a2

October 29, 2021

1 PART II: Implementation TASK

importing necessary library for the assignment

[ ]:
```
import torch
import torchvision
import torchvision.transforms as transforms
import torch.nn as nn
import torch.nn.functional as F
import matplotlib.pyplot as plt
import numpy as np
import torch.optim as optim
import time
```

2 Task1 - Data Preparation

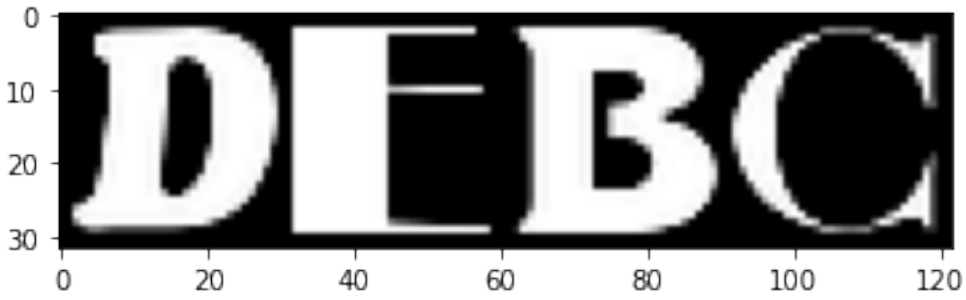
[ ]:
```
dataset_local = torchvision.datasets.ImageFolder('notMNIST_small',
                                                  transform = transforms.Compose(
                                                                                 ␣
 ↪[transforms.ToTensor(),
                                                                                  ␣
 ↪transforms.Normalize((0.5,),
                                                                                     ␣
 ↪                  (0.5,))]))
dataloader_local = torch.utils.data.DataLoader(dataset_local, batch_size=4,␣
 ↪shuffle=True, num_workers=2)
```

[ ]:
```
classes = ('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J')
```

[ ]:
```
dataiter = iter(dataloader_local)
images, labels = dataiter.next()
plt.imshow(np.transpose(torchvision.utils.make_grid(images).numpy(), (1,2,0)))
print("Ground Truth", ' '.join('%s' % classes[labels[j]] for j in range(4)))
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
```

```
Ground Truth D E B C
```



[ ]:
```
torch.manual_seed(1)
torch.cuda.manual_seed(1)
np.random.seed(1)
train = [idx for idx in torch.utils.data.RandomSampler(dataset_local)][:15000]
validation = [idx for idx in torch.utils.data.
 ↪RandomSampler(dataset_local)][15000:16000]
test = [idx for idx in torch.utils.data.RandomSampler(dataset_local)][16000:]
x_train_dataset = []
y_train_dataset = []
x_validation_dataset = []
y_validation_dataset = []
x_test_dataset = []
y_test_dataset = []

for i in train:
    if i == 1257:
        continue
    x_train_dataset.append(dataset_local[i][0].numpy())
    y_train_dataset.append(np.array([dataset_local[i][1]]))
for i in validation:
    x_validation_dataset.append(dataset_local[i][0].numpy())
    y_validation_dataset.append(np.array([dataset_local[i][1]]))
for i in test:
    x_test_dataset.append(dataset_local[i][0].numpy())
    y_test_dataset.append(np.array([dataset_local[i][1]]))
```

[ ]:
```
trainDataset = torch.utils.data.TensorDataset(torch.
 ↪FloatTensor(x_train_dataset), torch.LongTensor(y_train_dataset))
validationDatset = torch.utils.data.TensorDataset(torch.
 ↪FloatTensor(x_validation_dataset), torch.LongTensor(y_validation_dataset))
testDataset = torch.utils.data.TensorDataset(torch.FloatTensor(x_test_dataset),␣
 ↪torch.LongTensor(y_test_dataset))
```

```
trainLoader  = torch.utils.data.DataLoader(dataset = trainDataset,␣
↪batch_size=100, num_workers=1)
validationLoader  = torch.utils.data.DataLoader(dataset = validationDatset,␣
↪batch_size=100, num_workers=1)
testLoader = torch.utils.data.DataLoader(dataset = testDataset, batch_size=100,␣
↪shuffle=False, num_workers=1)
```

3 Task II - Neural Network Training (25 marks):

[ ]:
```
class SNN(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(28*28*3, 1000)
        self.fc2 = nn.Linear(1000, 10)
        self.relu = nn.ReLU()
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.softmax(x)
        return x
```

[ ]:
```
def load_model(lr):
    net = SNN()
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(net.parameters(), lr)
    return net, criterion, optimizer
```

[ ]:
```
# Using code -> from pytorchtools import EarlyStopping
# I used the code directly from the source code since I was not able to import␣
↪it
import numpy as np
import torch

class EarlyStopping:
    """Early stops the training if validation loss doesn't improve after a␣
↪given patience."""
    def __init__(self, patience=7, verbose=False, delta=0, path='checkpoint.
↪pt', trace_func=print):
        """
        Args:
```

```
            patience (int): How long to wait after last time validation loss␣
↪improved.
                            Default: 7
            verbose (bool): If True, prints a message for each validation loss␣
↪improvement.
                            Default: False
            delta (float): Minimum change in the monitored quantity to qualify␣
↪as an improvement.
                            Default: 0
            path (str): Path for the checkpoint to be saved to.
                            Default: 'checkpoint.pt'
            trace_func (function): trace print function.
                            Default: print
        """
        self.patience = patience
        self.verbose = verbose
        self.counter = 0
        self.best_score = None
        self.early_stop = False
        self.val_loss_min = np.Inf
        self.delta = delta
        self.path = path
        self.trace_func = trace_func
    def __call__(self, val_loss, model):

        score = -val_loss

        if self.best_score is None:
            self.best_score = score
            self.save_checkpoint(val_loss, model)
        elif score < self.best_score + self.delta:
            self.counter += 1
            self.trace_func(f'EarlyStopping counter: {self.counter} out of␣
↪{self.patience}')
            if self.counter >= self.patience:
                self.early_stop = True
        else:
            self.best_score = score
            self.save_checkpoint(val_loss, model)
            self.counter = 0

    def save_checkpoint(self, val_loss, model):
        '''Saves model when validation loss decrease.'''
        if self.verbose:
            self.trace_func(f'Validation loss decreased ({self.val_loss_min:.
↪6f} --> {val_loss:.6f}).  Saving model ...')
        torch.save(model.state_dict(), self.path)
```

```
        self.val_loss_min = val_loss
```

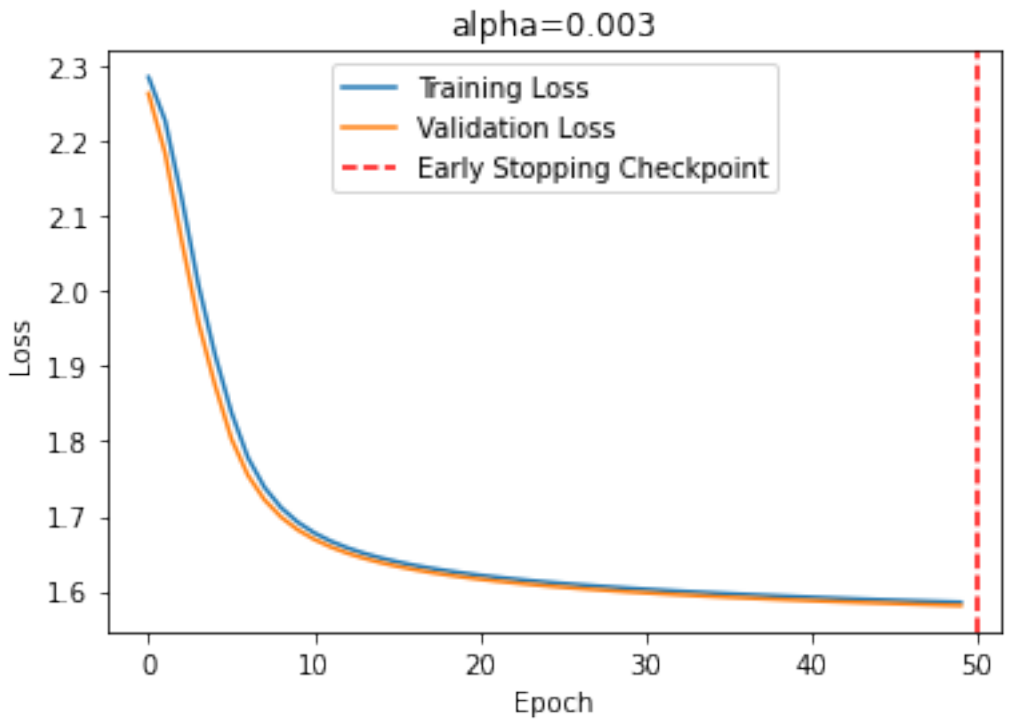
[ ]:
```
## When alpha = 0.003
alpha = 0.003
epochs = 50
seed = 0
model, criterion, optimizer = load_model(alpha)
lossRec = []
varlossRec = []
nRec = []
timeRec = []

torch.manual_seed(seed)

for e in range(epochs):
    running_loss = 0
    v_running_loss = 0
    batch_cnt = 0
    v_batch_cnt = 0

    early_stopping = EarlyStopping(patience = 7, verbose = True)

    for images, labels in trainLoader:
        # Flatten MNIST images into a 784 long vector
        images = images.view(images.shape[0], -1)
        batch_cnt+=1
        # Training pass
        optimizer.zero_grad()
        output = model(images)
        loss = criterion(output.squeeze(), labels.squeeze())
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    else:
        print(f"Training loss: {running_loss/len(trainLoader)}")

    with torch.no_grad():
        for v_image, v_labels in validationLoader:
            v_image = v_image.view(v_image.shape[0], -1)
            v_batch_cnt += 1
            optimizer.zero_grad()
            outputs_v = model(v_image)
            loss_v = criterion(outputs_v.squeeze(), v_labels.squeeze())
            v_running_loss += loss_v.item()
```

```
    nRec.append(e)
    lossRec.append(running_loss/batch_cnt)
    varlossRec.append(v_running_loss/v_batch_cnt)
    timeRec.append(time.time())

    early_stopping(v_running_loss/v_batch_cnt, model)
    if early_stopping.early_stop:
            print("Early stopping")
            break

plt.plot(nRec, lossRec, label="Training Loss")
plt.plot(nRec, varlossRec, label="Validation Loss")

minposs = varlossRec.index(min(varlossRec))+1
plt.axvline(minposs, linestyle='--', color='r',label='Early Stopping␣
 ↪Checkpoint')

plt.title("alpha=0.003")
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
print(lossRec[-1]) #Train loss
print(varlossRec[-1]) #validation loss
```

```
Training loss: 2.2855285580952964
Validation loss decreased (inf --> 2.263526).  Saving model …
Training loss: 2.2278530104955037
Validation loss decreased (inf --> 2.184058).  Saving model …
Training loss: 2.12489261786143
Validation loss decreased (inf --> 2.067452).  Saving model …
Training loss: 2.009773952166239
Validation loss decreased (inf --> 1.959527).  Saving model …
Training loss: 1.9164653754234313
Validation loss decreased (inf --> 1.875700).  Saving model …
Training loss: 1.8385619060198466
Validation loss decreased (inf --> 1.803307).  Saving model …
Training loss: 1.7781311901410422
Validation loss decreased (inf --> 1.754516).  Saving model …
Training loss: 1.7384886900583902
Validation loss decreased (inf --> 1.721859).  Saving model …
Training loss: 1.7115354204177857
Validation loss decreased (inf --> 1.698922).  Saving model …
Training loss: 1.6922832496960958
Validation loss decreased (inf --> 1.682085).  Saving model …
Training loss: 1.677914814154307
Validation loss decreased (inf --> 1.669243).  Saving model …
```

```
Training loss: 1.6667866055170695
Validation loss decreased (inf --> 1.659125).  Saving model …
Training loss: 1.657895300388336
Validation loss decreased (inf --> 1.650926).  Saving model …
Training loss: 1.650606750647227
Validation loss decreased (inf --> 1.644134).  Saving model …
Training loss: 1.6445040202140808
Validation loss decreased (inf --> 1.638394).  Saving model …
Training loss: 1.6393027885754903
Validation loss decreased (inf --> 1.633463).  Saving model …
Training loss: 1.634802455107371
Validation loss decreased (inf --> 1.629168).  Saving model …
Training loss: 1.630857207775116
Validation loss decreased (inf --> 1.625384).  Saving model …
Training loss: 1.627360234260559
Validation loss decreased (inf --> 1.622017).  Saving model …
Training loss: 1.624231179555257
Validation loss decreased (inf --> 1.618994).  Saving model …
Training loss: 1.621407778263092
Validation loss decreased (inf --> 1.616256).  Saving model …
Training loss: 1.6188414812088012
Validation loss decreased (inf --> 1.613759).  Saving model …
Training loss: 1.6164931591351828
Validation loss decreased (inf --> 1.611468).  Saving model …
Training loss: 1.6143316904703775
Validation loss decreased (inf --> 1.609355).  Saving model …
Training loss: 1.612331748008728
Validation loss decreased (inf --> 1.607397).  Saving model …
Training loss: 1.6104720441500346
Validation loss decreased (inf --> 1.605574).  Saving model …
Training loss: 1.6087352697054544
Validation loss decreased (inf --> 1.603870).  Saving model …
Training loss: 1.6071069351832072
Validation loss decreased (inf --> 1.602274).  Saving model …
Training loss: 1.6055754113197327
Validation loss decreased (inf --> 1.600773).  Saving model …
Training loss: 1.6041300996144612
Validation loss decreased (inf --> 1.599356).  Saving model …
Training loss: 1.6027620021502178
Validation loss decreased (inf --> 1.598016).  Saving model …
Training loss: 1.6014636929829915
Validation loss decreased (inf --> 1.596745).  Saving model …
Training loss: 1.600228922367096
Validation loss decreased (inf --> 1.595535).  Saving model …
Training loss: 1.599052050113678
Validation loss decreased (inf --> 1.594384).  Saving model …
Training loss: 1.5979279867808025
Validation loss decreased (inf --> 1.593285).  Saving model …
```

```
Training loss: 1.5968520991007487
Validation loss decreased (inf --> 1.592235).  Saving model …
Training loss: 1.5958208362261455
Validation loss decreased (inf --> 1.591229).  Saving model …
Training loss: 1.5948311217625937
Validation loss decreased (inf --> 1.590266).  Saving model …
Training loss: 1.5938796003659566
Validation loss decreased (inf --> 1.589342).  Saving model …
Training loss: 1.592963880697886
Validation loss decreased (inf --> 1.588455).  Saving model …
Training loss: 1.5920814776420593
Validation loss decreased (inf --> 1.587601).  Saving model …
Training loss: 1.5912305466334025
Validation loss decreased (inf --> 1.586779).  Saving model …
Training loss: 1.5904091715812683
Validation loss decreased (inf --> 1.585986).  Saving model …
Training loss: 1.5896154189109801
Validation loss decreased (inf --> 1.585221).  Saving model …
Training loss: 1.588847725391388
Validation loss decreased (inf --> 1.584482).  Saving model …
Training loss: 1.5881044085820515
Validation loss decreased (inf --> 1.583768).  Saving model …
Training loss: 1.5873844464619955
Validation loss decreased (inf --> 1.583077).  Saving model …
Training loss: 1.5866864442825317
Validation loss decreased (inf --> 1.582407).  Saving model …
Training loss: 1.586009221871694
Validation loss decreased (inf --> 1.581759).  Saving model …
Training loss: 1.5853518390655517
Validation loss decreased (inf --> 1.581132).  Saving model …
```




```
1.5853518390655517
1.5811322927474976
```

