# Practice Exercise: PyTorch Programming and Artificial Neural Network

**Q1** (a) Initialize a tensor of shape $3 \times 3$ filled with random floating-point numbers.

(b) Convert this tensor to a Numpy array.

(c) Convert the Numpy array back into a PyTorch tensor.

(d) Show the outputs at each step.

**Q2** (a) Create two tensors $A$ and $B$, each of shape $4 \times 4$, with values ranging from 1 to 16 (inclusive) for $A$ and 16 to 1 (inclusive) for $B$.

(b) Perform matrix multiplication between $A$ and $B$.

(c) Index the result to extract the second row.

(d) Display the resulting tensor.

**Q3** (a) Define a tensor $x$ of shape $2 \times 2$, filled with random values, and requires gradient computation.

(b) Define a function $f(x) = x^2 + 3x + 2$ and apply it to $x$.

(c) Compute the gradient of $f$ with respect to $x$ using backpropagation.

(d) Print the gradient of $x$.

**Q4** (a) Check if a GPU is available and move a tensor of shape $5 \times 5$, filled with ones, to the GPU.

(b) Define a simple linear model with a single layer having input and output dimensions of 5.

(c) Use SGD (Stochastic Gradient Descent) from PyTorch's optim package as the optimizer for the model, with a learning rate of 0.01.

(d) Perform a single step of the optimization (hint: you may need to define a dummy target tensor for this purpose).

(e) Display the model's weights after the optimization step.

**Q5** Using PyTorch, train a neural network with only one hidden layer containing 6 hidden units that can learn the classification output depicted in the following Table 1. In order to build a robust classifier, try adding Gaussian noise to the data points. Train the classifier on 4000 data points and

test it on 500 samples. While implementing the classifier, please check the following points. (**Hint:** Please have a look on the neural network classifier implemented for learning the output of continuous XNOR function in the PyTorch Tutorial notebook.)

| $A$ | $B$ | $C$ | $A \oplus (B \wedge C)$ |
|---|---|---|---|
| 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 |

Table 1: Truth table of 3 variables

(a) Calculate the classification accuracy. Try removing any nonlinearity from the classifier and note the difference in performance.

(b) Try including the following nonlinearities and note the difference in performance: Tanh(), ReLU(), Sigmoid().

**Q6** Extend the above neural network classifier that can classify data points according to the following function $f(A, B, C, D) = (A \oplus (B \vee C)) \odot D$. Similar to the above question, add Gaussian noise to the data points. Train the classifier on 4000 data points and test it on 500 samples.

(a) Calculate the classification accuracy. If your classification accuracy doesn't reach 100%, try increasing the number of training data points, number of hidden units and hidden layers.