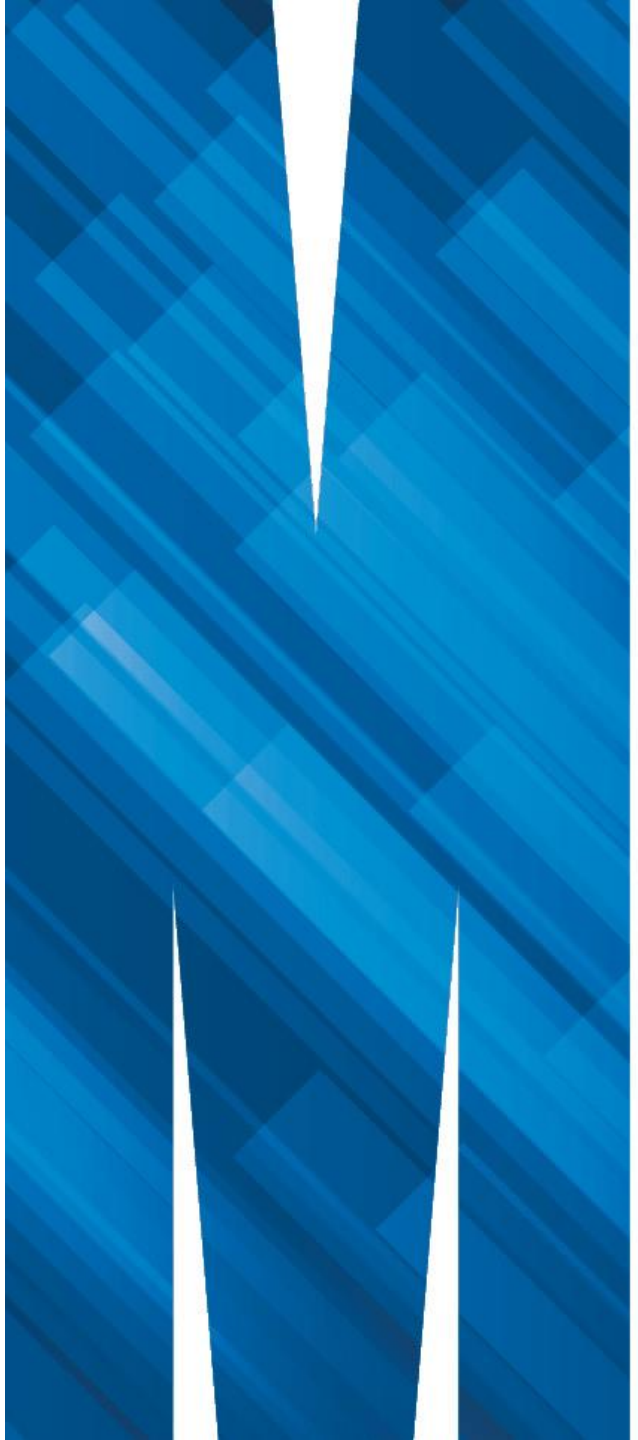
MONASH University

Introduction to Deep Learning and TensorFlow



Developed by Titus Tang 2020

Workshop Itinerary

Time	Event
09:00 – 09:15	Arrival and setup
09:15 – 10:15	Lecture 1 – Introduction to Deep Learning
10:15 – 10:30	Break
10:30 – 11:30	Workshop 1 - Introduction to Deep Learning
11:30 – 11:45	Break
11:45 – 12:45	Lecture 2 – Convolutional Neural Networks
12:45 – 13:00	Break
13:00 – 14:00	Workshop 2 – Convolutional Neural Networks
14:00 – 14:30	Question session

Workshop Content – Part 1

- Learn what is deep learning and how it works.
- Learn how to use TensorFlow v2 to build a simple neural network.
- Train and test our neural network.
- Extend and explore the capabilities of the neural network.

Workshop Content – Part 2

- Learn about convolutional neural networks and how they work.
- Build a conv-net using TensorFlow.
- Train and test our conv-net using a dataset of images.
- Explore the behaviour and capabilities of our conv-net.