[ ]:
```python
#Test loss
model.eval()
test_loss = 0.0
num_test = 0.0
t_running_loss = 0.0
for t_image, t_labels in testLoader:
    t_image = t_image.view(t_image.shape[0], -1)
    outputs_t = model(t_image)
    num_test+=1
    loss_t = criterion(outputs_t.squeeze(), t_labels.squeeze())
    t_running_loss += loss_t
test_loss = t_running_loss/num_test
print('Test Loss: {:.6f}\n'.format(test_loss))
```

```
Test Loss: 1.581616
```

[ ]:
```python
## When alpha = 0.003
alpha = 0.005
epochs = 50
seed = 0
```

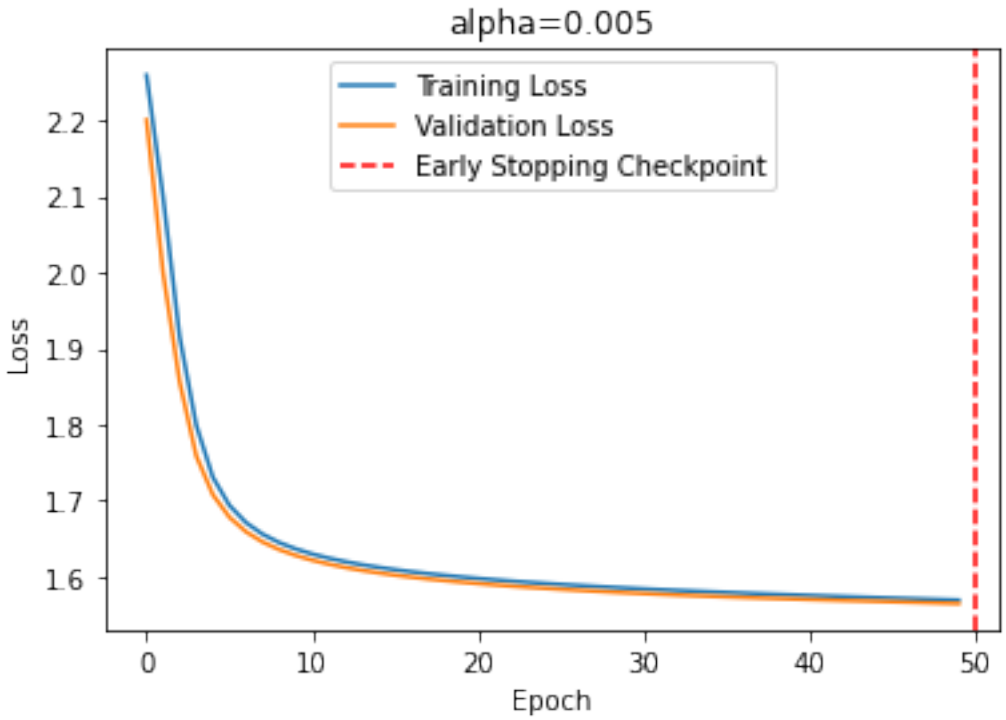
```
model, criterion, optimizer = load_model(alpha)
lossRec = []
varlossRec = []
nRec = []
timeRec = []

torch.manual_seed(seed)

for e in range(epochs):
    running_loss = 0
    v_running_loss = 0
    batch_cnt = 0
    v_batch_cnt = 0

    early_stopping = EarlyStopping(patience = 7, verbose = True)

    for images, labels in trainLoader:
        # Flatten MNIST images into a 784 long vector
        images = images.view(images.shape[0], -1)
        batch_cnt+=1
        # Training pass
        optimizer.zero_grad()
        output = model(images)
        loss = criterion(output.squeeze(), labels.squeeze())
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    else:
        print(f"Training loss: {running_loss/len(trainLoader)}")

    with torch.no_grad():
        for v_image, v_labels in validationLoader:
            v_image = v_image.view(v_image.shape[0], -1)
            v_batch_cnt += 1
            optimizer.zero_grad()
            outputs_v = model(v_image)
            loss_v = criterion(outputs_v.squeeze(), v_labels.squeeze())
            v_running_loss += loss_v.item()


    nRec.append(e)
    lossRec.append(running_loss/batch_cnt)
    varlossRec.append(v_running_loss/v_batch_cnt)
    timeRec.append(time.time())

    early_stopping(v_running_loss/v_batch_cnt, model)
```

```
    if early_stopping.early_stop:
            print("Early stopping")
            break

plt.plot(nRec, lossRec, label="Training Loss")
plt.plot(nRec, varlossRec, label="Validation Loss")

minposs = varlossRec.index(min(varlossRec))+1
plt.axvline(minposs, linestyle='--', color='r',label='Early Stopping␣
 ↪Checkpoint')

plt.title("alpha=0.005")
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
print(lossRec[-1]) #Train loss
print(varlossRec[-1]) #validation loss
```

```
Training loss: 2.2593942244847613
Validation loss decreased (inf --> 2.200590).  Saving model …
Training loss: 2.0991302824020384
Validation loss decreased (inf --> 1.999314).  Saving model …
Training loss: 1.9153385678927104
Validation loss decreased (inf --> 1.856007).  Saving model …
Training loss: 1.799416565100352
Validation loss decreased (inf --> 1.758992).  Saving model …
Training loss: 1.7310233775774637
Validation loss decreased (inf --> 1.708022).  Saving model …
Training loss: 1.6939350183804829
Validation loss decreased (inf --> 1.678416).  Saving model …
Training loss: 1.6713258639971416
Validation loss decreased (inf --> 1.659279).  Saving model …
Training loss: 1.6561205768585205
Validation loss decreased (inf --> 1.645873).  Saving model …
Training loss: 1.6451157093048097
Validation loss decreased (inf --> 1.635904).  Saving model …
Training loss: 1.6367077922821045
Validation loss decreased (inf --> 1.628145).  Saving model …
Training loss: 1.6300171113014221
Validation loss decreased (inf --> 1.621888).  Saving model …
Training loss: 1.6245241983731589
Validation loss decreased (inf --> 1.616703).  Saving model …
Training loss: 1.6199014695485432
Validation loss decreased (inf --> 1.612308).  Saving model …
Training loss: 1.6159319027264913
Validation loss decreased (inf --> 1.608512).  Saving model …
```

```
Training loss: 1.6124662089347839
Validation loss decreased (inf --> 1.605190).  Saving model …
Training loss: 1.6093986558914184
Validation loss decreased (inf --> 1.602247).  Saving model …
Training loss: 1.6066514650980632
Validation loss decreased (inf --> 1.599610).  Saving model …
Training loss: 1.6041664878527324
Validation loss decreased (inf --> 1.597228).  Saving model …
Training loss: 1.6019003574053448
Validation loss decreased (inf --> 1.595059).  Saving model …
Training loss: 1.599819401105245
Validation loss decreased (inf --> 1.593071).  Saving model …
Training loss: 1.5978965767224629
Validation loss decreased (inf --> 1.591235).  Saving model …
Training loss: 1.5961107285817464
Validation loss decreased (inf --> 1.589529).  Saving model …
Training loss: 1.5944445276260375
Validation loss decreased (inf --> 1.587939).  Saving model …
Training loss: 1.5928844785690308
Validation loss decreased (inf --> 1.586450).  Saving model …
Training loss: 1.5914190117518108
Validation loss decreased (inf --> 1.585051).  Saving model …
Training loss: 1.5900378926595051
Validation loss decreased (inf --> 1.583731).  Saving model …
Training loss: 1.5887329292297363
Validation loss decreased (inf --> 1.582483).  Saving model …
Training loss: 1.5874965874354046
Validation loss decreased (inf --> 1.581300).  Saving model …
Training loss: 1.5863233009974163
Validation loss decreased (inf --> 1.580178).  Saving model …
Training loss: 1.5852076347668966
Validation loss decreased (inf --> 1.579113).  Saving model …
Training loss: 1.5841442330678304
Validation loss decreased (inf --> 1.578102).  Saving model …
Training loss: 1.5831276178359985
Validation loss decreased (inf --> 1.577141).  Saving model …
Training loss: 1.5821540466944377
Validation loss decreased (inf --> 1.576228).  Saving model …
Training loss: 1.5812199465433756
Validation loss decreased (inf --> 1.575360).  Saving model …
Training loss: 1.5803218905131022
Validation loss decreased (inf --> 1.574535).  Saving model …
Training loss: 1.5794566853841145
Validation loss decreased (inf --> 1.573751).  Saving model …
Training loss: 1.5786209750175475
Validation loss decreased (inf --> 1.573004).  Saving model …
Training loss: 1.577811975479126
Validation loss decreased (inf --> 1.572291).  Saving model …
```

```
Training loss: 1.5770275298754375
Validation loss decreased (inf --> 1.571608).  Saving model …
Training loss: 1.5762648582458496
Validation loss decreased (inf --> 1.570952).  Saving model …
Training loss: 1.5755215525627135
Validation loss decreased (inf --> 1.570320).  Saving model …
Training loss: 1.574795282681783
Validation loss decreased (inf --> 1.569708).  Saving model …
Training loss: 1.57408438205719
Validation loss decreased (inf --> 1.569112).  Saving model …
Training loss: 1.573387173016866
Validation loss decreased (inf --> 1.568525).  Saving model …
Training loss: 1.5727023267745972
Validation loss decreased (inf --> 1.567944).  Saving model …
Training loss: 1.5720297265052796
Validation loss decreased (inf --> 1.567364).  Saving model …
Training loss: 1.5713696670532227
Validation loss decreased (inf --> 1.566782).  Saving model …
Training loss: 1.5707235256830852
Validation loss decreased (inf --> 1.566194).  Saving model …
Training loss: 1.5700927329063417
Validation loss decreased (inf --> 1.565602).  Saving model …
Training loss: 1.5694786429405212
Validation loss decreased (inf --> 1.565010).  Saving model …
```



```
1.5694786429405212
1.56500985622406
```

