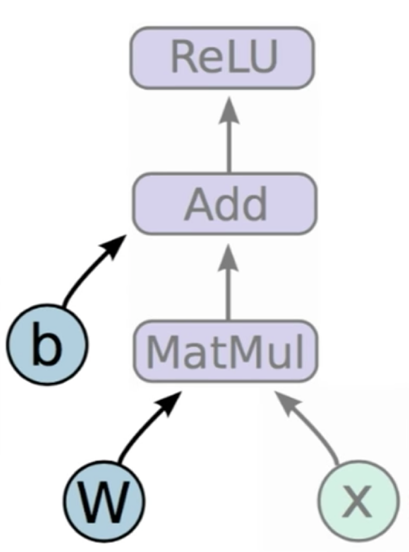
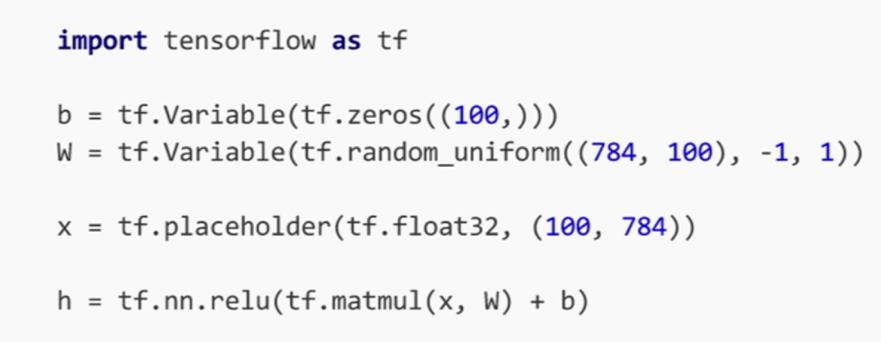
Lecture 7 | Introduction to Tensorflow

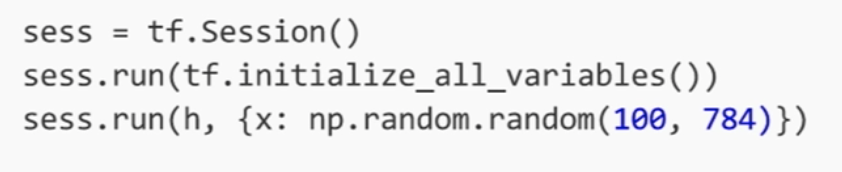
* Express numeric computation as a computation graph
  + Graph nodes are operations – which have any number of inputs and outputs
  + Graph edges are tensors – which flow between nodes



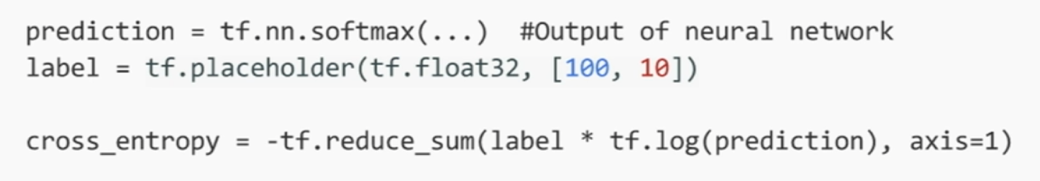
* + **(W and b)** are stateful nodes which output their current value, where state is retained across multiple executions of a graph. Gradient optimisation is automatic on all the variables. Variables are the things that you want to tune to minimise loss
  + **Placeholders (x)** are nodes whose value is fed in at execution time – our input
  + **Mathematical operations (MatMul, Add, ReLU)**

****

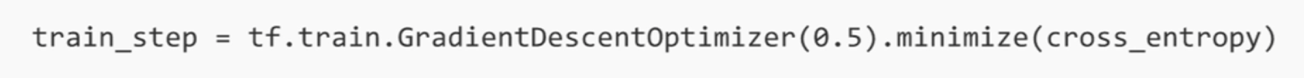
* + **b** is a tensorflow variable that takes in initial zeros of size 100, vector of 100 zeros
  + **W** is a tensorflow variable, taking uniformly distributed values between -1 and 1 of shapes 184 x 100
  + **X** is a tensorflow placeholder that takes in data type 32bit floats and has a shape of 100 x 784
  + **h** is the relu mathematical operation, which takes the output of the matmul operation between x and W and addition with b as an input
  + **Key notes:** By executing the code above, we haven’t actually manipulated any data yet. We are actually only building the symbols inside our graph. Therefore, no data is inputted into our system yet
* Where is the graph?
  + New nodes are automatically built into the underlying default graph
  + **tf.get\_default\_graph().get\_operations()** will show you all the nodes in your graph
* How do we run the graph?
  + We can deploy this graph with a **session**, a binding to a particular execution context (e.g. CPU/GPU). Therefore, we are going to deploy the graph on to a CPU or a GPU
  + sess.run(fetches, feeds)
    - **fetches:** list of graph nodes. Return the outputs of these nodes
    - **feeds:** dictionary mapping from graph nodes to concrete values. Specifies the value of each graph node given in the dictionary
  + To run a session:



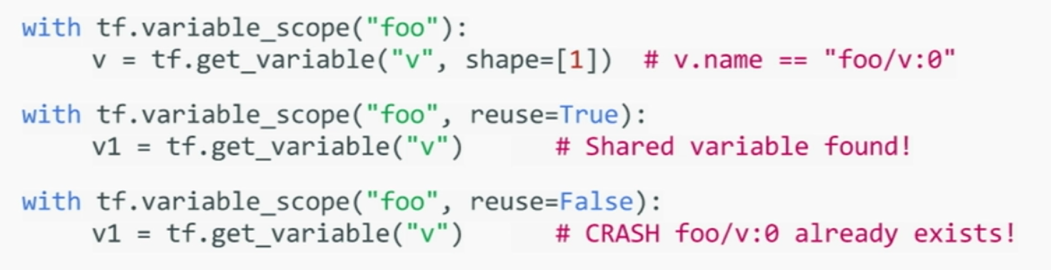
* How do we define the loss?
  + Use **placeholder** for **labels**
  + Build loss node using labels and predicitons



* How do we compute gradient?



* + tf.train.GradientDescentOptimizer() is an **Optimiser** object
  + 0.5 is our learning rate
  + The code line above adds optimisation operation to our computation graph
  + When we sess.run(train\_step), it is going to apply gradients onto all of the variables in our model. This is because the .minimise() function actually does two things in Tensorflow. First, it computes the gradient of our argument, cross\_entropy, w.r.t to all the variables in our graph. Then it is going to apply the gradient update to all those variables
  + Every graph node has an attached gradient operation. Gradient w.r.t. parameters computed with backpropagation
* Variable sharing
  + **tf.variable\_scope()** – provides simple name\_spacing to avoid clashes
  + **tf.get\_variable()** – creates/accesses variables from within a variable scope

****