Assumed Knowledge

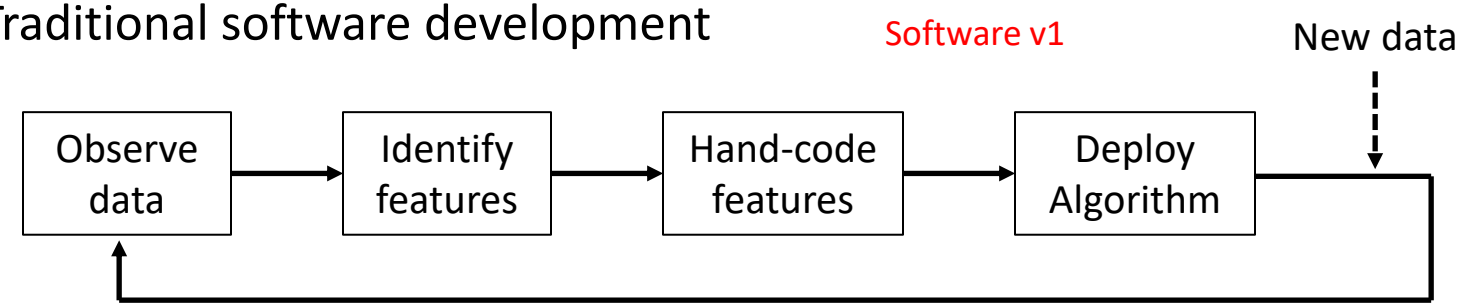
- Basic math and calculus.
- Basic Python programming skills.

vs

How would you traditionally design code that categorises images of dogs and cats?