[ ]:
```python
#Test loss
model.eval()
test_loss = 0.0
num_test = 0.0
t_running_loss = 0.0
for t_image, t_labels in testLoader:
    t_image = t_image.view(t_image.shape[0], -1)
    outputs_t = model(t_image)
    num_test+=1
    loss_t = criterion(outputs_t.squeeze(), t_labels.squeeze())
    t_running_loss += loss_t
test_loss = t_running_loss/num_test
print('Test Loss: {:.6f}\n'.format(test_loss))
```

```
Test Loss: 1.568158
```

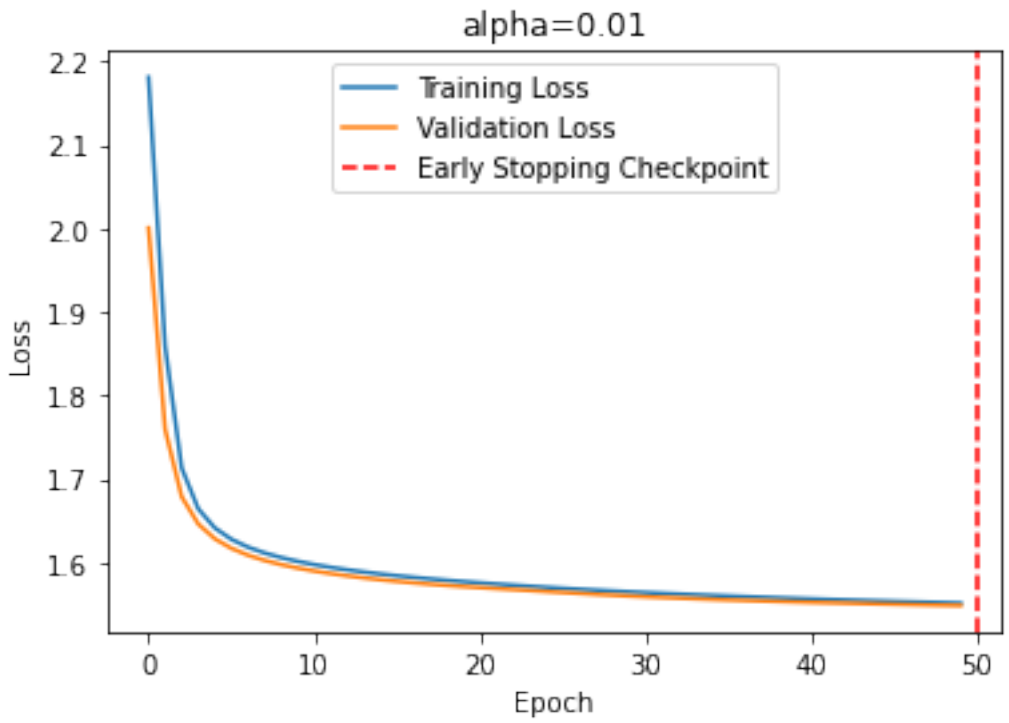
[ ]:
```python
## When alpha = 0.01
alpha = 0.01
epochs = 50
seed = 0
model, criterion, optimizer = load_model(alpha)
lossRec = []
varlossRec = []
nRec = []
timeRec = []

torch.manual_seed(seed)

for e in range(epochs):
    running_loss = 0
    v_running_loss = 0
    batch_cnt = 0
    v_batch_cnt = 0

    early_stopping = EarlyStopping(patience = 7, verbose = True)

    for images, labels in trainLoader:
        # Flatten MNIST images into a 784 long vector
        images = images.view(images.shape[0], -1)
        batch_cnt+=1
        # Training pass
        optimizer.zero_grad()
        output = model(images)
```

```
        loss = criterion(output.squeeze(), labels.squeeze())
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    else:
        print(f"Training loss: {running_loss/len(trainLoader)}")

    with torch.no_grad():
        for v_image, v_labels in validationLoader:
            v_image = v_image.view(v_image.shape[0], -1)
            v_batch_cnt += 1
            optimizer.zero_grad()
            outputs_v = model(v_image)
            loss_v = criterion(outputs_v.squeeze(), v_labels.squeeze())
            v_running_loss += loss_v.item()


    nRec.append(e)
    lossRec.append(running_loss/batch_cnt)
    varlossRec.append(v_running_loss/v_batch_cnt)
    timeRec.append(time.time())

    early_stopping(v_running_loss/v_batch_cnt, model)
    if early_stopping.early_stop:
            print("Early stopping")
            break

plt.plot(nRec, lossRec, label="Training Loss")
plt.plot(nRec, varlossRec, label="Validation Loss")

minposs = varlossRec.index(min(varlossRec))+1
plt.axvline(minposs, linestyle='--', color='r',label='Early Stopping␣
↪Checkpoint')

plt.title("alpha=0.01")
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
print(lossRec[-1]) #Train loss
print(varlossRec[-1]) #validation loss
```

```
Training loss: 2.181005589167277
Validation loss decreased (inf --> 2.000566).  Saving model …
Training loss: 1.8587437844276429
Validation loss decreased (inf --> 1.759259).  Saving model …
```

```
Training loss: 1.7130081168810527
Validation loss decreased (inf --> 1.678421).  Saving model …
Training loss: 1.6640508755048116
Validation loss decreased (inf --> 1.645865).  Saving model …
Training loss: 1.641198693116506
Validation loss decreased (inf --> 1.628138).  Saving model …
Training loss: 1.627540086110433
Validation loss decreased (inf --> 1.616702).  Saving model …
Training loss: 1.6181757609049479
Validation loss decreased (inf --> 1.608512).  Saving model …
Training loss: 1.6111844277381897
Validation loss decreased (inf --> 1.602245).  Saving model …
Training loss: 1.605657029946645
Validation loss decreased (inf --> 1.597230).  Saving model …
Training loss: 1.6011062335968018
Validation loss decreased (inf --> 1.593073).  Saving model …
Training loss: 1.5972486567497253
Validation loss decreased (inf --> 1.589536).  Saving model …
Training loss: 1.5939077130953472
Validation loss decreased (inf --> 1.586462).  Saving model …
Training loss: 1.5909696420033772
Validation loss decreased (inf --> 1.583749).  Saving model …
Training loss: 1.5883525649706522
Validation loss decreased (inf --> 1.581322).  Saving model …
Training loss: 1.585999864737193
Validation loss decreased (inf --> 1.579139).  Saving model …
Training loss: 1.5838667035102845
Validation loss decreased (inf --> 1.577169).  Saving model …
Training loss: 1.581914583047231
Validation loss decreased (inf --> 1.575389).  Saving model …
Training loss: 1.5801136962572733
Validation loss decreased (inf --> 1.573781).  Saving model …
Training loss: 1.5784387302398681
Validation loss decreased (inf --> 1.572321).  Saving model …
Training loss: 1.5768662484486897
Validation loss decreased (inf --> 1.570983).  Saving model …
Training loss: 1.5753769103686015
Validation loss decreased (inf --> 1.569741).  Saving model …
Training loss: 1.5739537866910298
Validation loss decreased (inf --> 1.568558).  Saving model …
Training loss: 1.5725838979085287
Validation loss decreased (inf --> 1.567396).  Saving model …
Training loss: 1.5712649218241375
Validation loss decreased (inf --> 1.566227).  Saving model …
Training loss: 1.5700044616063435
Validation loss decreased (inf --> 1.565046).  Saving model …
Training loss: 1.5688109230995178
Validation loss decreased (inf --> 1.563881).  Saving model …
```

```
Training loss: 1.5676837952931721
Validation loss decreased (inf --> 1.562754).  Saving model …
Training loss: 1.566615449587504
Validation loss decreased (inf --> 1.561675).  Saving model …
Training loss: 1.5655982184410095
Validation loss decreased (inf --> 1.560651).  Saving model …
Training loss: 1.5646256136894225
Validation loss decreased (inf --> 1.559685).  Saving model …
Training loss: 1.5636935830116272
Validation loss decreased (inf --> 1.558775).  Saving model …
Training loss: 1.5627996182441712
Validation loss decreased (inf --> 1.557917).  Saving model …
Training loss: 1.5619419805208843
Validation loss decreased (inf --> 1.557112).  Saving model …
Training loss: 1.5611180861790974
Validation loss decreased (inf --> 1.556356).  Saving model …
Training loss: 1.560326073964437
Validation loss decreased (inf --> 1.555645).  Saving model …
Training loss: 1.5595643329620361
Validation loss decreased (inf --> 1.554976).  Saving model …
Training loss: 1.5588306546211244
Validation loss decreased (inf --> 1.554343).  Saving model …
Training loss: 1.5581225593884787
Validation loss decreased (inf --> 1.553743).  Saving model …
Training loss: 1.5574381295839945
Validation loss decreased (inf --> 1.553173).  Saving model …
Training loss: 1.5567758385340373
Validation loss decreased (inf --> 1.552632).  Saving model …
Training loss: 1.5561342755953471
Validation loss decreased (inf --> 1.552114).  Saving model …
Training loss: 1.5555115421613057
Validation loss decreased (inf --> 1.551618).  Saving model …
Training loss: 1.5549063595136006
Validation loss decreased (inf --> 1.551143).  Saving model …
Training loss: 1.5543177858988444
Validation loss decreased (inf --> 1.550688).  Saving model …
Training loss: 1.553745137055715
Validation loss decreased (inf --> 1.550251).  Saving model …
Training loss: 1.5531867408752442
Validation loss decreased (inf --> 1.549828).  Saving model …
Training loss: 1.5526422580083212
Validation loss decreased (inf --> 1.549422).  Saving model …
Training loss: 1.5521109668413797
Validation loss decreased (inf --> 1.549030).  Saving model …
Training loss: 1.5515920042991638
Validation loss decreased (inf --> 1.548652).  Saving model …
Training loss: 1.5510855555534362
Validation loss decreased (inf --> 1.548287).  Saving model …
```




```
1.5510855555534362
1.5482874751091003
```

[ ]:
```python
#Test loss
model.eval()
test_loss = 0.0
num_test = 0.0
t_running_loss = 0.0
for t_image, t_labels in testLoader:
    t_image = t_image.view(t_image.shape[0], -1)
    outputs_t = model(t_image)
    num_test+=1
    loss_t = criterion(outputs_t.squeeze(), t_labels.squeeze())
    t_running_loss += loss_t
test_loss = t_running_loss/num_test
print('Test Loss: {:.6f}\n'.format(test_loss))
```

```
Test Loss: 1.552814
```

