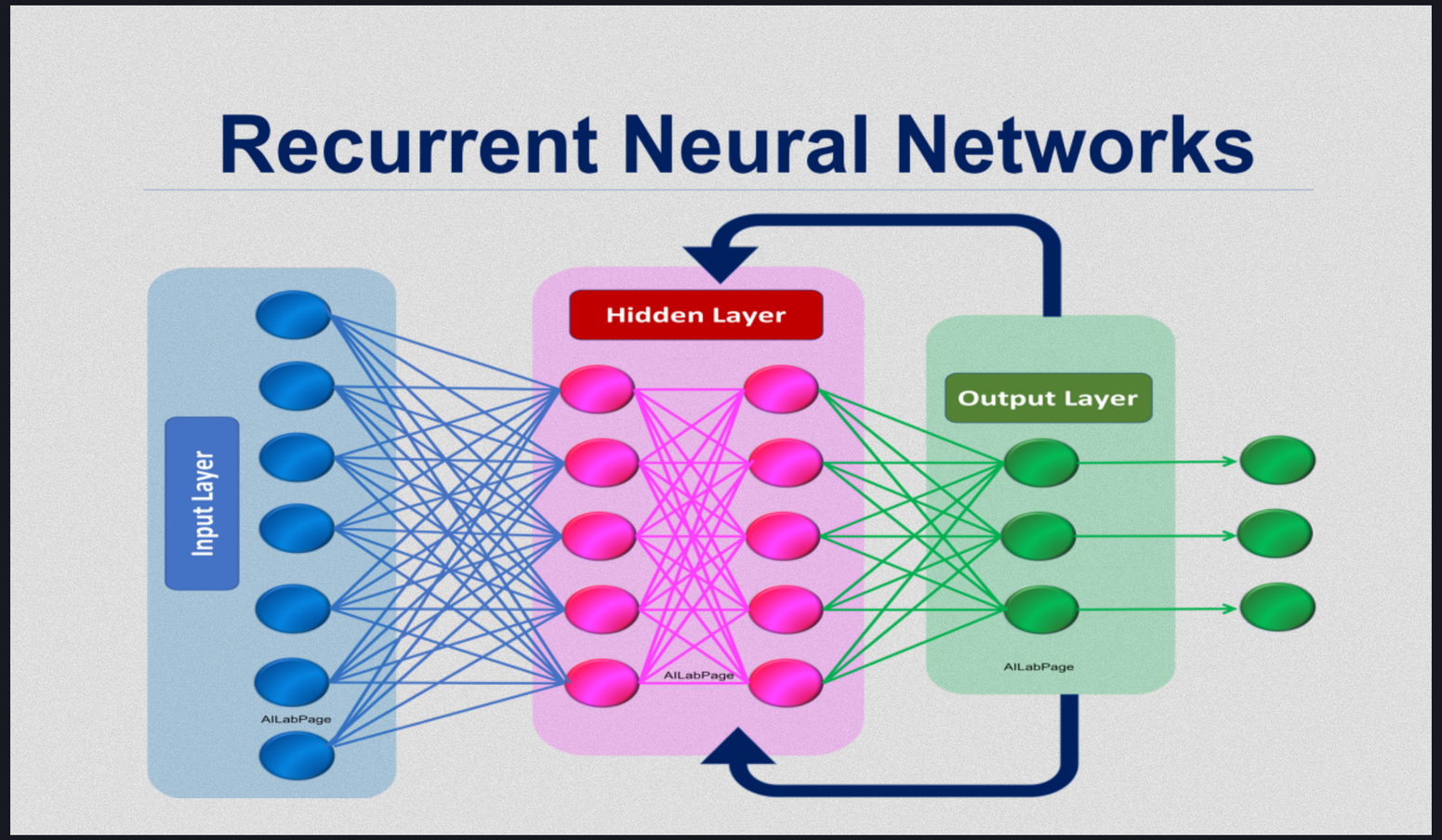
- Look at image on the right.
- Each stack of three nodes represents a feed-forward neural network, with input, hidden layers and output layer.
- A RNN can be thought of as being composed of a chain of these neural networks, where some information from each network is passed to the next in the chain
- In reality, it is a single neural network with loop properties within it's architecture



Theory

RNN

- A RNN as one single feed forward neural network, with the added feature that some information from the output layer is fed back to the previous layers each pass.



