Assignment #1: Pytorch Tutorials

Hanyang University
Sooyoung Kim[redacted], Gaon Choi[redacted]

Introduction

1. Visualizing models, data, and training with Tensorboard

Tensorboard is a tool designed for visualizing the results of neural network training runs. In this experiment, we will see some of its functionality, using the Fashion-MNIST dataset which can be read into Pytorch using torchvision.datasets. There are five specific goals for this section.

- ✓ Read in data and with appropriate transforms(nearly identical to the prior tutorial)
- ✓ Set up TensorBoard
- ✓ Write to TensorBoard
- ✓ Inspect a model architecture using TensorBoard
- ✓ Use TensorBoard to create interactive versions of the visualizations with less code

2. TRANSFER LEARNING FOR COMPUTER VISION TUTORIAL

In general, a large amount of data is required to create an available DNN.

However, it is not easy to produce and process the desired data, and it will take more time and effort to prepare the dataset than to produce DNNs. The technique of customizing by modifying a dataset verified as a solution to this problem and some of the pre-learned models is called transfer learning.

In this tutorial, the two main scenarios of transfer learning to practice are as follows.

1. Fine Tuning the convnet: Fine Tuning refers to a method of transforming an architecture to suit a new purpose and finely adjusting the weight of an already learned model based on an existing model.
2. ConvNet as fixed feature extractor: A method of re-learning only the last fully connected layer while maintaining both existing learned models and weights.

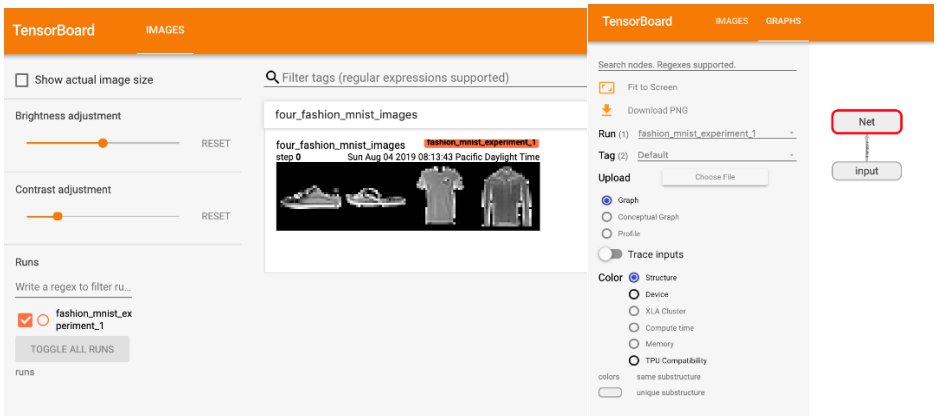
Experiment

1. Visualizing Models, Data, and Training with TensorBoard

One example of Fashion-MNIST is as follows.

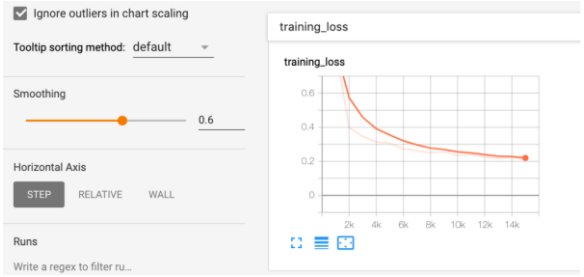
```
# write to tensorboard
writer.add_image('four_fashion_mnist_images', img_grid)
```

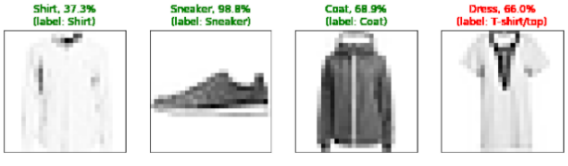
Loading the TensorBoard, we can adjust the image with the interactive GUI. Previously, we defined a class, named Net. We can check the visual structure of our model in the second picture.



Each image of Fashion-MNIST is 28 x 28 pixel image. Transforming these image into vectors, we can get 784-dimensional vector, which is relatively high dimensional data. Using add_embedding method, we can visualize the higher dimensional data into lower dimensional representation. Using "projector", we can see the projected form into three dimensional space of 784 dimensional data. Also, we can click and drag to rotate the three dimensional projection as below.