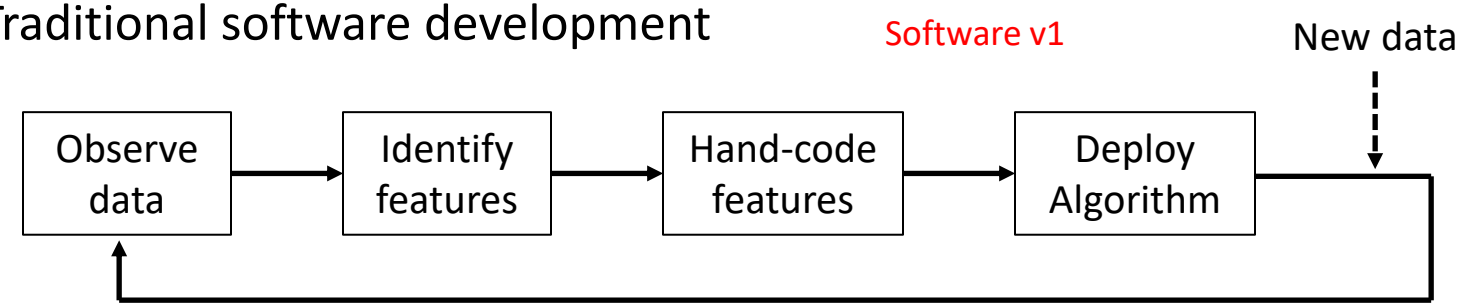
vs

Traditional software development

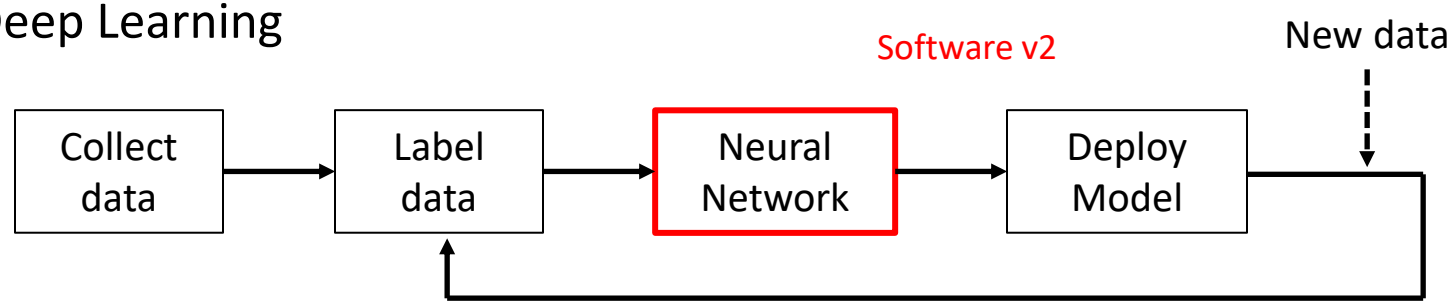


vs

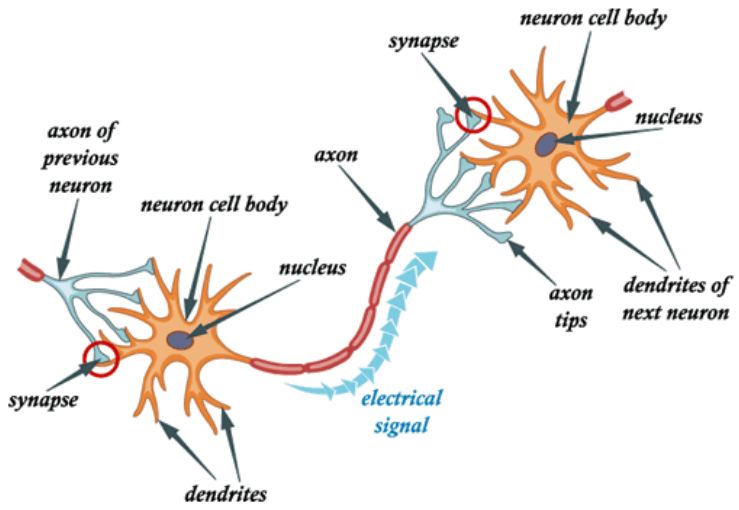
Traditional software development



Deep Learning

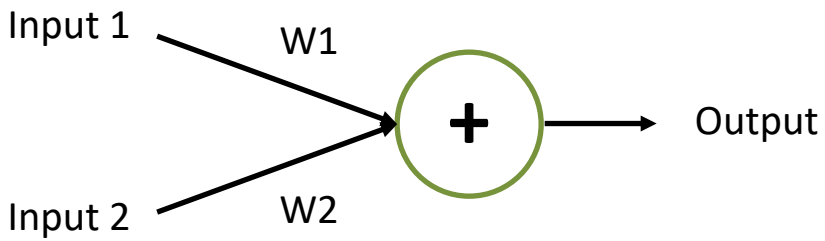


The Artificial Neural Network



- ✓ Modelled after the brain.
  - ❖ Each neuron takes input from multiple sources.
  - ❖ Neuron aggregates information and passes output to next layer.
- ✓ Fundamental unit – an artificial neuron, a perceptron.