Theory

RNN

- Lets dive deeper into RNNs.
- Elman and Jordan networks are simple examples of an RNN architecture.
- We can see the result of a given time step t in the RNN.
- Lets review the Elman network:

1. Learns a weight matrices for inputs x (identical to ffn)
2. Additionally learns a weight matrix for the output of previous time step
3. Adds bias term
4. Passes whole thing into an activation function
5. The result is then passed to next time step (next network to be processed)

Elman network[25]

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

Jordan network[26]

$$h_t = \sigma_h(W_h x_t + U_h y_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

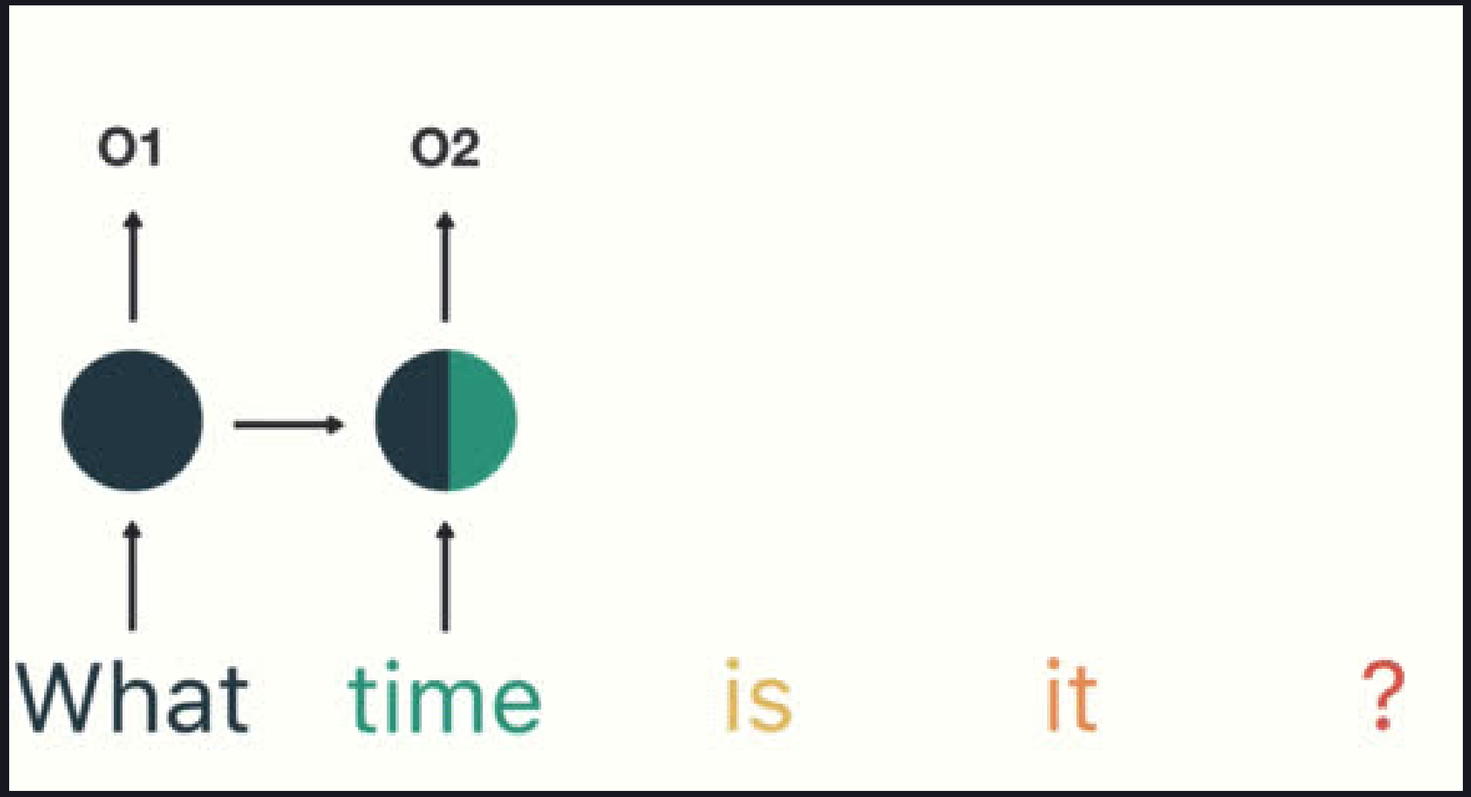
Variables and functions

- x_t: input vector
- h_t: hidden layer vector
- y_t: output vector
- W, U and b: parameter matrices and vector
- σ_h and σ_y: Activation functions

