AI21 Term Project

소프트웨어전공
2018044993 임소윤

1. 작성 코드

Model

```
In [74]:
class MLP(nn.Module):
    def __init__(self):
        super().__init__()

        self.in_dim = 28*28 #MNIST
        self.out_dim = 10

        self.fc1 = nn.Linear(self.in_dim, 512)
        self.fc2 = nn.Linear(512,256)
        self.fc3 = nn.Linear(256,128)
        self.fc4 = nn.Linear(128,64)
        self.fc5 = nn.Linear(64,self.out_dim)

        self.relu = nn.ReLU()
        self.log_softmax = nn.LogSoftmax()

    def forward(self,x):
        z1 = self.fc1(x.view(-1, self.in_dim))
        a1 = self.relu(z1)
        z2 = self.fc2(a1)
        a2 = self.relu(z2)
        a3 = self.relu(self.fc3(a2))
        a4 = self.relu(self.fc4(a3))
        logit = self.fc5(a4)
        return logit,(z1,a1,z2,a2)
```

```
In [76]:
z1 = torch.Tensor([])
a1 = torch.Tensor([])
z2 = torch.Tensor([])
a2 = torch.Tensor([])
y = torch.Tensor([])
```

```
In [77]:
for epoch in range(10): #loop over the dataset multiple times
    running_loss = 0.0
    for i, data in enumerate(train_loader, 0):
        #get the inputs; data is a list of [inputs, labels]
        inputs, labels = data

        #zero the parameter gradients
        optimizer.zero_grad()

        #forward + backward + optimize
        outputs, toch = model(inputs)

        if (epoch == 9):
            z1 = torch.cat((toch[0],z1),0)
            a1 = torch.cat((toch[1],a1),0)
            z2 = torch.cat((toch[2],z2),0)
            a2 = torch.cat((toch[3],a2),0)
            y = torch.cat((labels,y),0)

        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        #print statistics
        running_loss += loss.item()
        if (i+1)%2000==0:    #print every 2000 mini-batches
            print('[%d, %5d] loss: %.3f' %
                  (epoch+1, i+1, running_loss/2000))
            running_loss =0.0

print('Finished Training')
```

forward함수를 통해 z1,a1,z2,a2를 return받는다. Torch에 각각의 return 값을 저장해주기 위해 torch.Tensor를 사용했고, visualization함수에 입력하기 위해 z1,a1,z2,a2를 정리해준다.

(코드 작성 시 마주한 문제점)

처음 코드를 작성할 때 return 값을 아래와 같이 하려고 했으나 결과를 불러오는 데에서 자꾸만 오류가 생겼다.

```
logit = self.fc5(a4)
return logit,z1,a1,z2,a2
```

위처럼 코드를 작성하여 tensor를 사용하지 않고 코드를 작성하였으나 마지막 visualization함수에서 오류가 발생하였다.
해당 오류를 해결하기 위해 z1, a1, z2, a2를 묶어서 return값으로 받고, 이를 torch.cat을 이용해 각각을 쪼개어 보는 것으로 해결하였다.

그 외 방법으로 따로 기존의 forward함수 외에 visualization을 위한 forward함수를 위와 같이 작성하였으나, 역시 visualization함수까지 끌고 가는 데에 어려움이 있었다.

(Visualization 함수)

해당 함수를 통해 x,y를 입력하면 pca와 tsne 도표를 볼 수 있다.

```
def visualization(x,y):
    x = (x.data.view(-1,x.data.shape[1]) / 255.0).numpy()
    y = y.numpy()
    print(f'x.shape : {x.shape}')
    print(f'y.shape : {y.shape}')

    feat_cols = [f'pixel{i}' for i in range(x.shape[1])]
    df = pd.DataFrame(x, columns = feat_cols)
    df['y'] = y

    np.random.seed(42)
    rndperm = np.random.permutation(df.shape[0])

    pca = PCA(n_components=2)
    pca_result = pca.fit_transform(df[feat_cols].values)
    df['pca-one'] = pca_result[:,0]
    df['pca-two'] = pca_result[:,1]
    print('Explained variation per principal component: {}'.format(pca.explained_variance_ratio_))

    plt.figure(figsize=(13,9))
    sns.scatterplot(
        x="pca-one", y="pca-two",hue="y",
        palette=sns.color_palette("hls",10),
        data=df.loc[rndperm,:],
        legend="full", alpha=0.3)

    N=10000
    df_subset = df.loc[rndperm[:N],:].copy()
    data_subset = df_subset[feat_cols].values
    pca = PCA(n_components=2)
    pca_result = pca.fit_transform(data_subset)
    df_subset['pca-one']=pca_result[:,0]
    df_subset['pca-two']=pca_result[:,1]
    print('Explained variation per principal component: {}'.format(pca.explained_variance_ratio_))

    time_start = time.time()
    tsne = TSNE(n_components=2, verbose=1, perplexity=40, n_iter=300)
    tsne_results = tsne.fit_transform(data_subset)
    df_subset['tsne-2d-one']=tsne_results[:,0]
    df_subset['tsne-2d-two']=tsne_results[:,1]
    print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time_start))
```

```
plt.figure(figsize=(13,9))
sns.scatterplot(
    x="tsne-2d-one", y="tsne-2d-two",hue="y",
    palette=sns.color_palette("hls",10),
    data=df_subset,
    legend="full", alpha=0.3)
plt.show()
```