Modelling a perceptron

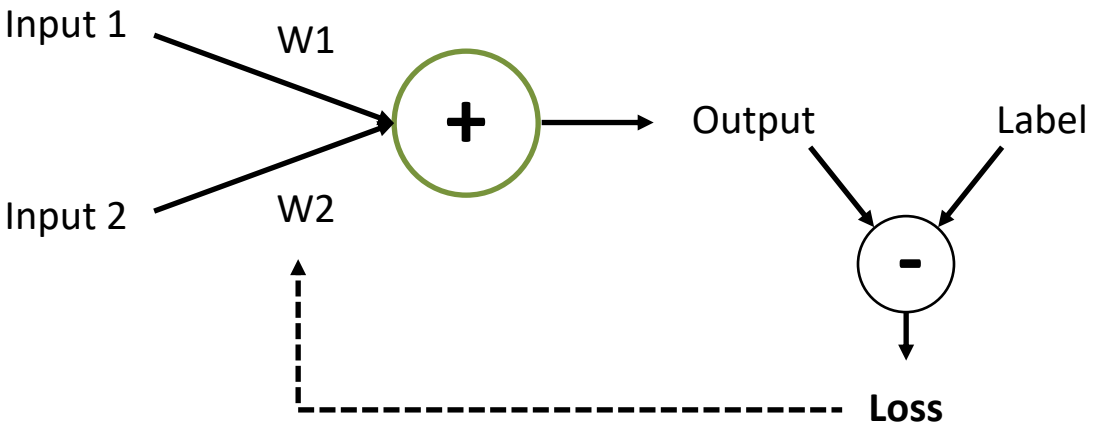


$$Output = (W1 * Input\ 1) + (W2 * Input\ 2)$$

In matrix form

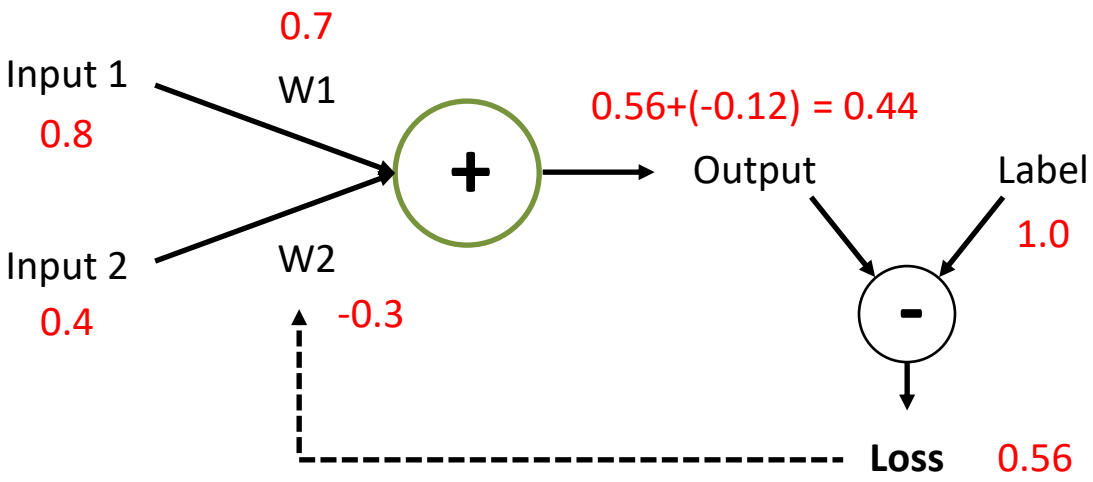
$$Output = [W1 \quad W2] \begin{bmatrix} Input\ 1 \\ Input\ 2 \end{bmatrix}$$

Training a perceptron



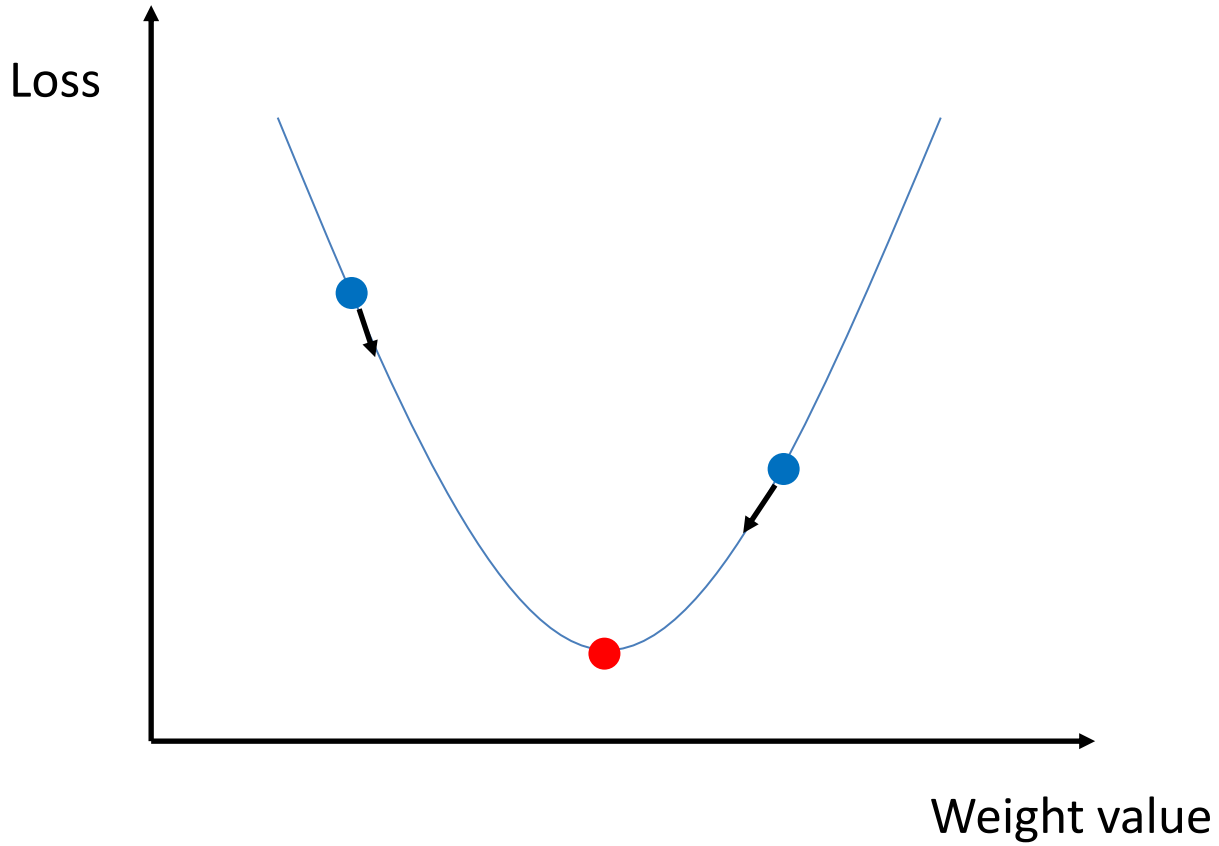
- ✓ Calculate the difference between the label (truth) and the output.
- ✓ This is the **loss**.
- ✓ Adjust the weights (behaviour) so that the output equals the label.
- ✓ You are **minimising the loss**.