Theory

RNN

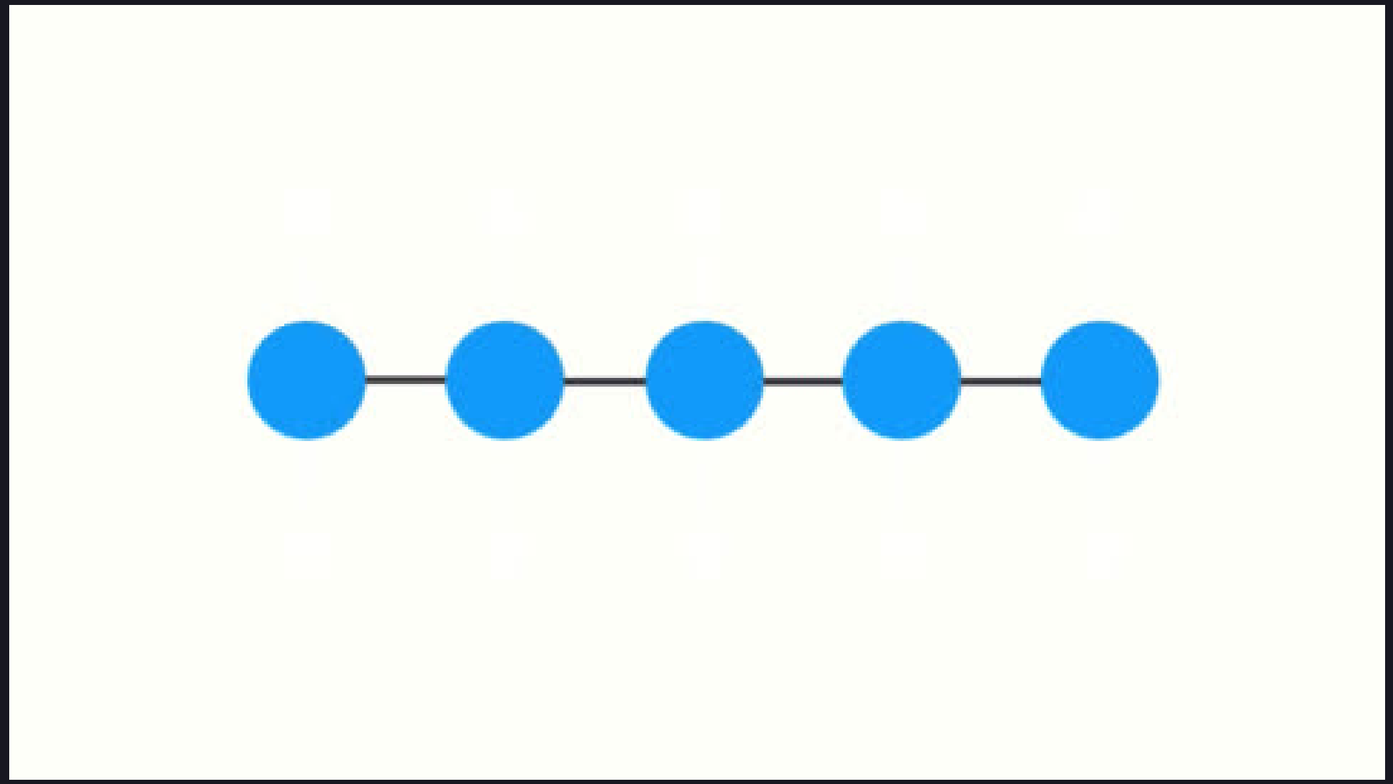
- Issues with RNNs
- Imagine a RNN used for a NLP task.
- Someone asks the model for the time (shown on the right)
- Each pass of the RNN focuses on one word in the sentence.
- The colors illustrate the importance/weight of previous time steps.
- As you can see, the information encoded for "what" and "time" are miniscule in the final step.
- This can cause problems when processing long sequences.



