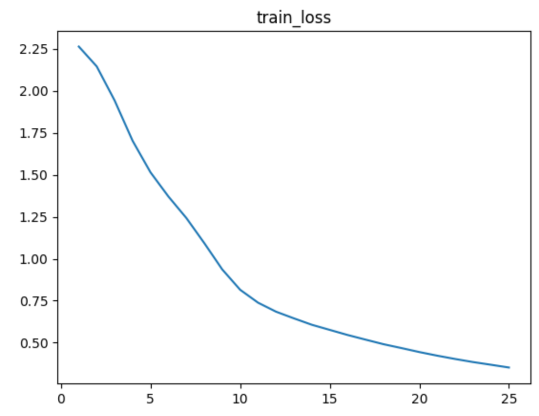
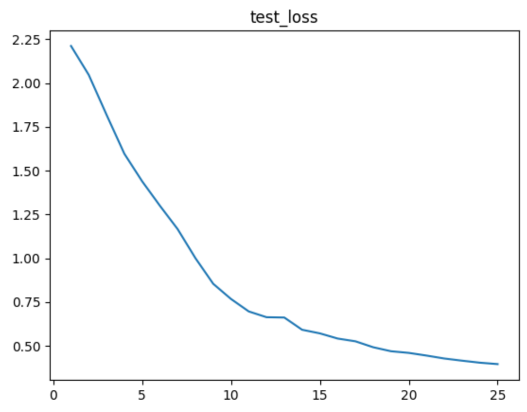
In this phase we trained a model based on pytorch CNN-Model. We used this network for first layer:

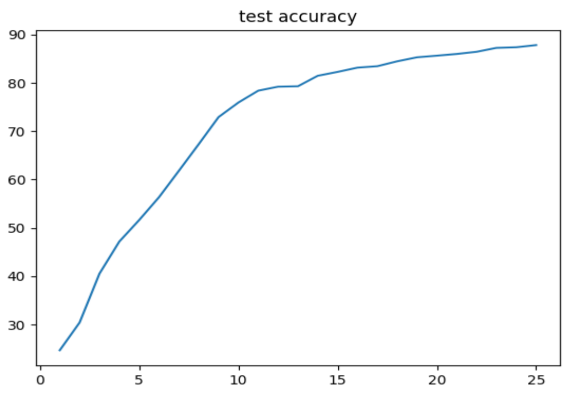
```
demo_model(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(6, 16, kernel_size=(3, 3), stride=(1, 1))
  (conv3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc1): Linear(in_features=3456, out_features=1024, bias=True)
  (fc2): Linear(in_features=1024, out_features=128, bias=True)
  (fc3): Linear(in_features=128, out_features=10, bias=True)
)
```

Conv layer had 3200 features and we added 256 more, on next step.

After that, we choose batch size = 32 and epoch = 25 and learning rate = 0.005 to create the model. in this part, chose CrossEntropy as loss function. Result filtered by number of epoch is the below plot:

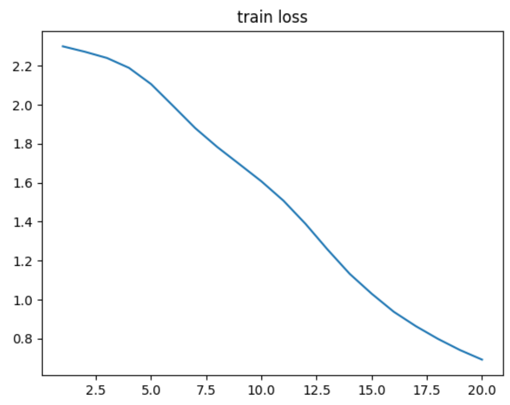


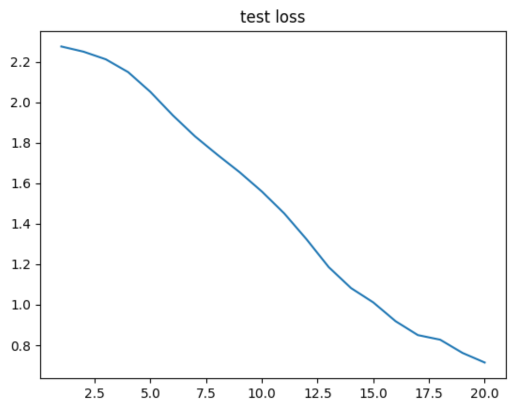


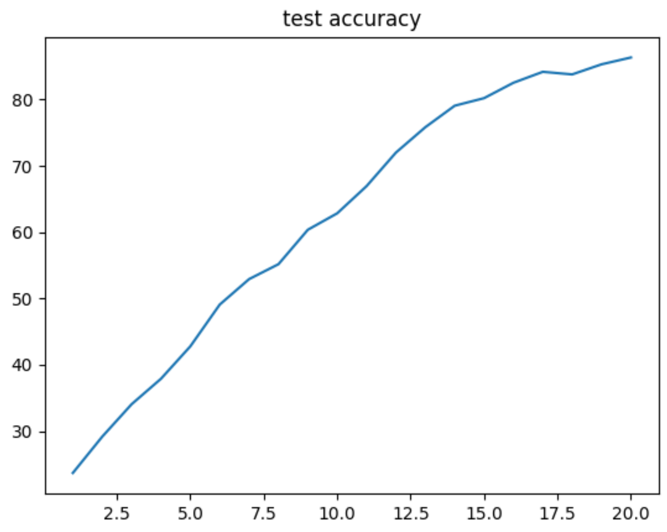


In the next part, we choose two loss function and implemented triplet loss, according to details explained in pdf. We used previous network and chosed landa equal to 0.1. also the margin is 0.005

