Question #01

NumPy can be used to perform a wide variety of mathematical operations on arrays.

PyTorch is an open source machine learning (ML) framework based on the Python programming language and the Torch library.

Torchvision is a library for Computer Vision that goes hand in hand with PyTorch.

Torchvision provides many **built-in datasets** in the torchvision.datasets : CIFAR10, CIFAR100, MNIST, etc.

Matplotlib helps to see my graphs visually. (**%inline** : in this command **%** is magic command which says show my graph in Jupyter no as a pop-up…)

Augmentation = transformation is a technique of artificially increasing the training set by creating modified copies of a dataset using existing data. It includes making minor changes to the dataset or using deep learning to generate new data points.

There are a lot of built in transforms : *https://pytorch.org/vision/stable/transforms.html*

Compose class helps to apply several transforms.

PIL stands for **Python Imaging Library,** and it's the original library that enabled Python to deal with images.
Data loader. Combines a dataset and a sampler, and provides an iterable over the given dataset. (Main functionality is Batches : a technique to help coordinate the update of multiple layers in the model).
Matmul == mm : used to get derivatives.

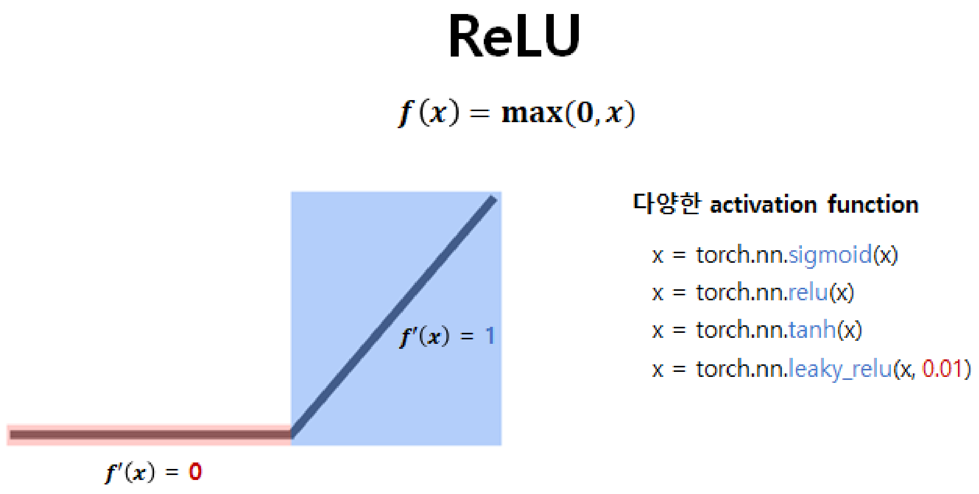
PyTorch contains a torch. **The nn module** is used to train and build the layers of neural networks such as input, hidden, and output. Torch. nn base class helps wrap the torch's parameters, functions, and layers.

an `__init__()` call to the parent class must be made before assignment on the child.

In the NN module, **forward()** function does the actual message passing and computation.

ReLU :

A rectified linear unit (ReLU) is an activation function that introduces the property of non-linearity to a deep learning model and solves the vanishing gradients issue.



```
x = torch.nn.sigmoid(x)
x = torch.nn.relu(x)
x = torch.nn.tanh(x)
x = torch.nn.leaky_relu(x, 0.01)
```

Epoch is when a model sees all examples once.

Images in the dataset = 100; Batch_size = 10; Iteration - 100 // 10 = 10? In this case, 10 iterations == 1 epoch;

Images in the dataset = 100; Batch_size = 10; Epoch = 10; Iteration - 1 epoch has (100 // 10) = 10 iterations, then 10 epochs have 10 * 10 = 100

Cross-entropy loss is used when adjusting model weights during training. The aim is to minimize the loss, i.e, the smaller the loss the better the model. A perfect model has a cross-entropy loss of 0.
(https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html)

Intuitively Understanding the Cross Entropy

$$H(P^* | P) = -\sum_i \underbrace{P^*(i)}_{\text{TRUE CLASS DISTIRBUTION}} \log \underbrace{P(i)}_{\text{PREDICTED CLASS DISTIRBUTION}}$$

1) The main **difference** between **CPU and GPU** architecture is that a CPU is designed to handle a wide-range of tasks quickly (as measured by CPU clock speed), but are limited in the concurrency of tasks that can be running. A GPU is designed to quickly render high-resolution images and video concurrently.

2) The Adam optimizer is also an optimization technique used for machine learning and deep learning, and comes under gradient descent algorithm.