Theory

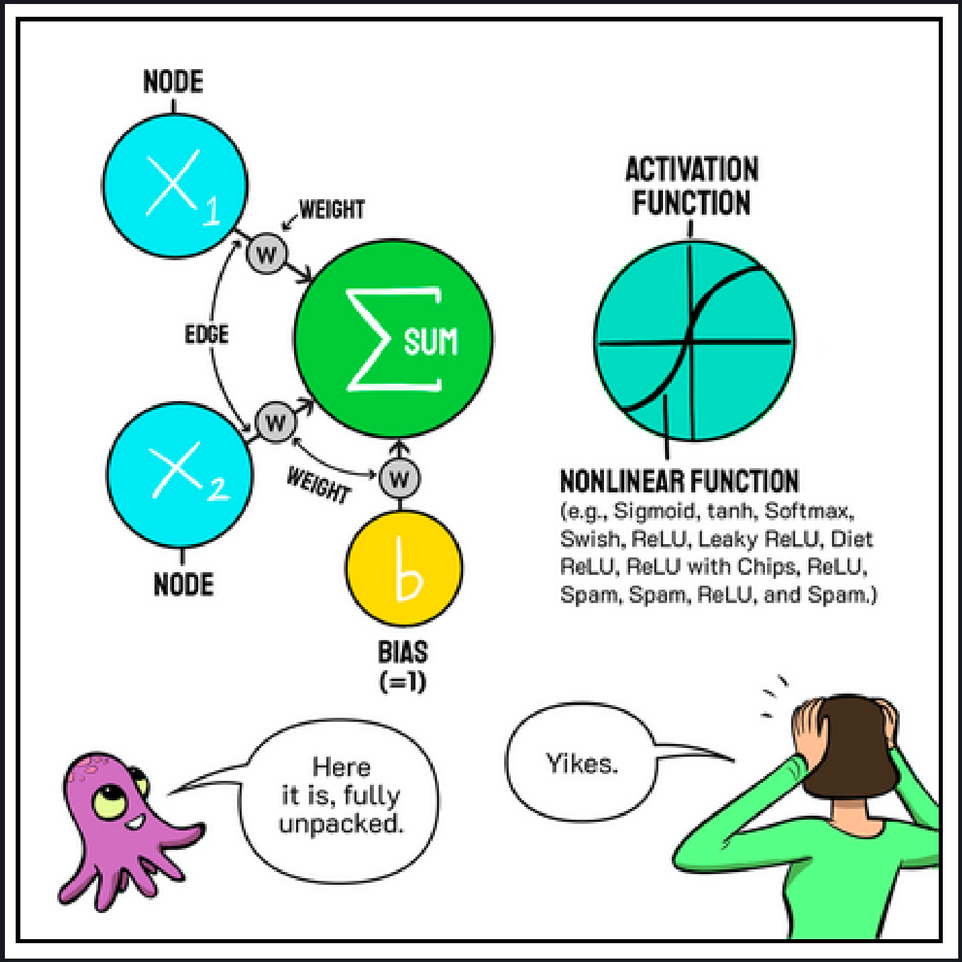
RNN

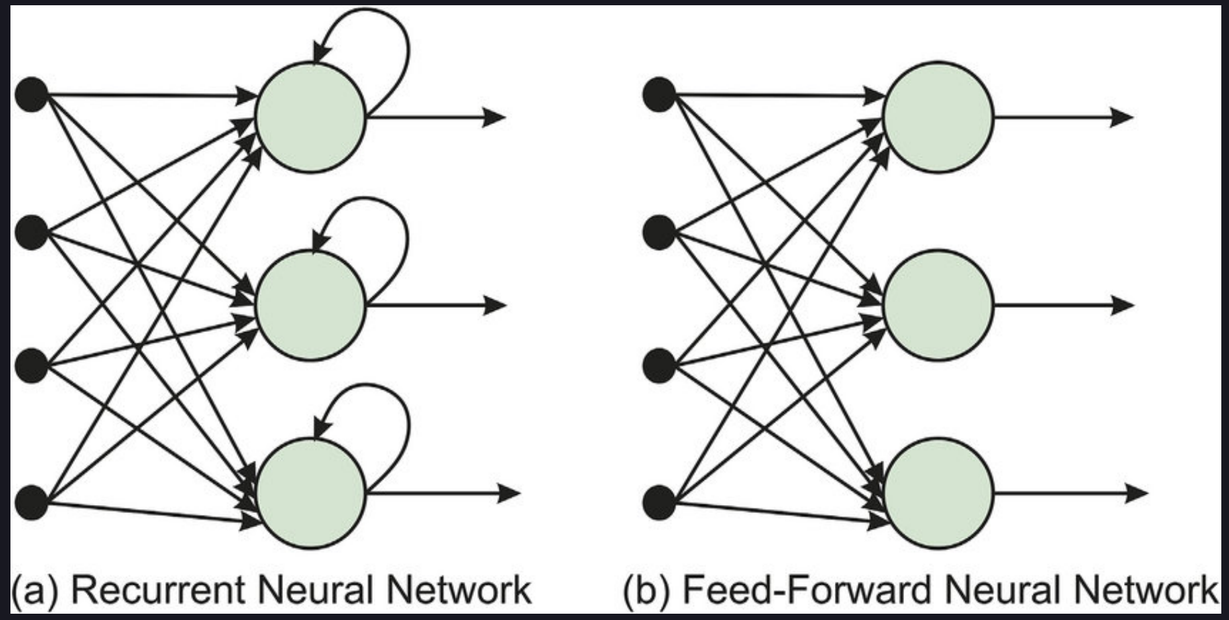
- Issues with RNNs
- Vanishing gradient problem
- Due to the nature of the backpropagation algorithm, the earlier the weights in the network, the smaller the weight updates.
- This means that the earlier weights stop "learning"
- Therefore, RNNs cannot learn long range dependencies in sequences



Theory

LSTM (LONG SHORT-TERM MEMORY)