Let's think about the step of training using the actual dataset after the model design is completed. It will be necessary to check the pattern of decreasing loss within the designed model for each iteration. In addition, there are times when it is necessary to check whether the data in the validation set has been properly tested after all the training is completed. Such tasks can be visually performed through the TensorBoard.

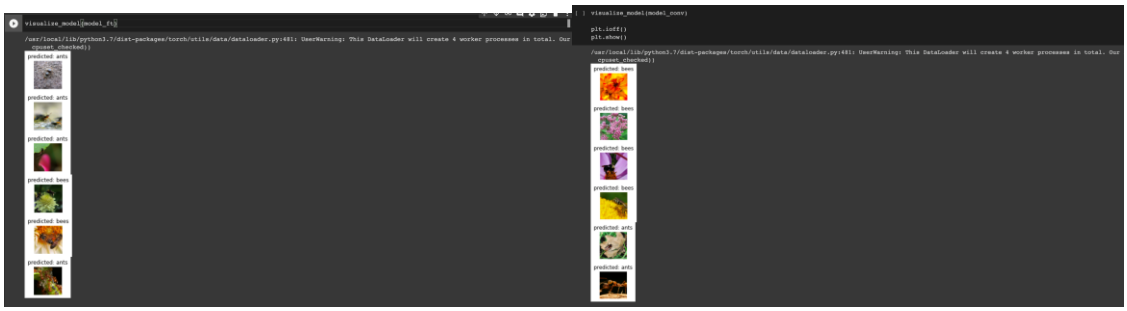




2. TRANSFER LEARNING FOR COMPUTER VISION TUTORIAL

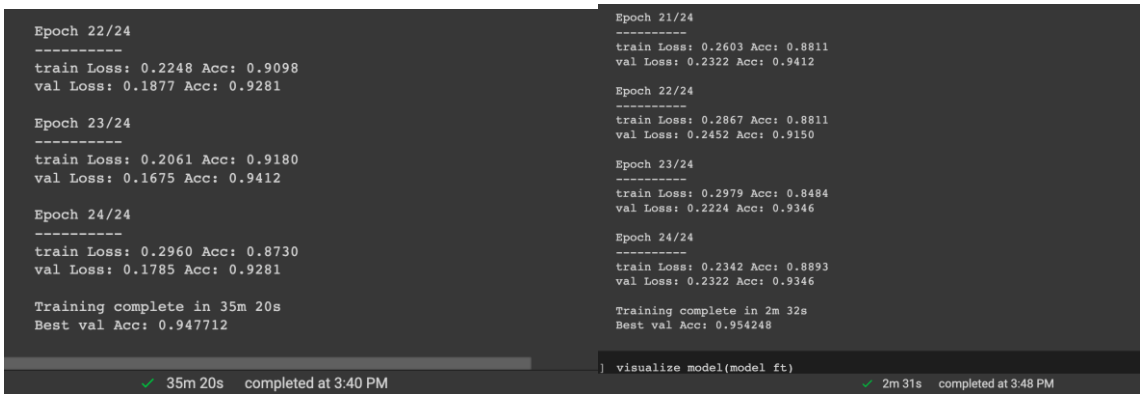
The results of both tutorials are as follows.

- (Left) Result of "Fine Tuning the convnet" / (Right) Result of "ConvNet as fixed feature extractor"



In addition, in order to compare the learning time according to the Runtime Type among the contents presented in the tutorial, the execution time was measured based on "Fine Tuning the convnet."

As a result of the measurement, it took **35 minutes and 20 seconds to use the CPU** as follows, while it took **2 minutes and 31 seconds to use the GPU**, so I could feel a clear difference.



* The tutorial was all run within colab

Conclusion

After studying TensorBoard, we realized that it is an useful idea to visually illustrate models. In designing models, training real datasets, one of the important steps is investigating the characteristics and statistics(e.g. distribution) of given datasets. After that, we can design a model with appropriate structure, that considers the datasets.

Through PyTorch's "TRANSFER LEARNING FOR COMPUTER VISION" Tutorial, we could feel the two main scenarios of transfer learning, and the importance of hardware acceleration once again through the difference in GPU and CPU operations.