[ ]:
```python
## When alpha = 0.05
alpha = 0.05
epochs = 50
seed = 0
```

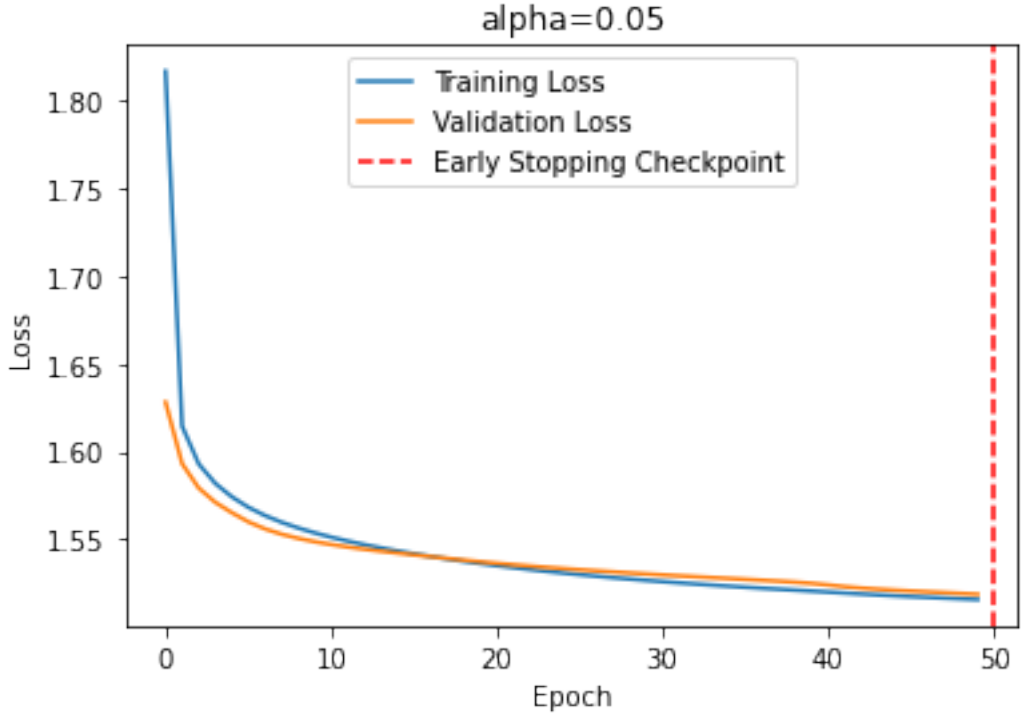
```
model, criterion, optimizer = load_model(alpha)
import time
lossRec = []
varlossRec = []
nRec = []
timeRec = []

torch.manual_seed(seed)

for e in range(epochs):
    running_loss = 0
    v_running_loss = 0
    batch_cnt = 0
    v_batch_cnt = 0

    early_stopping = EarlyStopping(patience = 7, verbose = True)

    for images, labels in trainLoader:
        # Flatten MNIST images into a 784 long vector
        images = images.view(images.shape[0], -1)
        batch_cnt+=1
        # Training pass
        optimizer.zero_grad()
        output = model(images)
        loss = criterion(output.squeeze(), labels.squeeze())
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    else:
        print(f"Training loss: {running_loss/len(trainLoader)}")

    with torch.no_grad():
        for v_image, v_labels in validationLoader:
            v_image = v_image.view(v_image.shape[0], -1)
            v_batch_cnt += 1
            optimizer.zero_grad()
            outputs_v = model(v_image)
            loss_v = criterion(outputs_v.squeeze(), v_labels.squeeze())
            v_running_loss += loss_v.item()


    nRec.append(e)
    lossRec.append(running_loss/batch_cnt)
    varlossRec.append(v_running_loss/v_batch_cnt)
    timeRec.append(time.time())
```

```
    early_stopping(v_running_loss/v_batch_cnt, model)
    if early_stopping.early_stop:
            print("Early stopping")
            break

plt.plot(nRec, lossRec, label="Training Loss")
plt.plot(nRec, varlossRec, label="Validation Loss")

minposs = varlossRec.index(min(varlossRec))+1
plt.axvline(minposs, linestyle='--', color='r',label='Early Stopping␣
 ↪Checkpoint')

plt.title("alpha=0.05")
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
print(lossRec[-1]) #Train loss
print(varlossRec[-1]) #validation loss
```

```
Training loss: 1.8168927772839865
Validation loss decreased (inf --> 1.628187).  Saving model …
Training loss: 1.614474957784017
Validation loss decreased (inf --> 1.593067).  Saving model …
Training loss: 1.5929615799585979
Validation loss decreased (inf --> 1.579185).  Saving model …
Training loss: 1.5818282636006673
Validation loss decreased (inf --> 1.571068).  Saving model …
Training loss: 1.5741884430249533
Validation loss decreased (inf --> 1.565138).  Saving model …
Training loss: 1.5682002393404644
Validation loss decreased (inf --> 1.559905).  Saving model …
Training loss: 1.563508656024933
Validation loss decreased (inf --> 1.555887).  Saving model …
Training loss: 1.5596681722005208
Validation loss decreased (inf --> 1.552860).  Saving model …
Training loss: 1.5564340662956238
Validation loss decreased (inf --> 1.550506).  Saving model …
Training loss: 1.5536346777280172
Validation loss decreased (inf --> 1.548583).  Saving model …
Training loss: 1.5511517961819967
Validation loss decreased (inf --> 1.546971).  Saving model …
Training loss: 1.5489193852742513
Validation loss decreased (inf --> 1.545565).  Saving model …
Training loss: 1.5468962009747822
Validation loss decreased (inf --> 1.544314).  Saving model …
Training loss: 1.5450384203592937
```

```
Validation loss decreased (inf --> 1.543187).  Saving model …
Training loss: 1.5433140484491985
Validation loss decreased (inf --> 1.542114).  Saving model …
Training loss: 1.5416998561223347
Validation loss decreased (inf --> 1.541078).  Saving model …
Training loss: 1.540180638631185
Validation loss decreased (inf --> 1.540059).  Saving model …
Training loss: 1.5387486592928568
Validation loss decreased (inf --> 1.539059).  Saving model …
Training loss: 1.537397419611613
Validation loss decreased (inf --> 1.538077).  Saving model …
Training loss: 1.5361259571711223
Validation loss decreased (inf --> 1.537131).  Saving model …
Training loss: 1.534918664296468
Validation loss decreased (inf --> 1.536252).  Saving model …
Training loss: 1.533778239885966
Validation loss decreased (inf --> 1.535426).  Saving model …
Training loss: 1.532701907157898
Validation loss decreased (inf --> 1.534659).  Saving model …
Training loss: 1.5316869942347209
Validation loss decreased (inf --> 1.533935).  Saving model …
Training loss: 1.5307235368092855
Validation loss decreased (inf --> 1.533256).  Saving model …
Training loss: 1.529804809888204
Validation loss decreased (inf --> 1.532593).  Saving model …
Training loss: 1.528929316997528
Validation loss decreased (inf --> 1.531962).  Saving model …
Training loss: 1.5280940334002178
Validation loss decreased (inf --> 1.531346).  Saving model …
Training loss: 1.5273018892606098
Validation loss decreased (inf --> 1.530751).  Saving model …
Training loss: 1.526545873483022
Validation loss decreased (inf --> 1.530188).  Saving model …
Training loss: 1.525824733575185
Validation loss decreased (inf --> 1.529637).  Saving model …
Training loss: 1.5251323866844178
Validation loss decreased (inf --> 1.529114).  Saving model …
Training loss: 1.524463509718577
Validation loss decreased (inf --> 1.528605).  Saving model …
Training loss: 1.5238206307093303
Validation loss decreased (inf --> 1.528103).  Saving model …
Training loss: 1.5232050903638203
Validation loss decreased (inf --> 1.527606).  Saving model …
Training loss: 1.522617613474528
Validation loss decreased (inf --> 1.527101).  Saving model …
Training loss: 1.5220540388425192
Validation loss decreased (inf --> 1.526587).  Saving model …
Training loss: 1.5215086189905802
```

```
Validation loss decreased (inf --> 1.526063).  Saving model …
Training loss: 1.520972131093343
Validation loss decreased (inf --> 1.525492).  Saving model …
Training loss: 1.5204256065686543
Validation loss decreased (inf --> 1.524804).  Saving model …
Training loss: 1.519854416847229
Validation loss decreased (inf --> 1.523920).  Saving model …
Training loss: 1.5192923561731975
Validation loss decreased (inf --> 1.522907).  Saving model …
Training loss: 1.5187456838289897
Validation loss decreased (inf --> 1.522117).  Saving model …
Training loss: 1.518229948679606
Validation loss decreased (inf --> 1.521482).  Saving model …
Training loss: 1.5177484631538392
Validation loss decreased (inf --> 1.520928).  Saving model …
Training loss: 1.5172919241587322
Validation loss decreased (inf --> 1.520437).  Saving model …
Training loss: 1.5168589997291564
Validation loss decreased (inf --> 1.519974).  Saving model …
Training loss: 1.5164399282137553
Validation loss decreased (inf --> 1.519526).  Saving model …
Training loss: 1.5160382437705993
Validation loss decreased (inf --> 1.519055).  Saving model …
Training loss: 1.5156447474161785
Validation loss decreased (inf --> 1.518591).  Saving model …
```



```
1.5156447474161785
1.5185910940170289
```

[ ]:
```python
#Test loss
model.eval()
test_loss = 0.0
num_test = 0.0
t_running_loss = 0.0
for t_image, t_labels in testLoader:
    t_image = t_image.view(t_image.shape[0], -1)
    outputs_t = model(t_image)
    num_test+=1
    loss_t = criterion(outputs_t.squeeze(), t_labels.squeeze())
    t_running_loss += loss_t
test_loss = t_running_loss/num_test
print('Test Loss: {:.6f}\n'.format(test_loss))
```

```
Test Loss: 1.518972
```

[ ]:
```python
## When alpha = 0.1
alpha = 0.1
epochs = 50
seed = 0
model, criterion, optimizer = load_model(alpha)
import time
lossRec = []
varlossRec = []
nRec = []
timeRec = []

torch.manual_seed(seed)

for e in range(epochs):
    running_loss = 0
    v_running_loss = 0
    batch_cnt = 0
    v_batch_cnt = 0

    early_stopping = EarlyStopping(patience = 7, verbose = True)

    for images, labels in trainLoader:
        # Flatten MNIST images into a 784 long vector
        images = images.view(images.shape[0], -1)
        batch_cnt+=1
        # Training pass
```