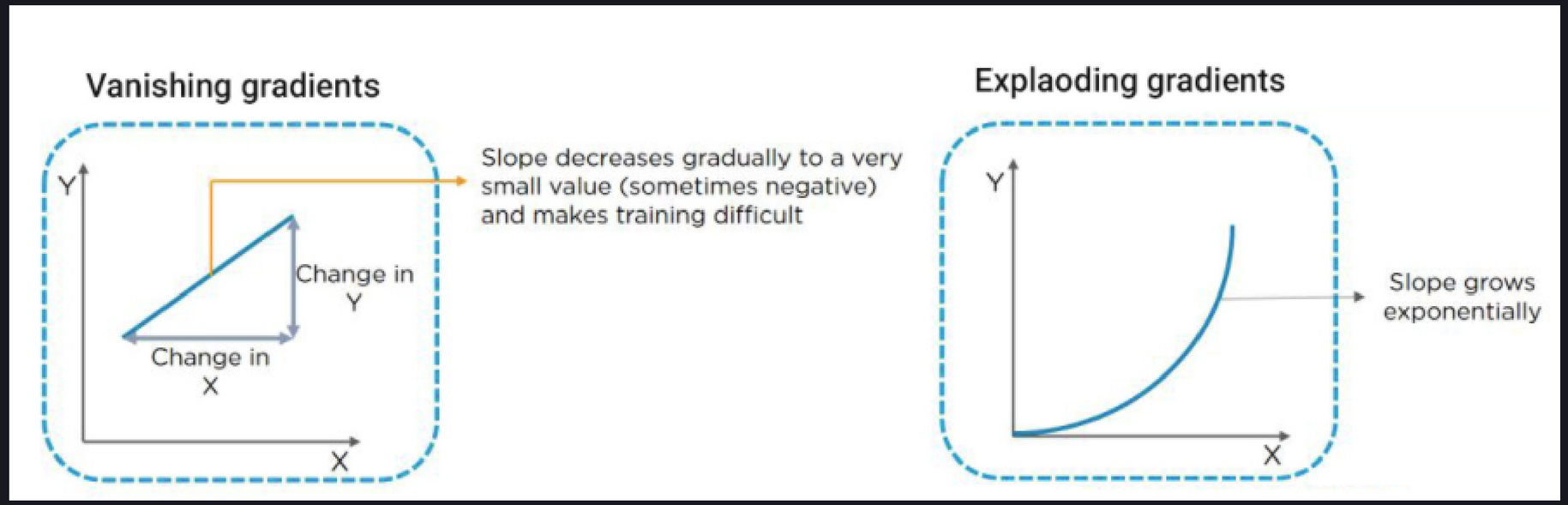




Theory

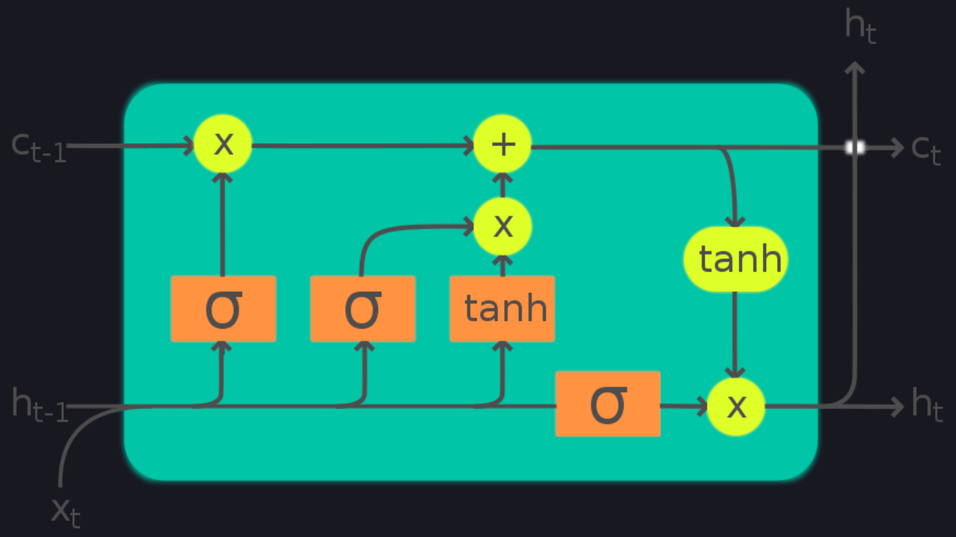
LSTM

RECAP OF VANISHING GRADIENT



Theory

LSTM



Legend:

Layer

ComponentwiseCopy

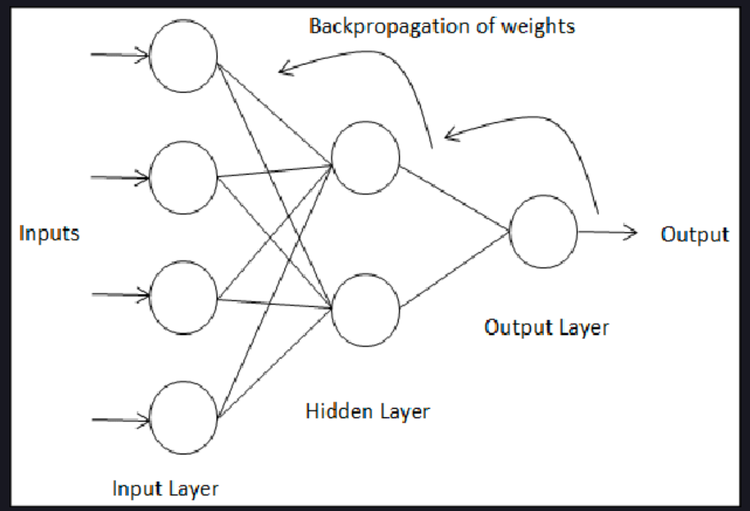
Concatenate

Theory

LSTM

LSTMs help preserve the error that can be backpropagated through time and layers.

