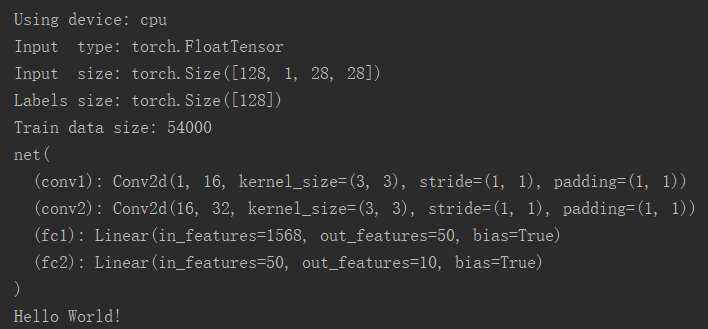
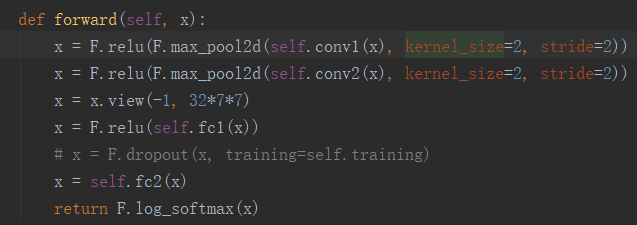
# Lab 2: Pytorch Pipeline

**Initial Template (1p)**

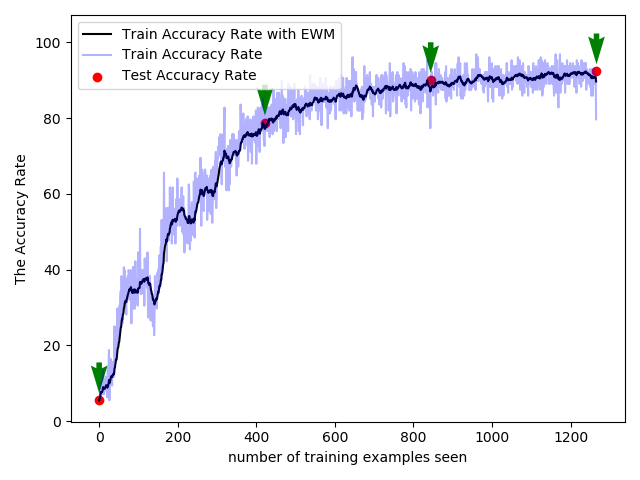


I used Pycharm, so the activation function, the softmax function and the loss function are attached. Because the F.log\_softmax combined these two function together for convenient computing.

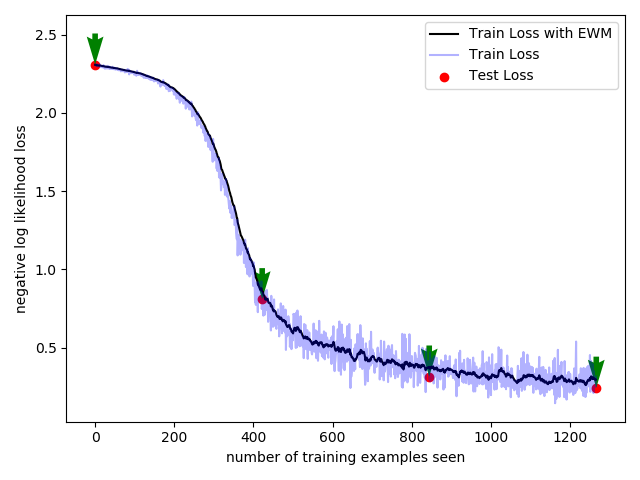


### Full Pipeline (5p)

This figure below shows the accuracy rate. Red dot are the accuracy rate using Test dataset after each one epoch. Black line is the accuracy curve with EWM, and the partially transparant blue line is the raw accuracy curve.



This figure below shows the loss curve. Red dot are the loss using Test dataset after each one epoch. Black line is the loss curve with EWM, and the partially transparant blue line is the raw loss curve.