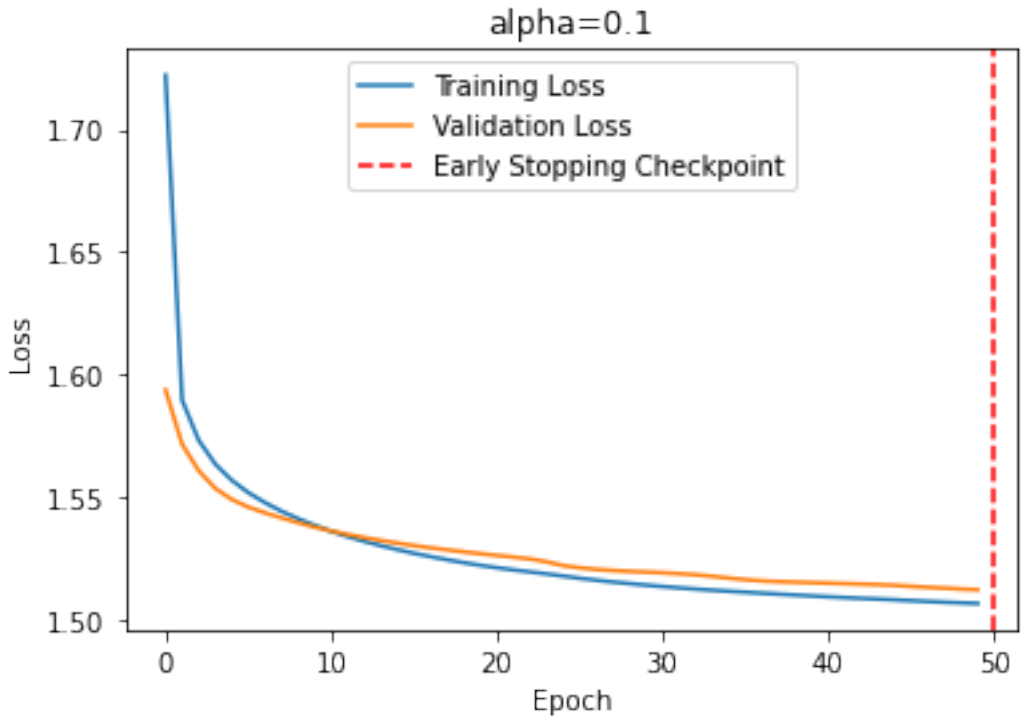
```
        optimizer.zero_grad()
        output = model(images)
        loss = criterion(output.squeeze(), labels.squeeze())
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    else:
        print(f"Training loss: {running_loss/len(trainLoader)}")

    with torch.no_grad():
        for v_image, v_labels in validationLoader:
            v_image = v_image.view(v_image.shape[0], -1)
            v_batch_cnt += 1
            optimizer.zero_grad()
            outputs_v = model(v_image)
            loss_v = criterion(outputs_v.squeeze(), v_labels.squeeze())
            v_running_loss += loss_v.item()


    nRec.append(e)
    lossRec.append(running_loss/batch_cnt)
    varlossRec.append(v_running_loss/v_batch_cnt)
    timeRec.append(time.time())

    early_stopping(v_running_loss/v_batch_cnt, model)
    if early_stopping.early_stop:
            print("Early stopping")
            break

plt.plot(nRec, lossRec, label="Training Loss")
plt.plot(nRec, varlossRec, label="Validation Loss")

minposs = varlossRec.index(min(varlossRec))+1
plt.axvline(minposs, linestyle='--', color='r',label='Early Stopping␣
 ↪Checkpoint')

plt.title("alpha=0.1")
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
print(lossRec[-1]) #Train loss
print(varlossRec[-1]) #validation loss
```

```
Training loss: 1.7221400435765584
Validation loss decreased (inf --> 1.593807).  Saving model …
```

```
Training loss: 1.5896101077397664
Validation loss decreased (inf --> 1.571605).  Saving model …
Training loss: 1.5732409302393595
Validation loss decreased (inf --> 1.560838).  Saving model …
Training loss: 1.5635332767168681
Validation loss decreased (inf --> 1.553644).  Saving model …
Training loss: 1.5568880899747213
Validation loss decreased (inf --> 1.549061).  Saving model …
Training loss: 1.5518608776728313
Validation loss decreased (inf --> 1.545965).  Saving model …
Training loss: 1.5477789211273194
Validation loss decreased (inf --> 1.543628).  Saving model …
Training loss: 1.5442955295244853
Validation loss decreased (inf --> 1.541575).  Saving model …
Training loss: 1.5412394825617473
Validation loss decreased (inf --> 1.539614).  Saving model …
Training loss: 1.5385259167353311
Validation loss decreased (inf --> 1.537801).  Saving model …
Training loss: 1.5360881694157917
Validation loss decreased (inf --> 1.536210).  Saving model …
Training loss: 1.5339239382743834
Validation loss decreased (inf --> 1.534790).  Saving model …
Training loss: 1.53198433081309
Validation loss decreased (inf --> 1.533532).  Saving model …
Training loss: 1.5302230215072632
Validation loss decreased (inf --> 1.532374).  Saving model …
Training loss: 1.528604150613149
Validation loss decreased (inf --> 1.531259).  Saving model …
Training loss: 1.5271145105361938
Validation loss decreased (inf --> 1.530265).  Saving model …
Training loss: 1.5257503040631613
Validation loss decreased (inf --> 1.529359).  Saving model …
Training loss: 1.5244804040590922
Validation loss decreased (inf --> 1.528544).  Saving model …
Training loss: 1.5233048550287882
Validation loss decreased (inf --> 1.527727).  Saving model …
Training loss: 1.5222177219390869
Validation loss decreased (inf --> 1.526965).  Saving model …
Training loss: 1.521235253016154
Validation loss decreased (inf --> 1.526252).  Saving model …
Training loss: 1.5203481602668762
Validation loss decreased (inf --> 1.525579).  Saving model …
Training loss: 1.5195281505584717
Validation loss decreased (inf --> 1.524835).  Saving model …
Training loss: 1.5187170513470967
Validation loss decreased (inf --> 1.523648).  Saving model …
Training loss: 1.5178307835261027
Validation loss decreased (inf --> 1.522112).  Saving model …
```

```
Training loss: 1.5169382588068645
Validation loss decreased (inf --> 1.521206).  Saving model …
Training loss: 1.5161503124237061
Validation loss decreased (inf --> 1.520559).  Saving model …
Training loss: 1.5154327758153279
Validation loss decreased (inf --> 1.520119).  Saving model …
Training loss: 1.5147717102368672
Validation loss decreased (inf --> 1.519767).  Saving model …
Training loss: 1.514150755405426
Validation loss decreased (inf --> 1.519468).  Saving model …
Training loss: 1.5135683838526408
Validation loss decreased (inf --> 1.519179).  Saving model …
Training loss: 1.5130305083592732
Validation loss decreased (inf --> 1.518812).  Saving model …
Training loss: 1.5125384060541789
Validation loss decreased (inf --> 1.518351).  Saving model …
Training loss: 1.5120813433329265
Validation loss decreased (inf --> 1.517732).  Saving model …
Training loss: 1.5116573150952657
Validation loss decreased (inf --> 1.517016).  Saving model …
Training loss: 1.511251548131307
Validation loss decreased (inf --> 1.516351).  Saving model …
Training loss: 1.5108523591359457
Validation loss decreased (inf --> 1.515885).  Saving model …
Training loss: 1.5104638965924582
Validation loss decreased (inf --> 1.515572).  Saving model …
Training loss: 1.5100929737091064
Validation loss decreased (inf --> 1.515357).  Saving model …
Training loss: 1.5097327224413555
Validation loss decreased (inf --> 1.515153).  Saving model …
Training loss: 1.5093882314364115
Validation loss decreased (inf --> 1.514948).  Saving model …
Training loss: 1.5090588720639546
Validation loss decreased (inf --> 1.514747).  Saving model …
Training loss: 1.5087484788894654
Validation loss decreased (inf --> 1.514570).  Saving model …
Training loss: 1.5084552717208863
Validation loss decreased (inf --> 1.514335).  Saving model …
Training loss: 1.5081575242678324
Validation loss decreased (inf --> 1.514059).  Saving model …
Training loss: 1.5078200419743857
Validation loss decreased (inf --> 1.513636).  Saving model …
Training loss: 1.5074906412760416
Validation loss decreased (inf --> 1.513245).  Saving model …
Training loss: 1.50719012260437
Validation loss decreased (inf --> 1.512936).  Saving model …
Training loss: 1.5069112865130108
Validation loss decreased (inf --> 1.512541).  Saving model …
```

```
Training loss: 1.5066682839393615
Validation loss decreased (inf --> 1.512292).  Saving model …
```




```
1.5066682839393615
1.5122920513153075
```

[ ]:
```python
#Test loss
model.eval()
test_loss = 0.0
num_test = 0.0
t_running_loss = 0.0
for t_image, t_labels in testLoader:
    t_image = t_image.view(t_image.shape[0], -1)
    outputs_t = model(t_image)
    num_test+=1
    loss_t = criterion(outputs_t.squeeze(), t_labels.squeeze())
    t_running_loss += loss_t
test_loss = t_running_loss/num_test
print('Test Loss: {:.6f}\n'.format(test_loss))
```

```
Test Loss: 1.510421
```

Obervation: the data is less overfitted starting when learning rate is around 0.05 and also data reaches the minimum loss quicker with the learning rate around this value.

4 Task III - Testing the Number of Hidden Units (25 marks):

[ ]:
```
class SNN2(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(28*28*3, 100)
        self.fc2 = nn.Linear(100, 10)
        self.relu = nn.ReLU()
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.softmax(x)
        return x

class SNN3(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(28*28*3, 500)
        self.fc2 = nn.Linear(500, 10)
        self.relu = nn.ReLU()
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.softmax(x)
        return x

class SNN4(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(28*28*3, 1000)
        self.fc2 = nn.Linear(1000, 10)
        self.relu = nn.ReLU()
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.softmax(x)
        return x
```

[ ]:
```
def load_model2(lr):
    net = SNN2()
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(net.parameters(), lr)
    return net, criterion, optimizer
def load_model3(lr):
    net = SNN3()
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(net.parameters(), lr)
    return net, criterion, optimizer
def load_model4(lr):
    net = SNN4()
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(net.parameters(), lr)
    return net, criterion, optimizer
```

