National University of Singapore
School of Computing
CS5242: Neural Networks and Deep Learning
Tutorial 1: PYTORCH TENSORS

WHAT IS PYTORCH? It's a Python-based scientific computing package targeted at two sets of audiences:

- **A replacement for NumPy to use the power of GPUs**
- **A deep learning research platform that provides maximum flexibility and speed**

Tensors are similar to NumPy's ndarrays, with the addition being that Tensors can also be used on a GPU to accelerate computing.

Q1. Create a tensor `"a"` initialized from the list `[2, 5, 8, 9]`. Now print attributes of tensor `"a"` i.e `shape, dtype, device`. Convert it to Numpy array `"e"`.

Q2. Create a tensor `"b"` of size `4 * 5` initialized with all `0's`. Add `2` to `"b"`. Create another tensor `"c"` of the exact same as `"b"` initialized with all ones. Now create tensor `"d = b + c"`.

Q3. Create a Numpy array `"f"` of size `5 * 4` initialized with random numbers. Convert it to tensor `"g"`. Create `"h"` similar to `"f"` with random numbers. Transpose `"h"` and do matrix multiplication of `"h"` and `"f"`.

Q4. Deep Learning is based on artificial neural networks which are typically built with units called "neurons". Each neuron has some number of weighted inputs. These weighted inputs are summed together then passed through an activation function to get the unit's output.

1. Define a 1-D tensor `"X"` containing numbers from `1 to 5`.
2. Define a 1-D tensor `"W1"` containing `5` random weights.
3. Define a tensor `"b"` contained `1` random weight.
4. Compute tensor `"H = X * W1 + b"` using pytorch. (Note the shape of tensors)
5. Apply sigmoid activation function to calculate the `"Y"`, such that `"Y = 1 / (1 + exp(-H))"`

You have just coded to write a neuron !!

Q5. Adding multiple neurons in a single layer (3 outputs)

1. Define 2-D tensor `"W2"` containing random numbers of shape `5 * 3`
2. Define 1-D tensor of `"b2"` containing 3 random numbers.

3. Compute tensor “`H2 = [h1 h2 h3]`” using the “`H2 = X * W2 + b2`” using pytorch
4. Apply sigmoid activation function to calculate “`Y2 = [y1 y2 y3]`” such that “`Y = 1 / (1 + exp(-H2))`”

Now there are 3 neurons outputs!!

Q6. Stacking neurons on top of others. The real power of a neural network comes when we apply one function on top of others. (3 output units, 3 neurons in the previous layer)

1. Define 2-D tensor “`W3`” containing random numbers of shape `3 * 3`
2. Define 1-D tensor “`b3`” containing `3` random number
3. Compute “`H3 = W3 * Y2 + b`”. Compute the expected shape of `H3`.
4. Apply softmax pytorch activation function to `H3`.

Now you have a neural network taking 5 inputs and giving 3 outputs and having 1 hidden layer!!