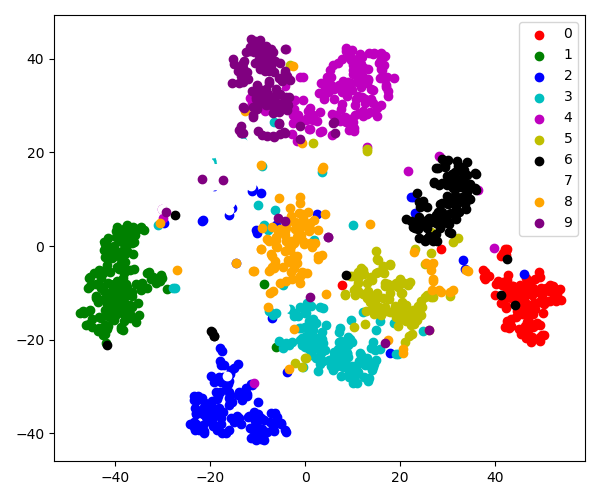


And the **history.pickle**, the **checkpoint\_epoch1.checkpoint** dictionary, **model\_cuda.pth**, and the **optimizer\_cuda.pth** are saved in the result fold.

### Analyzis (4p)

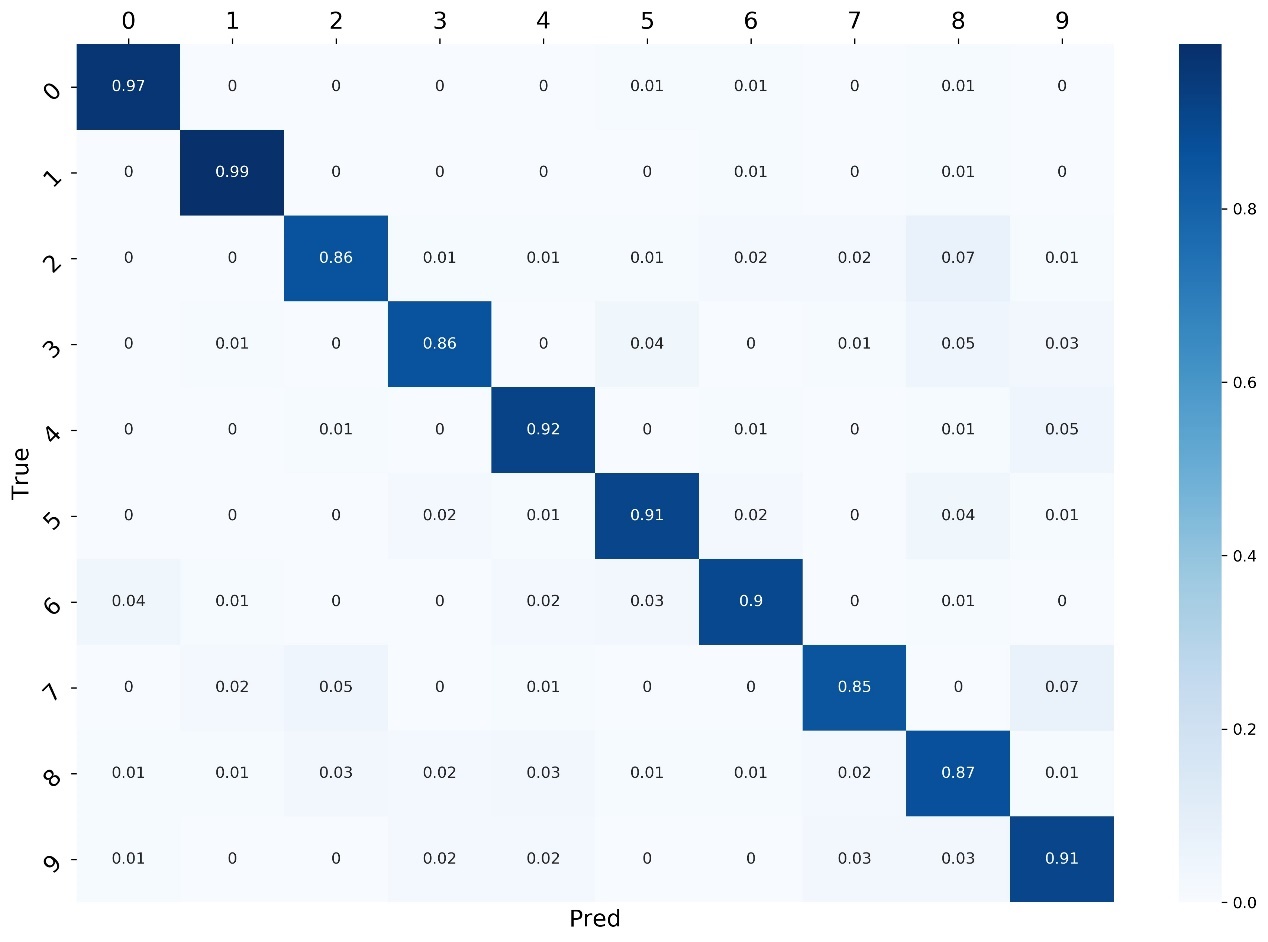
Although the boundary is not so perfect, when using **nearest neighbor classifie,** the accuracy rate can reach 0ver 90%.