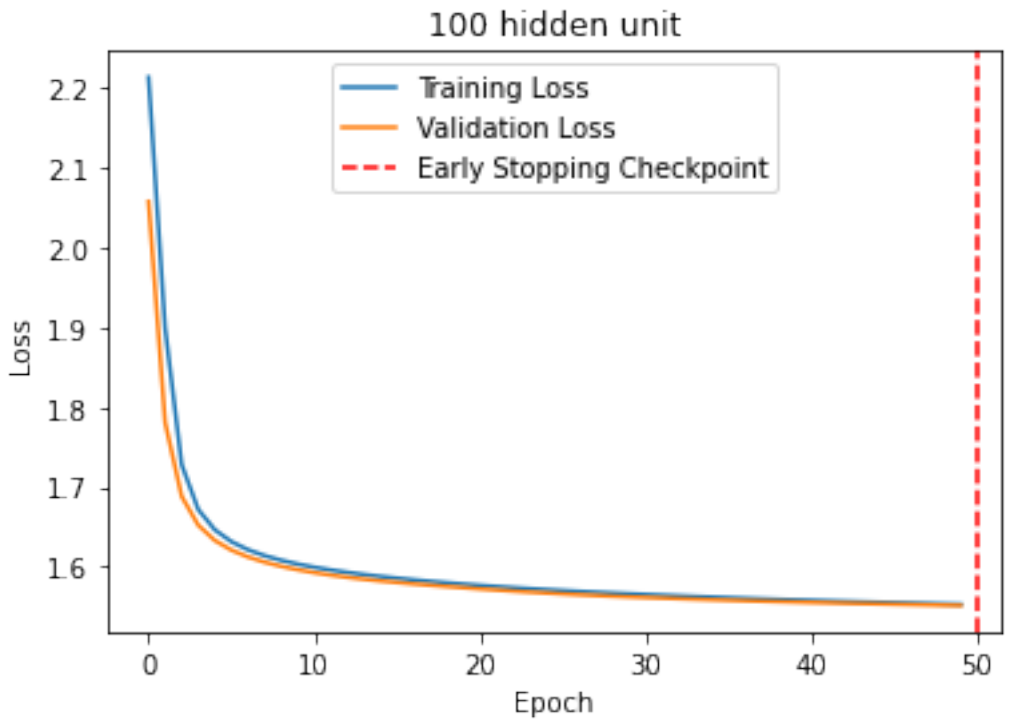
[ ]:
```
alpha = 0.01
epochs = 50
seed = 0
model2, criterion, optimizer = load_model2(alpha)
lossRec = []
varlossRec = []
nRec = []
timeRec = []

torch.manual_seed(seed)

for e in range(epochs):
    running_loss = 0
    v_running_loss = 0
    batch_cnt = 0
    v_batch_cnt = 0

    early_stopping = EarlyStopping(patience = 7, verbose = True)

    for images, labels in trainLoader:
        # Flatten MNIST images into a 784 long vector
        images = images.view(images.shape[0], -1)
        batch_cnt+=1
        # Training pass
        optimizer.zero_grad()
        output = model2(images)
        loss = criterion(output.squeeze(), labels.squeeze())
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
```

```
    else:
        print(f"Training loss: {running_loss/len(trainLoader)}")

    with torch.no_grad():
        for v_image, v_labels in validationLoader:
            v_image = v_image.view(v_image.shape[0], -1)
            v_batch_cnt += 1
            optimizer.zero_grad()
            outputs_v = model2(v_image)
            loss_v = criterion(outputs_v.squeeze(), v_labels.squeeze())
            v_running_loss += loss_v.item()


    nRec.append(e)
    lossRec.append(running_loss/batch_cnt)
    varlossRec.append(v_running_loss/v_batch_cnt)
    timeRec.append(time.time())

    early_stopping(v_running_loss/v_batch_cnt, model2)
    if early_stopping.early_stop:
            print("Early stopping")
            break

plt.plot(nRec, lossRec, label="Training Loss")
plt.plot(nRec, varlossRec, label="Validation Loss")

minposs = varlossRec.index(min(varlossRec))+1
plt.axvline(minposs, linestyle='--', color='r',label='Early Stopping␣
 ↪Checkpoint')

plt.title("100 hidden unit")
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
print(lossRec[-1]) #Train loss
print(varlossRec[-1]) #validation loss
```

```
Training loss: 2.2130413834253946
Validation loss decreased (inf --> 2.057395).  Saving model …
Training loss: 1.8992188501358032
Validation loss decreased (inf --> 1.780993).  Saving model …
Training loss: 1.7280770285924276
Validation loss decreased (inf --> 1.688225).  Saving model …
Training loss: 1.6718641344706218
Validation loss decreased (inf --> 1.652180).  Saving model …
Training loss: 1.646504645347595
```

```
Validation loss decreased (inf --> 1.633119).  Saving model …
Training loss: 1.6316192483901977
Validation loss decreased (inf --> 1.621037).  Saving model …
Training loss: 1.6215402388572693
Validation loss decreased (inf --> 1.612481).  Saving model …
Training loss: 1.6141113313039144
Validation loss decreased (inf --> 1.605974).  Saving model …
Training loss: 1.6083044743537902
Validation loss decreased (inf --> 1.600785).  Saving model …
Training loss: 1.6035695258776348
Validation loss decreased (inf --> 1.596498).  Saving model …
Training loss: 1.5995862261454263
Validation loss decreased (inf --> 1.592875).  Saving model …
Training loss: 1.5961494175593058
Validation loss decreased (inf --> 1.589756).  Saving model …
Training loss: 1.593136528333028
Validation loss decreased (inf --> 1.587039).  Saving model …
Training loss: 1.5904571843147277
Validation loss decreased (inf --> 1.584642).  Saving model …
Training loss: 1.5880402207374573
Validation loss decreased (inf --> 1.582501).  Saving model …
Training loss: 1.585837816397349
Validation loss decreased (inf --> 1.580573).  Saving model …
Training loss: 1.5838048871358235
Validation loss decreased (inf --> 1.578802).  Saving model …
Training loss: 1.5819103089968363
Validation loss decreased (inf --> 1.577147).  Saving model …
Training loss: 1.5801300628980002
Validation loss decreased (inf --> 1.575566).  Saving model …
Training loss: 1.5784535479545594
Validation loss decreased (inf --> 1.574023).  Saving model …
Training loss: 1.5768770098686218
Validation loss decreased (inf --> 1.572536).  Saving model …
Training loss: 1.575406048297882
Validation loss decreased (inf --> 1.571115).  Saving model …
Training loss: 1.5740337165196736
Validation loss decreased (inf --> 1.569795).  Saving model …
Training loss: 1.5727495137850445
Validation loss decreased (inf --> 1.568552).  Saving model …
Training loss: 1.5715416065851848
Validation loss decreased (inf --> 1.567394).  Saving model …
Training loss: 1.5703946940104168
Validation loss decreased (inf --> 1.566300).  Saving model …
Training loss: 1.5693044861157734
Validation loss decreased (inf --> 1.565271).  Saving model …
Training loss: 1.5682674471537272
Validation loss decreased (inf --> 1.564314).  Saving model …
Training loss: 1.5672759636243185
```

```
Validation loss decreased (inf --> 1.563416).  Saving model …
Training loss: 1.5663294712702434
Validation loss decreased (inf --> 1.562582).  Saving model …
Training loss: 1.5654222647349039
Validation loss decreased (inf --> 1.561788).  Saving model …
Training loss: 1.5645520480473836
Validation loss decreased (inf --> 1.561035).  Saving model …
Training loss: 1.5637174328168233
Validation loss decreased (inf --> 1.560317).  Saving model …
Training loss: 1.562914386590322
Validation loss decreased (inf --> 1.559637).  Saving model …
Training loss: 1.5621444384257
Validation loss decreased (inf --> 1.558983).  Saving model …
Training loss: 1.5614020776748658
Validation loss decreased (inf --> 1.558363).  Saving model …
Training loss: 1.5606867170333862
Validation loss decreased (inf --> 1.557767).  Saving model …
Training loss: 1.5599953103065491
Validation loss decreased (inf --> 1.557197).  Saving model …
Training loss: 1.5593279719352722
Validation loss decreased (inf --> 1.556649).  Saving model …
Training loss: 1.5586825847625732
Validation loss decreased (inf --> 1.556118).  Saving model …
Training loss: 1.5580569235483805
Validation loss decreased (inf --> 1.555608).  Saving model …
Training loss: 1.5574511726697287
Validation loss decreased (inf --> 1.555123).  Saving model …
Training loss: 1.5568639238675435
Validation loss decreased (inf --> 1.554652).  Saving model …
Training loss: 1.5562947432200114
Validation loss decreased (inf --> 1.554205).  Saving model …
Training loss: 1.555741221110026
Validation loss decreased (inf --> 1.553769).  Saving model …
Training loss: 1.5552044788996378
Validation loss decreased (inf --> 1.553343).  Saving model …
Training loss: 1.5546818256378174
Validation loss decreased (inf --> 1.552931).  Saving model …
Training loss: 1.5541717974344889
Validation loss decreased (inf --> 1.552523).  Saving model …
Training loss: 1.5536747439702352
Validation loss decreased (inf --> 1.552129).  Saving model …
Training loss: 1.5531887245178222
Validation loss decreased (inf --> 1.551740).  Saving model …
```




```
1.5531887245178222
1.5517399907112122
```

[ ]:
```
#Test loss
model2.eval()
test_loss = 0.0
num_test = 0.0
t_running_loss = 0.0
for t_image, t_labels in testLoader:
    t_image = t_image.view(t_image.shape[0], -1)
    outputs_t = model2(t_image)
    num_test+=1
    loss_t = criterion(outputs_t.squeeze(), t_labels.squeeze())
    t_running_loss += loss_t
test_loss = t_running_loss/num_test
print('Test Loss: {:.6f}\n'.format(test_loss))
```

```
Test Loss: 1.554810
```

[ ]:
```
alpha = 0.01
epochs = 50
seed = 0
model3, criterion, optimizer = load_model3(alpha)
```

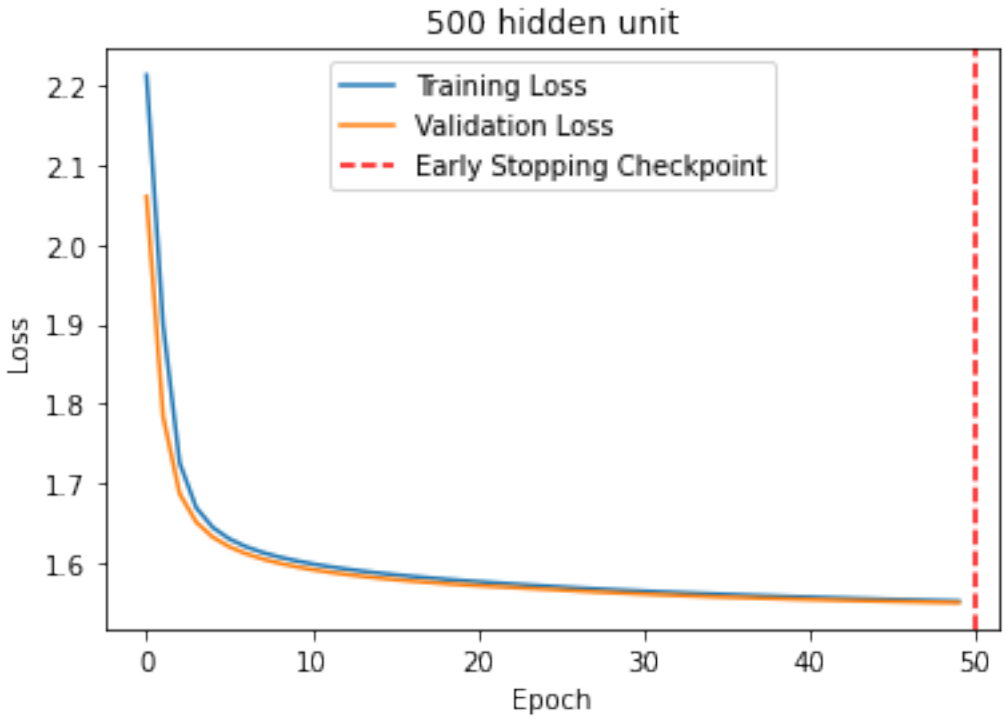
```
lossRec = []
varlossRec = []
nRec = []
timeRec = []

torch.manual_seed(seed)

for e in range(epochs):
    running_loss = 0
    v_running_loss = 0
    batch_cnt = 0
    v_batch_cnt = 0

    early_stopping = EarlyStopping(patience = 7, verbose = True)

    for images, labels in trainLoader:
        # Flatten MNIST images into a 784 long vector
        images = images.view(images.shape[0], -1)
        batch_cnt+=1
        # Training pass
        optimizer.zero_grad()
        output = model3(images)
        loss = criterion(output.squeeze(), labels.squeeze())
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    else:
        print(f"Training loss: {running_loss/len(trainLoader)}")

    with torch.no_grad():
        for v_image, v_labels in validationLoader:
            v_image = v_image.view(v_image.shape[0], -1)
            v_batch_cnt += 1
            optimizer.zero_grad()
            outputs_v = model3(v_image)
            loss_v = criterion(outputs_v.squeeze(), v_labels.squeeze())
            v_running_loss += loss_v.item()


    nRec.append(e)
    lossRec.append(running_loss/batch_cnt)
    varlossRec.append(v_running_loss/v_batch_cnt)
    timeRec.append(time.time())

    early_stopping(v_running_loss/v_batch_cnt, model3)
    if early_stopping.early_stop:
```

```
            print("Early stopping")
            break

plt.plot(nRec, lossRec, label="Training Loss")
plt.plot(nRec, varlossRec, label="Validation Loss")

minposs = varlossRec.index(min(varlossRec))+1
plt.axvline(minposs, linestyle='--', color='r',label='Early Stopping␣
 ↪Checkpoint')

plt.title("500 hidden unit")
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
print(lossRec[-1]) #Train loss
print(varlossRec[-1]) #validation loss
```

```
Training loss: 2.2130774974823
Validation loss decreased (inf --> 2.060644).  Saving model …
Training loss: 1.8993440794944763
Validation loss decreased (inf --> 1.784194).  Saving model …
Training loss: 1.7258010220527649
Validation loss decreased (inf --> 1.687022).  Saving model …
Training loss: 1.6695747979482014
Validation loss decreased (inf --> 1.651495).  Saving model …
Training loss: 1.6448125187555949
Validation loss decreased (inf --> 1.632552).  Saving model …
Training loss: 1.6303546659151713
Validation loss decreased (inf --> 1.620445).  Saving model …
Training loss: 1.6205769205093383
Validation loss decreased (inf --> 1.611806).  Saving model …
Training loss: 1.6133470114072164
Validation loss decreased (inf --> 1.605209).  Saving model …
Training loss: 1.6076711853345236
Validation loss decreased (inf --> 1.599937).  Saving model …
Training loss: 1.6030133660634358
Validation loss decreased (inf --> 1.595585).  Saving model …
Training loss: 1.5990645972887676
Validation loss decreased (inf --> 1.591902).  Saving model …
Training loss: 1.5956467660268148
Validation loss decreased (inf --> 1.588732).  Saving model …
Training loss: 1.5926383344332378
Validation loss decreased (inf --> 1.585965).  Saving model …
Training loss: 1.589955721696218
Validation loss decreased (inf --> 1.583509).  Saving model …
Training loss: 1.587540872891744
```

```
Validation loss decreased (inf --> 1.581305).  Saving model …
Training loss: 1.585350136756897
Validation loss decreased (inf --> 1.579318).  Saving model …
Training loss: 1.5833489569028218
Validation loss decreased (inf --> 1.577517).  Saving model …
Training loss: 1.5815072202682494
Validation loss decreased (inf --> 1.575884).  Saving model …
Training loss: 1.5797985378901165
Validation loss decreased (inf --> 1.574401).  Saving model …
Training loss: 1.5782002814610798
Validation loss decreased (inf --> 1.573049).  Saving model …
Training loss: 1.5766905919710794
Validation loss decreased (inf --> 1.571811).  Saving model …
Training loss: 1.5752519806226095
Validation loss decreased (inf --> 1.570659).  Saving model …
Training loss: 1.5738713836669922
Validation loss decreased (inf --> 1.569561).  Saving model …
Training loss: 1.5725414315859476
Validation loss decreased (inf --> 1.568476).  Saving model …
Training loss: 1.5712592840194701
Validation loss decreased (inf --> 1.567371).  Saving model …
Training loss: 1.57002894560496
Validation loss decreased (inf --> 1.566245).  Saving model …
Training loss: 1.5688621234893798
Validation loss decreased (inf --> 1.565134).  Saving model …
Training loss: 1.567761800289154
Validation loss decreased (inf --> 1.564063).  Saving model …
Training loss: 1.5667235898971557
Validation loss decreased (inf --> 1.563045).  Saving model …
Training loss: 1.565739302635193
Validation loss decreased (inf --> 1.562078).  Saving model …
Training loss: 1.5648015451431274
Validation loss decreased (inf --> 1.561165).  Saving model …
Training loss: 1.5639058001836141
Validation loss decreased (inf --> 1.560304).  Saving model …
Training loss: 1.5630480575561523
Validation loss decreased (inf --> 1.559489).  Saving model …
Training loss: 1.5622252678871156
Validation loss decreased (inf --> 1.558719).  Saving model …
Training loss: 1.5614344676335652
Validation loss decreased (inf --> 1.557991).  Saving model …
Training loss: 1.5606729586919148
Validation loss decreased (inf --> 1.557302).  Saving model …
Training loss: 1.5599395219484966
Validation loss decreased (inf --> 1.556647).  Saving model …
Training loss: 1.5592334151268006
Validation loss decreased (inf --> 1.556024).  Saving model …
Training loss: 1.5585511962572733
```

```
Validation loss decreased (inf --> 1.555432).  Saving model …
Training loss: 1.5578924004236858
Validation loss decreased (inf --> 1.554869).  Saving model …
Training loss: 1.557255784670512
Validation loss decreased (inf --> 1.554330).  Saving model …
Training loss: 1.5566395608584087
Validation loss decreased (inf --> 1.553815).  Saving model …
Training loss: 1.556042095820109
Validation loss decreased (inf --> 1.553325).  Saving model …
Training loss: 1.555461429754893
Validation loss decreased (inf --> 1.552851).  Saving model …
Training loss: 1.5548971462249757
Validation loss decreased (inf --> 1.552395).  Saving model …
Training loss: 1.5543478894233704
Validation loss decreased (inf --> 1.551955).  Saving model …
Training loss: 1.5538134296735129
Validation loss decreased (inf --> 1.551530).  Saving model …
Training loss: 1.5532924071947734
Validation loss decreased (inf --> 1.551122).  Saving model …
Training loss: 1.5527848633130392
Validation loss decreased (inf --> 1.550729).  Saving model …
Training loss: 1.5522894740104676
Validation loss decreased (inf --> 1.550347).  Saving model …
```



```
1.5522894740104676
1.550346553325653
```

