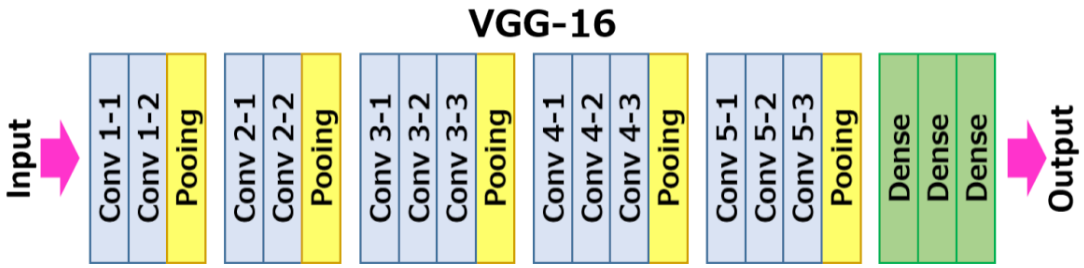
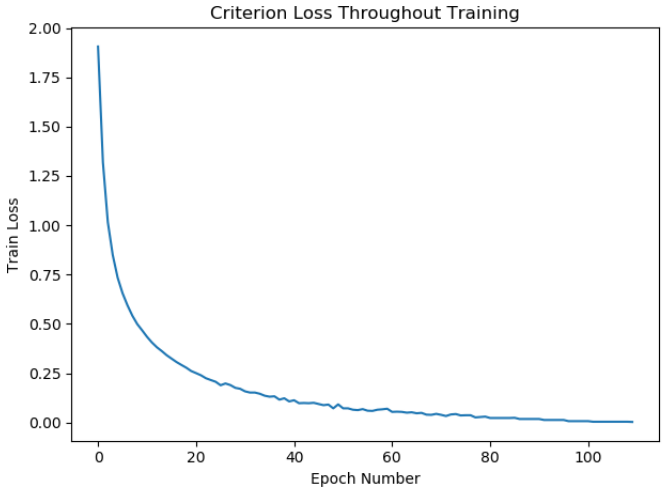
For this problem, I implemented using PyTorch a convolution neural network (CNN). The architecture of the problem used was based on the implementation of successful VGG 16 model [1] which upon my literature investigation I figured that it can perform relatively well on CIFAR10 dataset. The architecture of VGG16 is a sequential model with all the layers being arranged in a sequence with 2 x convolution layer of 64 channels each and a 3x3 kernel and same padding. Followed by a maxpool layer of 2x2 and stride of 2x2. This is then followed by 2 x convolution layer of 128 channels of 3x3 kernel and same padding followed by a maxpool layer of 2x2 and stride of 2x2. Then followed by this time 3 x convolution layer of 256 channel of 3x3 kernel and same padding followed by maxpool layer of 2x2 and stride of 2x2. Then we have 3 x covolution layer of 512 channel of 3x3 kernel and same padding followed by maxpool layer of 2x2 and stride of 2x2. And lastly another 3 x convolution layer of 512 channel and 3x3 kernel and same padding followed by maxpool layer of 2x2 and stride of 2x2. Notice that each layer also has a ReLU activation. We then have self.classifier = nn.Linear(512, 10) at the end. The image below taken from NeuroHive (https://neurohive.io/en/popular-networks/vgg16/) is a close resemblance of the visual representation of the architecture:



The visualization of the training process is given below where the criterion loss is defined as Cross-Entropy Loss:



The hyperparameters tuned were number of training epochs, and the learning rate the choice of gradient descent algorithm and the normalization of each image. I normalized each image as well as added

cropping and random horizontal flips for the training data to improve the generalization done during the training. The normalization array used for each image (training and test) were transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)) following the investigation done from the previous literature.

I used Stochastic Gradient Descent (SGD) with momentum. The value of momentum coefficient β was set to 0.9. The learning rate was set to 0.001. I run the training for 110 epochs. I also captured a running loss throughout training for each epoch for every 2000 inputs (up to 12000). The reported visualization of training above is based on the average of 6 data points at each epoch. The main structure of the code was taken from CIFAR10 PyTorch tutorial that was instructed in the problem statement and my code was based on editing this base structure and implementing the ideas of [1] for the accuracy reported below. (https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html)

The achieved final accuracy of on 10,000 test images is: **88%**

(Accuracy of plane: 91%, car: 96%, bird: 82%, cat: 79%, deer: 85%, dog: 83%, frog: 88%, horse: 93%, ship: 90%, truck: 94%).

References:

[1] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).