from sklearn.neighbors import KNeighborsClassifier  
neighs = KNeighborsClassifier(n\_neighbors=3)  
neighs.fit(X, Y)  
neighs\_tsne = KNeighborsClassifier(n\_neighbors=3)  
neighs\_tsne.fit(X\_2d\_tsne, Y)  
A1 = neighs.predict(X\_last)  
A2 = neighs\_tsne.predict(X\_2d\_tsne\_last)

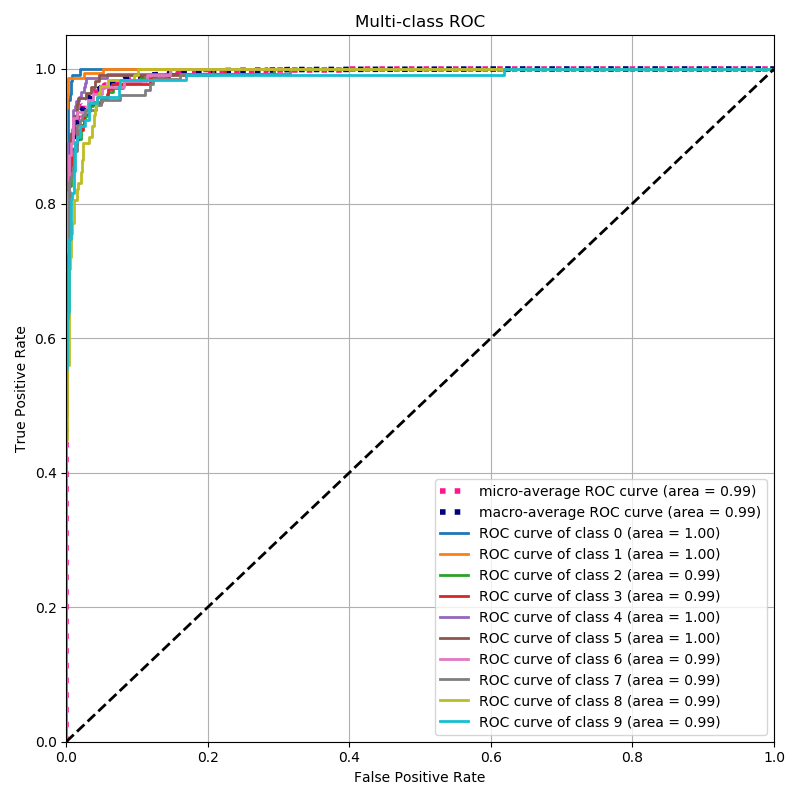


tSNE for MNIST data

From confusion matrix, we can see how many classes were wrongly classifed. From ROC curve, we can see the classification ability, which means if we accept a 5% error rate, all classes can achiece over 80% accuravy rate.



*Confusion Matrix*



*ROC Curve*