Training a perceptron



- ✓ The weights represent “trust” for each input.
- ✓ A negative weight means to take the opposite of the input.
- ✓ The loss is used to update the weights W1 and W2.
- ✓ How do you change the weights to minimise the loss?

Gradient Descent



Backpropagation using Gradient Descent

- ✓ Tweak the weights one by one to see which change reduces the loss.
- ✓ Essentially finding the rate of change of the loss with respect to a weight.
- ✓ Toy loss function:

$$Loss = Label - Output$$

$$= Label - ((W1 * Input\ 1) + (W2 * Input\ 2))$$

Therefore

$$\frac{d\ Loss}{d\ W1} = -Input\ 1 \qquad \frac{d\ Loss}{d\ W2} = -Input\ 2$$

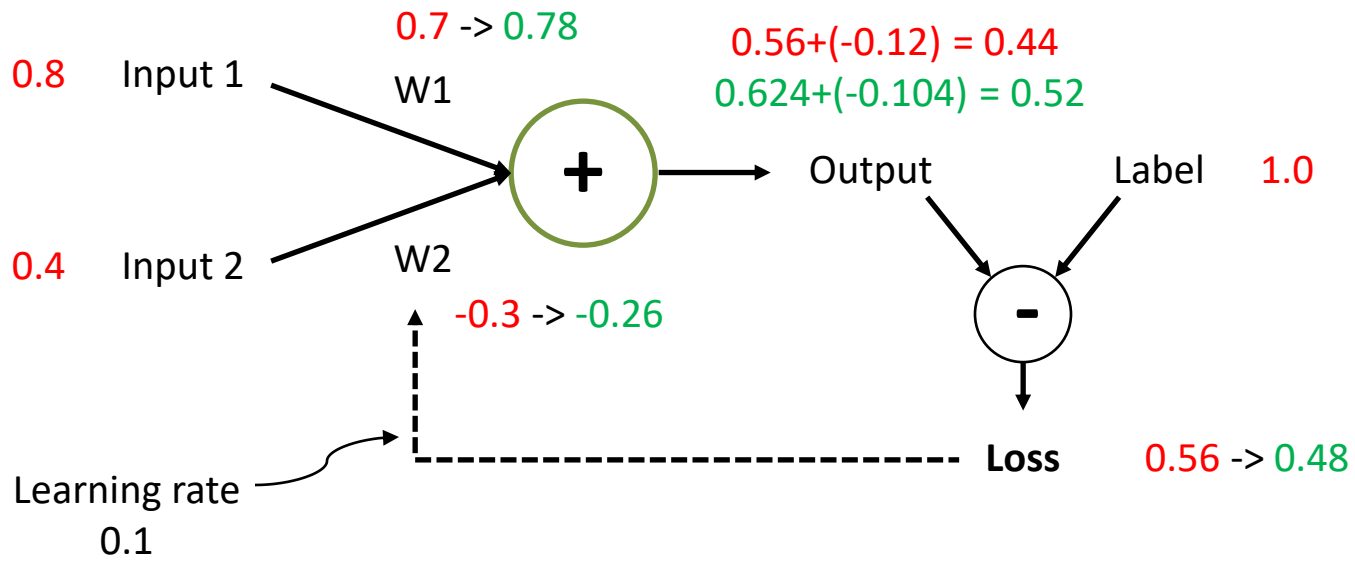
- ✓ You want to move in the direction of decreasing gradient, hence gradient **descent**.