[ ]:
```
#Test loss
model3.eval()
test_loss = 0.0
num_test = 0.0
t_running_loss = 0.0
for t_image, t_labels in testLoader:
    t_image = t_image.view(t_image.shape[0], -1)
    outputs_t = model3(t_image)
    num_test+=1
    loss_t = criterion(outputs_t.squeeze(), t_labels.squeeze())
    t_running_loss += loss_t
test_loss = t_running_loss/num_test
print('Test Loss: {:.6f}\n'.format(test_loss))
```

```
Test Loss: 1.553071
```

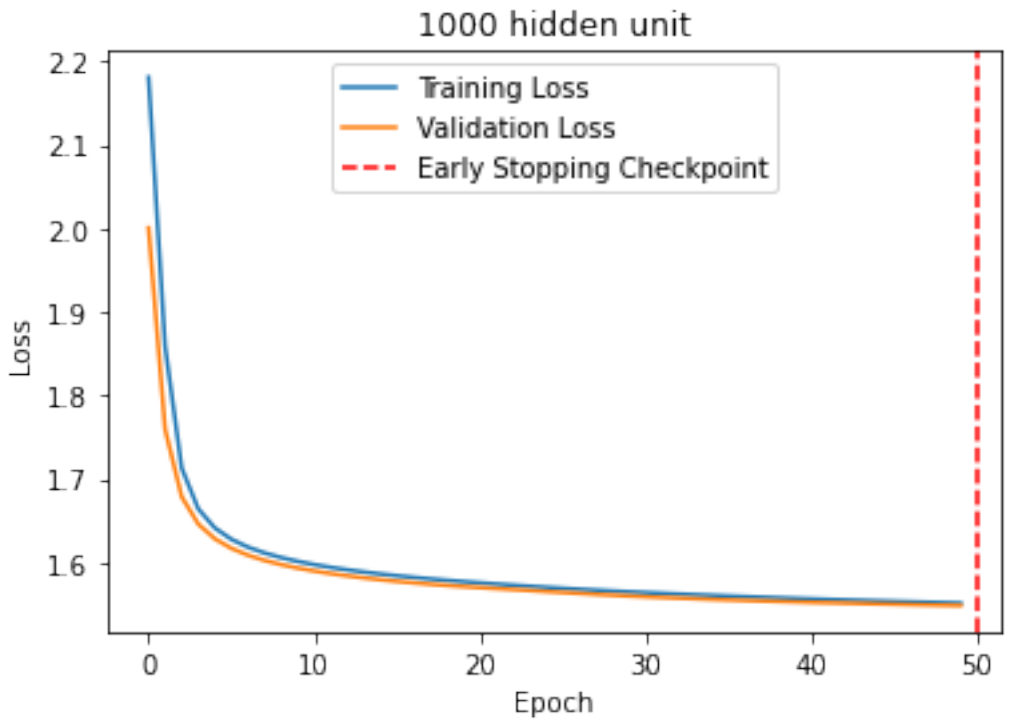
[ ]:
```
alpha = 0.01
epochs = 50
seed = 0
model4, criterion, optimizer = load_model4(alpha)
lossRec = []
varlossRec = []
nRec = []
timeRec = []

torch.manual_seed(seed)

for e in range(epochs):
    running_loss = 0
    v_running_loss = 0
    batch_cnt = 0
    v_batch_cnt = 0

    early_stopping = EarlyStopping(patience = 7, verbose = True)

    for images, labels in trainLoader:
        # Flatten MNIST images into a 784 long vector
        images = images.view(images.shape[0], -1)
        batch_cnt+=1
        # Training pass
        optimizer.zero_grad()
        output = model4(images)
        loss = criterion(output.squeeze(), labels.squeeze())
```

```
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    else:
        print(f"Training loss: {running_loss/len(trainLoader)}")

    with torch.no_grad():
        for v_image, v_labels in validationLoader:
            v_image = v_image.view(v_image.shape[0], -1)
            v_batch_cnt += 1
            optimizer.zero_grad()
            outputs_v = model4(v_image)
            loss_v = criterion(outputs_v.squeeze(), v_labels.squeeze())
            v_running_loss += loss_v.item()


    nRec.append(e)
    lossRec.append(running_loss/batch_cnt)
    varlossRec.append(v_running_loss/v_batch_cnt)
    timeRec.append(time.time())

    early_stopping(v_running_loss/v_batch_cnt, model4)
    if early_stopping.early_stop:
            print("Early stopping")
            break

plt.plot(nRec, lossRec, label="Training Loss")
plt.plot(nRec, varlossRec, label="Validation Loss")

minposs = varlossRec.index(min(varlossRec))+1
plt.axvline(minposs, linestyle='--', color='r',label='Early Stopping␣
 ↪Checkpoint')

plt.title("1000 hidden unit")
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
print(lossRec[-1]) #Train loss
print(varlossRec[-1]) #validation loss
```

```
Training loss: 2.181005589167277
Validation loss decreased (inf --> 2.000566).  Saving model …
Training loss: 1.8587437844276429
Validation loss decreased (inf --> 1.759259).  Saving model …
Training loss: 1.7130081168810527
```

```
Validation loss decreased (inf --> 1.678421).  Saving model …
Training loss: 1.6640508755048116
Validation loss decreased (inf --> 1.645865).  Saving model …
Training loss: 1.641198693116506
Validation loss decreased (inf --> 1.628138).  Saving model …
Training loss: 1.627540086110433
Validation loss decreased (inf --> 1.616702).  Saving model …
Training loss: 1.6181757609049479
Validation loss decreased (inf --> 1.608512).  Saving model …
Training loss: 1.6111844277381897
Validation loss decreased (inf --> 1.602245).  Saving model …
Training loss: 1.605657029946645
Validation loss decreased (inf --> 1.597230).  Saving model …
Training loss: 1.6011062335968018
Validation loss decreased (inf --> 1.593073).  Saving model …
Training loss: 1.5972486567497253
Validation loss decreased (inf --> 1.589536).  Saving model …
Training loss: 1.5939077130953472
Validation loss decreased (inf --> 1.586462).  Saving model …
Training loss: 1.5909696420033772
Validation loss decreased (inf --> 1.583749).  Saving model …
Training loss: 1.5883525649706522
Validation loss decreased (inf --> 1.581322).  Saving model …
Training loss: 1.585999864737193
Validation loss decreased (inf --> 1.579139).  Saving model …
Training loss: 1.5838667035102845
Validation loss decreased (inf --> 1.577169).  Saving model …
Training loss: 1.581914583047231
Validation loss decreased (inf --> 1.575389).  Saving model …
Training loss: 1.5801136962572733
Validation loss decreased (inf --> 1.573781).  Saving model …
Training loss: 1.5784387302398681
Validation loss decreased (inf --> 1.572321).  Saving model …
Training loss: 1.5768662484486897
Validation loss decreased (inf --> 1.570983).  Saving model …
Training loss: 1.5753769103686015
Validation loss decreased (inf --> 1.569741).  Saving model …
Training loss: 1.5739537866910298
Validation loss decreased (inf --> 1.568558).  Saving model …
Training loss: 1.5725838979085287
Validation loss decreased (inf --> 1.567396).  Saving model …
Training loss: 1.5712649218241375
Validation loss decreased (inf --> 1.566227).  Saving model …
Training loss: 1.5700044616063435
Validation loss decreased (inf --> 1.565046).  Saving model …
Training loss: 1.5688109230995178
Validation loss decreased (inf --> 1.563881).  Saving model …
Training loss: 1.5676837952931721
```

```
Validation loss decreased (inf --> 1.562754).  Saving model …
Training loss: 1.566615449587504
Validation loss decreased (inf --> 1.561675).  Saving model …
Training loss: 1.5655982184410095
Validation loss decreased (inf --> 1.560651).  Saving model …
Training loss: 1.5646256136894225
Validation loss decreased (inf --> 1.559685).  Saving model …
Training loss: 1.5636935830116272
Validation loss decreased (inf --> 1.558775).  Saving model …
Training loss: 1.5627996182441712
Validation loss decreased (inf --> 1.557917).  Saving model …
Training loss: 1.5619419805208843
Validation loss decreased (inf --> 1.557112).  Saving model …
Training loss: 1.5611180861790974
Validation loss decreased (inf --> 1.556356).  Saving model …
Training loss: 1.560326073964437
Validation loss decreased (inf --> 1.555645).  Saving model …
Training loss: 1.5595643329620361
Validation loss decreased (inf --> 1.554976).  Saving model …
Training loss: 1.5588306546211244
Validation loss decreased (inf --> 1.554343).  Saving model …
Training loss: 1.5581225593884787
Validation loss decreased (inf --> 1.553743).  Saving model …
Training loss: 1.5574381295839945
Validation loss decreased (inf --> 1.553173).  Saving model …
Training loss: 1.5567758385340373
Validation loss decreased (inf --> 1.552632).  Saving model …
Training loss: 1.5561342755953471
Validation loss decreased (inf --> 1.552114).  Saving model …
Training loss: 1.5555115421613057
Validation loss decreased (inf --> 1.551618).  Saving model …
Training loss: 1.5549063595136006
Validation loss decreased (inf --> 1.551143).  Saving model …
Training loss: 1.5543177858988444
Validation loss decreased (inf --> 1.550688).  Saving model …
Training loss: 1.553745137055715
Validation loss decreased (inf --> 1.550251).  Saving model …
Training loss: 1.5531867408752442
Validation loss decreased (inf --> 1.549828).  Saving model …
Training loss: 1.5526422580083212
Validation loss decreased (inf --> 1.549422).  Saving model …
Training loss: 1.5521109668413797
Validation loss decreased (inf --> 1.549030).  Saving model …
Training loss: 1.5515920042991638
Validation loss decreased (inf --> 1.548652).  Saving model …
Training loss: 1.5510855555534362
Validation loss decreased (inf --> 1.548287).  Saving model …
```




```
1.5510855555534362
1.5482874751091003
```