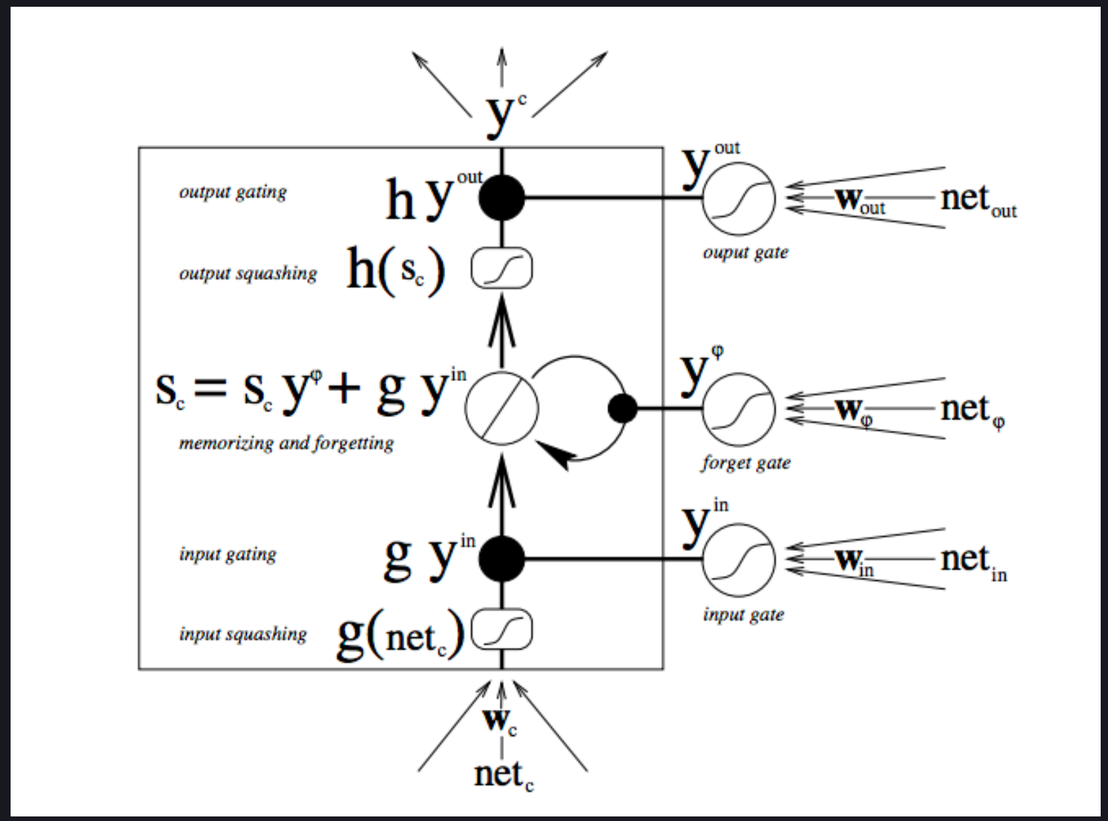
- They do this by maintaining a constant error.
- **This is one of the central challenges to machine learning and AI, since algorithms are commonly faced with sparse reward signals.**



Theory

LSTM

The diagram below illustrates how data flows through a memory cell and is controlled by its gates.