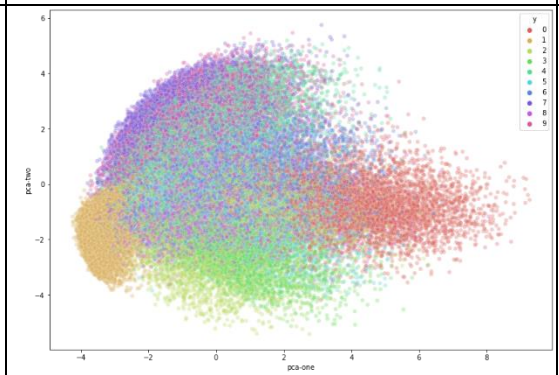
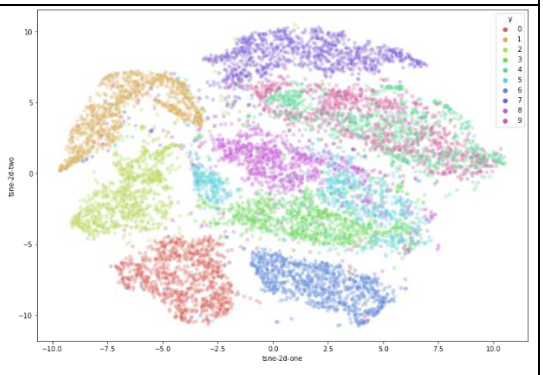
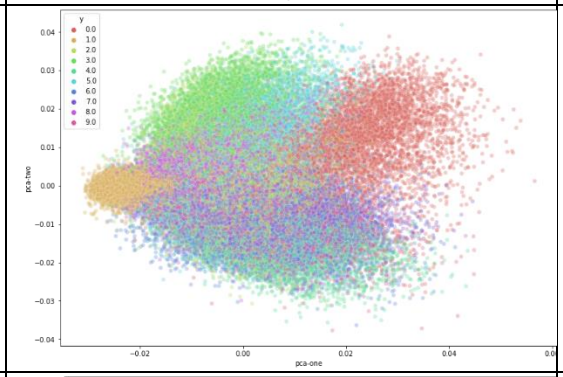
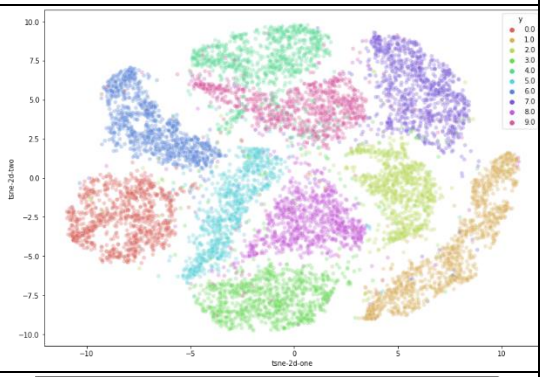
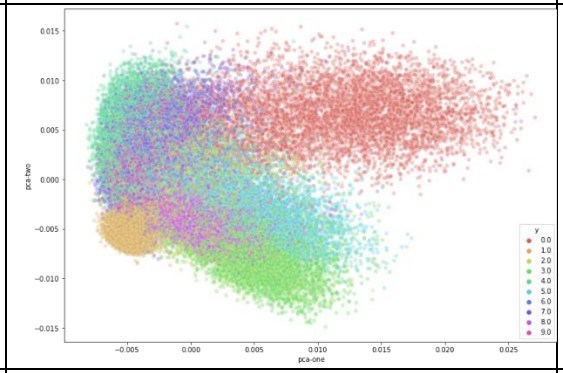
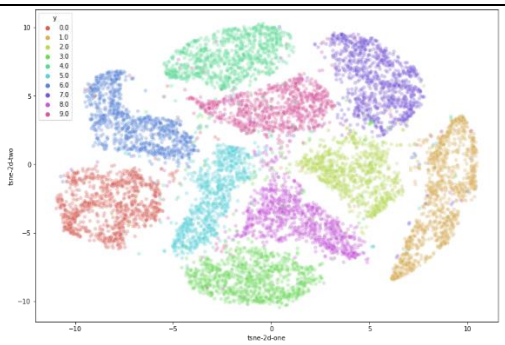
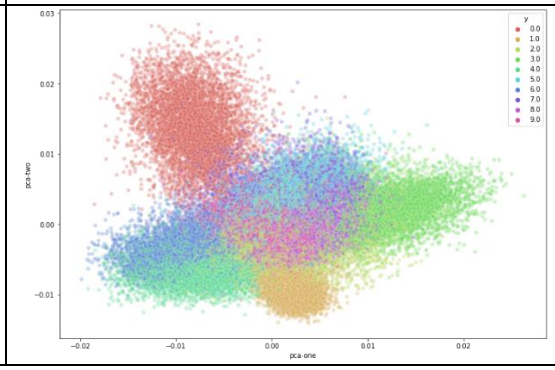
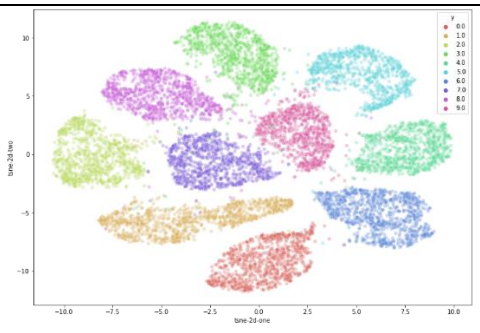
```
this=train_data.data
this=this.reshape(60000,784)
visualization(this,train_data.targets)
```