$$new\ W1 = W1 - L\frac{d\ Loss}{d\ W1} = W1 - L(-Input\ 1)$$

$$new\ W2 = W2 - L\frac{d\ Loss}{d\ W2} = W2 - L(-Input\ 2)$$

L = Learning rate

Backpropagation using Gradient Descent

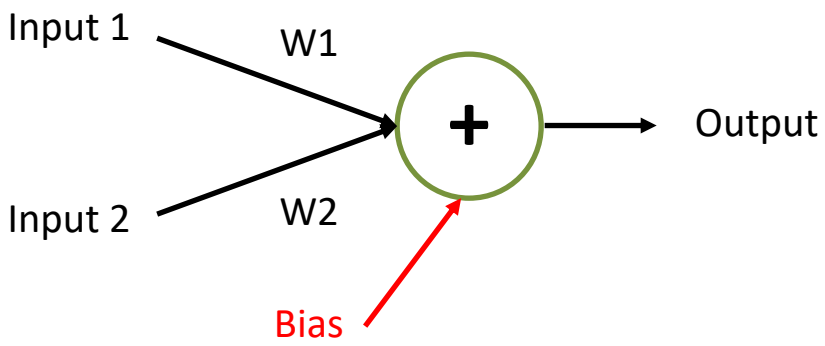


$$new\ W1 = W1 - L(-Input\ 1) = 0.7 - 0.1(-0.8) = 0.78$$

$$new\ W2 = W2 - L(-Input\ 2) = -0.3 - 0.1(-0.4) = -0.26$$

Loss has reduced from 0.56 to 0.48

Adding a Bias

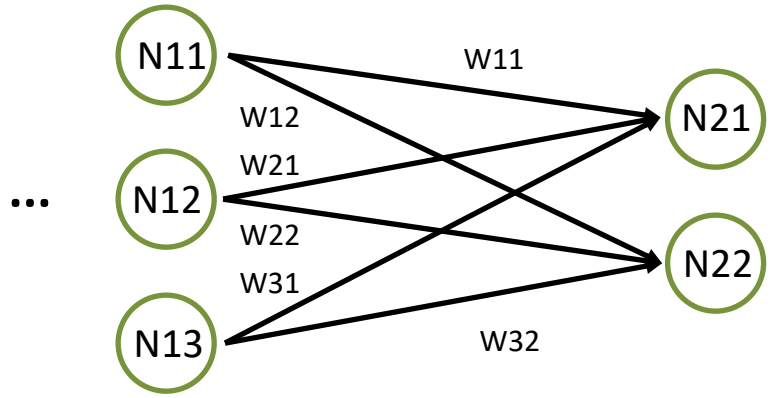


$$Output = (W1 * Input\ 1) + (W2 * Input\ 2) + Bias$$

In matrix form

$$Output = \begin{bmatrix} W1 & W2 & Bias \end{bmatrix} \begin{bmatrix} Input\ 1 \\ Input\ 2 \\ 1 \end{bmatrix}$$

Multilayer Perceptron



$$N21 = (N11 * W11) + (N12 * W21) + (N13 * W31) + Bias1$$

...

$$N22 = (N11 * W12) + (N12 * W22) + (N13 * W32) + Bias2$$

In matrix form

$$[N21 \quad N22] = \underbrace{[N11 \quad N12 \quad N13 \quad 1]}_{\text{\# of inputs}} \begin{bmatrix} W11 & W12 \\ W21 & W22 \\ W31 & W32 \\ Bias1 & Bias2 \end{bmatrix}$$

\# of nodes