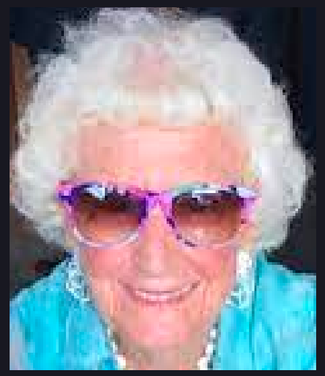
Theory

INTERPRETABILITY

Recap of NNs

1. Kernels/ Detectors

2. Grandma Kernels



kernel

Image

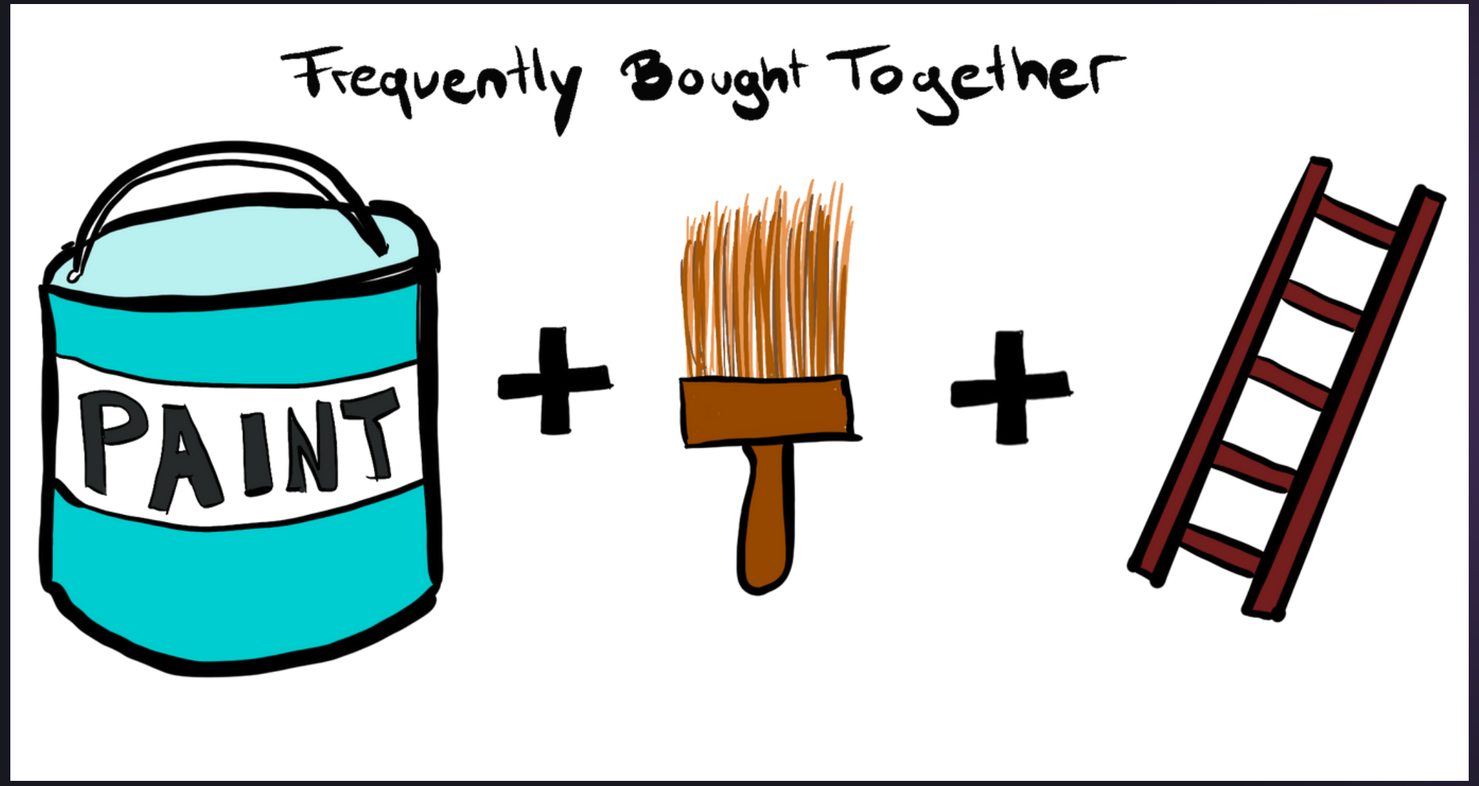
Theory

INTERPRETABILITY

Understanding why the model made the decision that it made.

1. Predict the model's output

2. Understand the cause of a decision



Theory

INTERPRETABILITY

But why ?

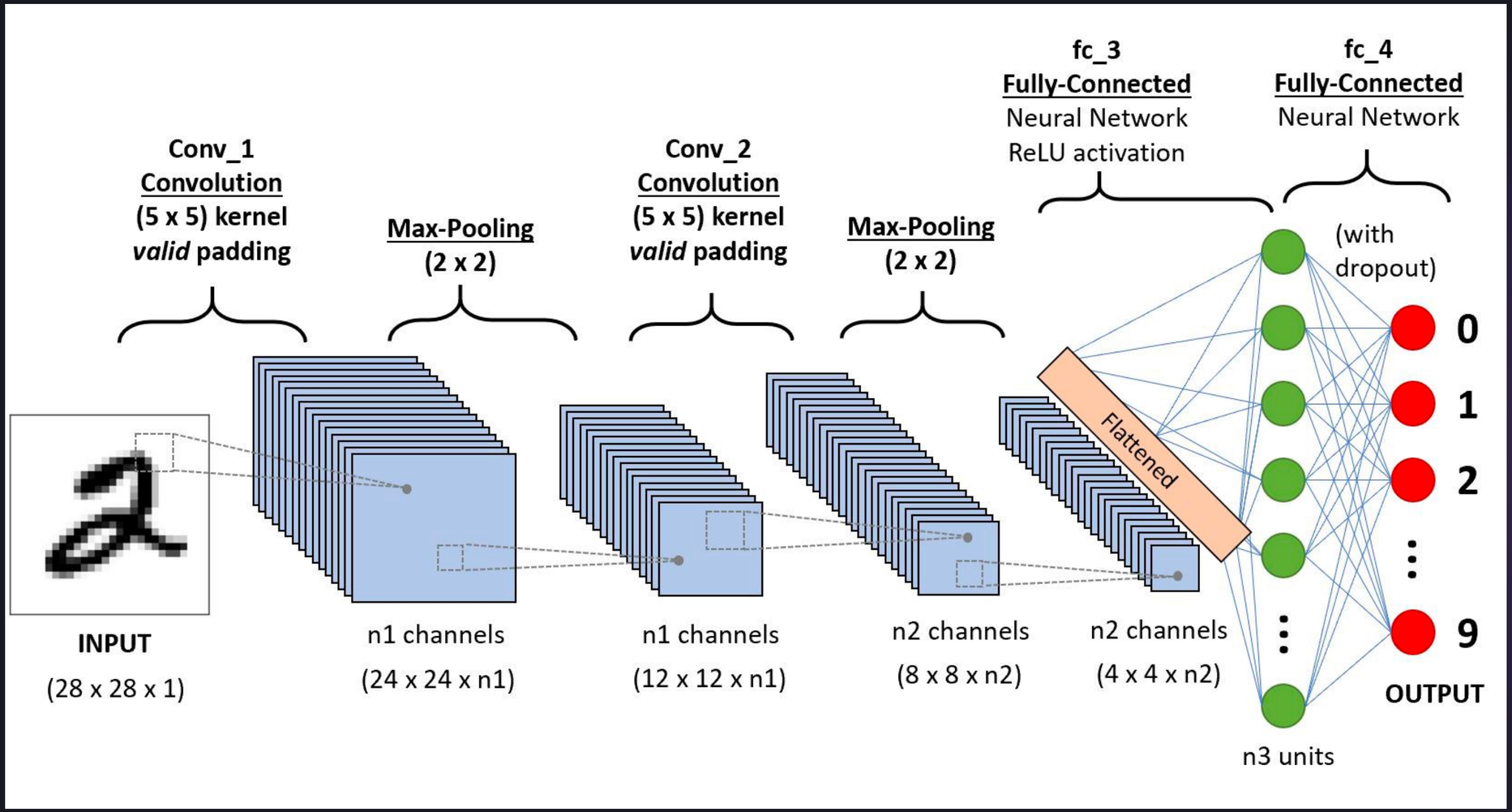
1. Make models more reliable/trustworthy
2. Detect failure cases.
3. Identify unwanted bias
4. Debug