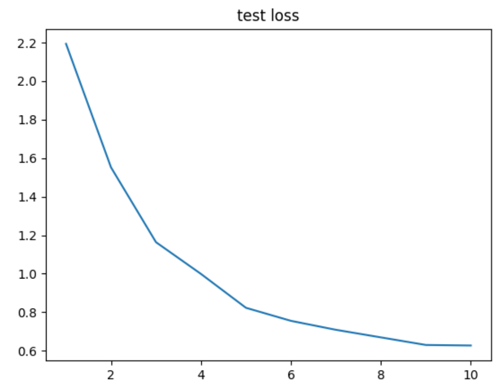
This time we did 20 epochs for training model. results are as below plots:

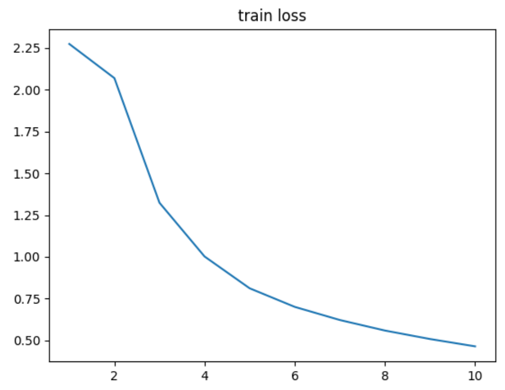


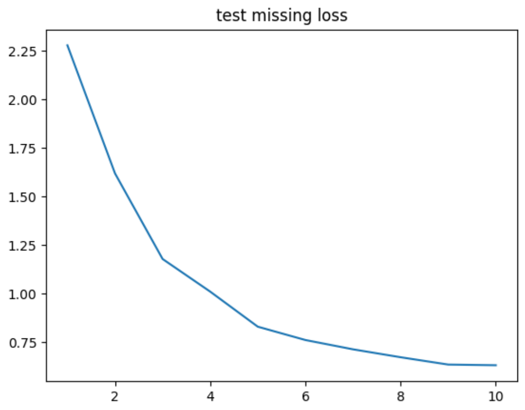


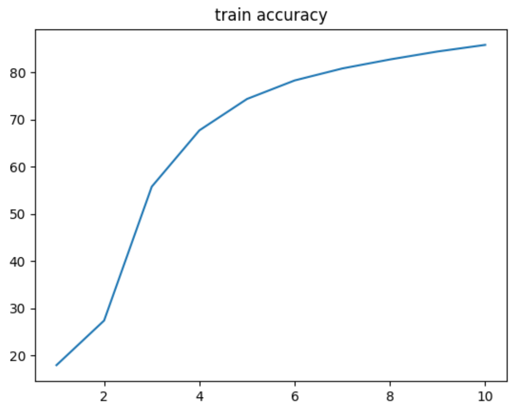


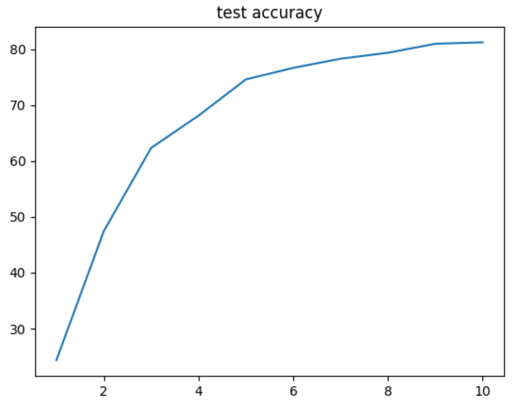
In bonus part, we first build a prepared model (cnn). The inputs were images and outputs were 256 features. So we could train this model due to find missed features. But in this method we got 13% accuracy which wasn't suitable. So we used another way. We train the model with both real features and zero like features first, then run test measures on it. Therefore we learn a new model which works for both test and test_missing datasets. We chose 10 epochs and results are:

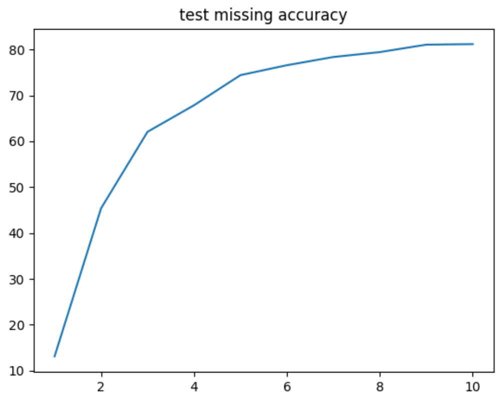





test missing loss


train accuracy


test accuracy


test missing accuracy