Assignment #7

TensorFlow and **PyTorch** are two of the most popular deep learning frameworks, each with its own strengths and weaknesses. Here's a comparison between them:

- **Ease of Use and Flexibility**:
  # **TensorFlow** traditionally had a steeper learning curve due to its static graph computation paradigm. However, with the introduction of TensorFlow 2.0, eager execution mode became default, making it more intuitive and similar to PyTorch.
  # **PyTorch** has been praised for its simplicity and flexibility. Its dynamic computation graph allows for easier debugging and more straightforward model construction.

- **Community and Ecosystem**:
  # **TensorFlow** has a larger user base and extensive documentation. It's backed by Google and has been used in numerous production deployments across various industries.
  # **PyTorch** has gained rapid adoption, particularly in research settings. Its community is known for its responsiveness and contributions to the ecosystem.

- **Model Deployment**:
  # **TensorFlow** offers TensorFlow Serving for deploying models in production environments. It also supports TensorFlow Lite for deploying models on mobile and embedded devices.
  # **PyTorch** provides TorchScript for model deployment and supports integration with ONNX for interoperability with other frameworks and deployment platforms.

- **Debugging and Visualization**:
  # **TensorFlow** has TensorBoard, a powerful visualization tool for monitoring training progress and inspecting models.
  # **PyTorch** has tools like Visdom and TensorBoardX for visualization, though its visualization ecosystem isn't as mature as TensorFlow's.

- **Performance**:
  # Both **TensorFlow** and **PyTorch** offer similar performance for most deep learning tasks. Performance differences are often negligible and depend more on the specific implementation rather than the framework itself.

- **Production Readiness**:
  # **TensorFlow** has been widely adopted in production environments and has comprehensive support for deployment tools and platforms.
  # **PyTorch** has made significant strides in improving its production readiness, but TensorFlow still holds an edge in this regard due to its longer history and support from Google.

- **Research vs Production**:
  # **PyTorch** is often favored in research settings due to its flexibility and ease of experimentation.
  # **TensorFlow** is preferred in production environments where scalability, performance, and deployment options are crucial.

In summary, both TensorFlow and PyTorch are powerful frameworks with their own strengths. The choice between them often comes down to factors such as ease of use, community support, and specific project requirements. Many practitioners use both frameworks depending on the task at hand and their individual preferences.

For all your work, submit a Notebook (either Jupyter or Colab.)

Demonstrate on the following toy dataset [consists of grayscale images of handwritten digits (0-9)] how to build, train, and evaluate a simple neural network for the MNIST dataset **using TensorFlow and PyTorch**. Both frameworks offer similar functionality, but the syntax and APIs are different.

<u>**Phases**</u>:
(**#1**) **Import necessary libraries**.

(**#2**) **Load and preprocess the dataset**:
TF:
mnist = tf.keras.datasets.mnist

PT:
trainset = torchvision.datasets.MNIST(root='./data', train=True, ….)
trainloader = torch.utils.data.DataLoader(trainset, …)

testset = torchvision.datasets.MNIST(root='./data', train=False, ….)
testloader = torch.utils.data.DataLoader(testset, …)

As far as preprocessing, just perform a **Normalization** task.

(**#3**) **Define and build the models**.
Be creative, create as many hidden layers as you see fit.
Be careful with the type of Activation Functions used on each layer.

(**#4**) **Define Loss Function, Optimizer, and Metric**.
Loss = CrossEntropy
Optimizer = ADAM w/ lr = 0.001
Metric = Accuracy

(**#5**) **Train the models**.
Fit the models for 100 epochs.

(**#6**) **Evaluate the models**.
Print out the 'accuracy' for both models (TF, PT). Which one is better? Any possible tuning?