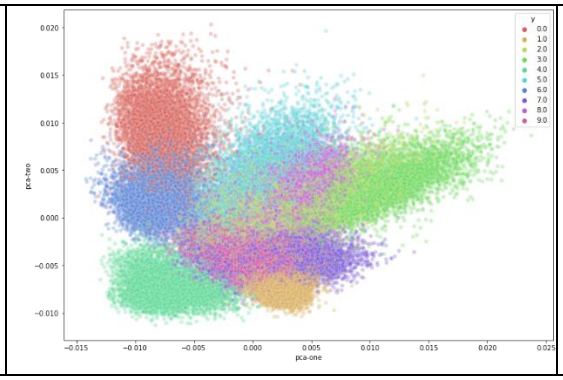
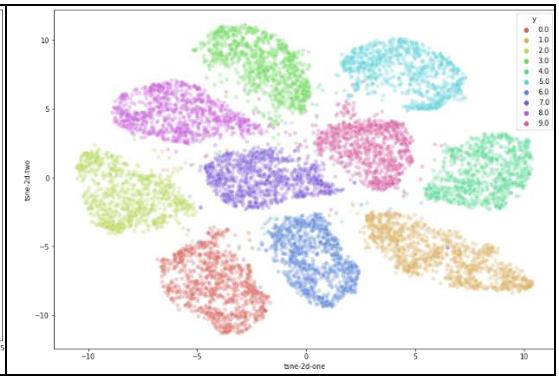
a[0]에 대한 결과를 받을 수 있다.

```
In [87]: visualization(z1,y)
```

이런 식으로 a1, z2, a2도 x의 자리에 입력해주면 x자리에 들어간 값에 대한 결과를 받아볼 수 있다.

2. 결과

	PCA	t-SNE
X(a[0])		
z[1]		
a[1]		
z[2]		

a[2]		

3. 결과분석

t-SNE의 결과를 보면 갈수록 점점 영역이 분리되는 것을 볼 수 있다. 그러나 PCA의 경우, 분리된 듯 보이나, 분산된 영역이 넓어 영역이 겹쳐 있는 것을 볼 수가 있다.

T-SNE의 경우 영역들이 처음과 다른 영역을 가지기도 했지만, PCA는 영역 분리도 제대로 되어있지 않으면서 시계반대방향으로 회전하는 듯한 양상을 보인다.