Theory

INTERPRETABILITY

Dimensions = (input_size(after padding) - kernel_size + 1)/(stride + 1)

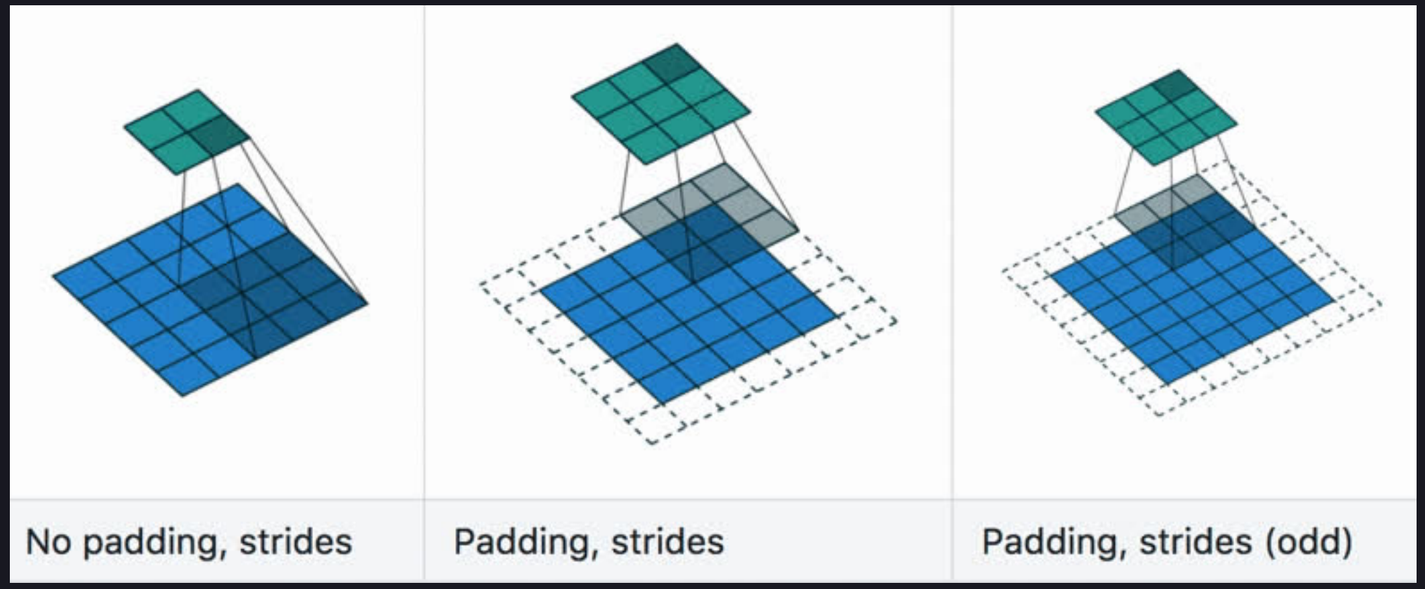
Recap



Theory

INTERPRETABILITY

Padding - adding additional pixels around the image so that the border pixels have equal weightage as the rest
Strides - the step size while performing convolutions



QUESTIONS?

SEE YOU NEXT TIME