[ ]:
```
#Test loss
model4.eval()
test_loss = 0.0
num_test = 0.0

t_running_loss = 0.0
for t_image, t_labels in testLoader:
    t_image = t_image.view(t_image.shape[0], -1)
    outputs_t = model4(t_image)
    num_test+=1
    loss_t = criterion(outputs_t.squeeze(), t_labels.squeeze())
    t_running_loss += loss_t
test_loss = t_running_loss/num_test
print('Test Loss: {:.6f}\n'.format(test_loss))
```

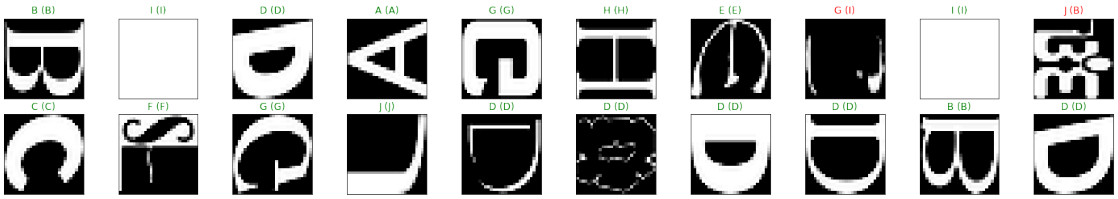
```
Test Loss: 1.552814
```

[ ]:
```
dataiter = iter(testLoader)
images, labels = dataiter.next()
images2 = images.view(images.shape[0], -1)
```

```python
model3.eval()
outputs_t = model3(images2)
_, preds = torch.max(outputs_t, 1)
fig = plt.figure(figsize=(25, 4))
for idx in np.arange(20):
    ax = fig.add_subplot(2, 20/2, idx+1, xticks=[], yticks=[])
    ax.imshow(np.squeeze(images[idx].T), cmap='gray')
    ax.set_title("{} ({})".format(str(classes[preds[idx].item()]),␣
 ↪str(classes[labels[idx].item()])),
                 color=("green" if preds[idx]==labels[idx] else "red"))
```

```
<ipython-input-153-629db365ad6a>:9: MatplotlibDeprecationWarning: Passing non-
integers as three-element position specification is deprecated since 3.3 and
will be removed two minor releases later.
  ax = fig.add_subplot(2, 20/2, idx+1, xticks=[], yticks=[])
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
```



When hidden units are 500, the loss after 50 epoch has been minized. This shows that either too small or large number of layers does not have correlation with the result of NN.

5 Task IV - Testing the Number of Layers (25 marks):

[ ]:
```
class MLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(28*28*3, 500)
        self.fc2 = nn.Linear(500, 500)
        self.fc3 = nn.Linear(500, 10)
        self.relu = nn.ReLU()
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.fc3(x)
        x = self.softmax(x)
        return x
```

[ ]:
```
def load_model5(lr):
    net = MLP()
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(net.parameters(), lr)
    return net, criterion, optimizer
```

[ ]:
```
alpha = 0.01
epochs = 50
seed = 0
model5, criterion, optimizer = load_model5(alpha)
```

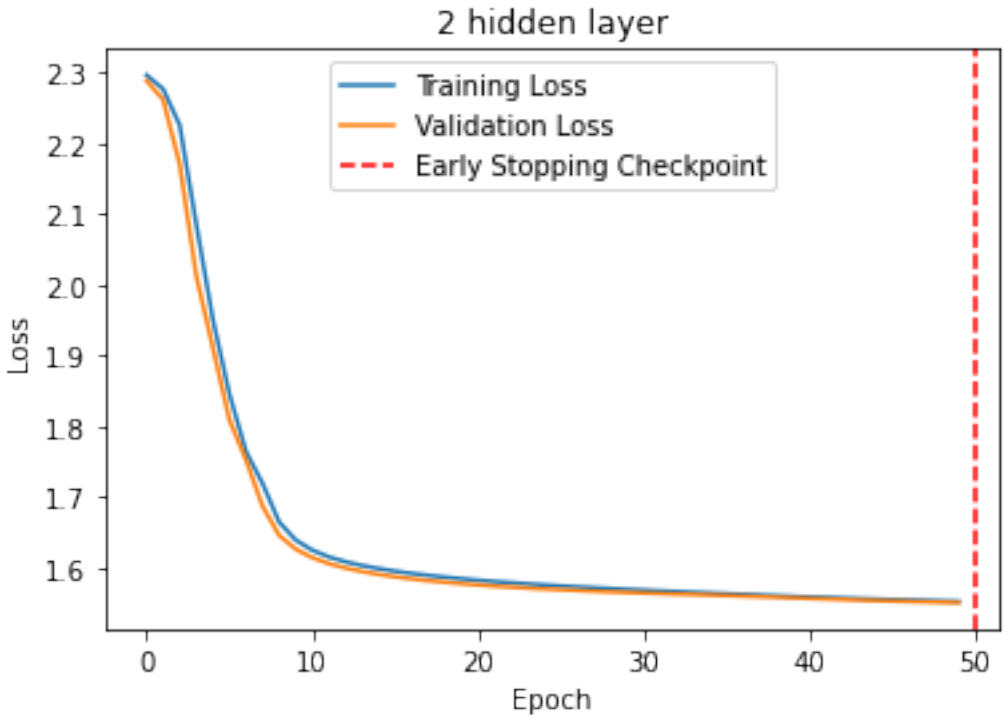
```
lossRec = []
varlossRec = []
nRec = []
timeRec = []

torch.manual_seed(seed)

for e in range(epochs):
    running_loss = 0
    v_running_loss = 0
    batch_cnt = 0
    v_batch_cnt = 0

    early_stopping = EarlyStopping(patience = 7, verbose = True)

    for images, labels in trainLoader:
        # Flatten MNIST images into a 784 long vector
        images = images.view(images.shape[0], -1)
        batch_cnt+=1
        # Training pass
        optimizer.zero_grad()
        output = model5(images)
        loss = criterion(output.squeeze(), labels.squeeze())
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    else:
        print(f"Training loss: {running_loss/len(trainLoader)}")

    with torch.no_grad():
        for v_image, v_labels in validationLoader:
            v_image = v_image.view(v_image.shape[0], -1)
            v_batch_cnt += 1
            optimizer.zero_grad()
            outputs_v = model5(v_image)
            loss_v = criterion(outputs_v.squeeze(), v_labels.squeeze())
            v_running_loss += loss_v.item()


    nRec.append(e)
    lossRec.append(running_loss/batch_cnt)
    varlossRec.append(v_running_loss/v_batch_cnt)
    timeRec.append(time.time())

    early_stopping(v_running_loss/v_batch_cnt, model5)
    if early_stopping.early_stop:
```

```
            print("Early stopping")
            break

plt.plot(nRec, lossRec, label="Training Loss")
plt.plot(nRec, varlossRec, label="Validation Loss")

minposs = varlossRec.index(min(varlossRec))+1
plt.axvline(minposs, linestyle='--', color='r',label='Early Stopping␣
 ↪Checkpoint')

plt.title("2 hidden layer")
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
print(lossRec[-1]) #Train loss
print(varlossRec[-1]) #validation loss
```

```
Training loss: 2.2958771896362307
Validation loss decreased (inf --> 2.288181).  Saving model …
Training loss: 2.276225158373515
Validation loss decreased (inf --> 2.260999).  Saving model …
Training loss: 2.225175773302714
Validation loss decreased (inf --> 2.169616).  Saving model …
Training loss: 2.0853132104873655
Validation loss decreased (inf --> 2.012571).  Saving model …
Training loss: 1.9508160217603048
Validation loss decreased (inf --> 1.910475).  Saving model …
Training loss: 1.8451425909996033
Validation loss decreased (inf --> 1.808545).  Saving model …
Training loss: 1.763903213342031
Validation loss decreased (inf --> 1.751959).  Saving model …
Training loss: 1.7186936783790587
Validation loss decreased (inf --> 1.686429).  Saving model …
Training loss: 1.6642463143666586
Validation loss decreased (inf --> 1.645344).  Saving model …
Training loss: 1.6387910318374634
Validation loss decreased (inf --> 1.625845).  Saving model …
Training loss: 1.6243958338101705
Validation loss decreased (inf --> 1.613819).  Saving model …
Training loss: 1.6149254910151163
Validation loss decreased (inf --> 1.605472).  Saving model …
Training loss: 1.6080501500765483
Validation loss decreased (inf --> 1.599223).  Saving model …
Training loss: 1.602716236114502
Validation loss decreased (inf --> 1.594280).  Saving model …
Training loss: 1.5983773589134216
```

```
Validation loss decreased (inf --> 1.590218).  Saving model …
Training loss: 1.5947199591000876
Validation loss decreased (inf --> 1.586808).  Saving model …
Training loss: 1.5915551511446635
Validation loss decreased (inf --> 1.583888).  Saving model …
Training loss: 1.588761528333028
Validation loss decreased (inf --> 1.581347).  Saving model …
Training loss: 1.5862662164370218
Validation loss decreased (inf --> 1.579103).  Saving model …
Training loss: 1.5840127571423848
Validation loss decreased (inf --> 1.577112).  Saving model …
Training loss: 1.5819629216194153
Validation loss decreased (inf --> 1.575326).  Saving model …
Training loss: 1.5800851130485534
Validation loss decreased (inf --> 1.573711).  Saving model …
Training loss: 1.5783558050791422
Validation loss decreased (inf --> 1.572237).  Saving model …
Training loss: 1.5767548823356627
Validation loss decreased (inf --> 1.570885).  Saving model …
Training loss: 1.5752645881970724
Validation loss decreased (inf --> 1.569638).  Saving model …
Training loss: 1.5738726290067038
Validation loss decreased (inf --> 1.568492).  Saving model …
Training loss: 1.5725658496220907
Validation loss decreased (inf --> 1.567436).  Saving model …
Training loss: 1.571331700483958
Validation loss decreased (inf --> 1.566462).  Saving model …
Training loss: 1.5701574738820394
Validation loss decreased (inf --> 1.565558).  Saving model …
Training loss: 1.5690332476298015
Validation loss decreased (inf --> 1.564717).  Saving model …
Training loss: 1.5679496765136718
Validation loss decreased (inf --> 1.563928).  Saving model …
Training loss: 1.5668963805834453
Validation loss decreased (inf --> 1.563184).  Saving model …
Training loss: 1.5658660395940145
Validation loss decreased (inf --> 1.562477).  Saving model …
Training loss: 1.5648532088597615
Validation loss decreased (inf --> 1.561784).  Saving model …
Training loss: 1.563854209582011
Validation loss decreased (inf --> 1.561087).  Saving model …
Training loss: 1.562869443098704
Validation loss decreased (inf --> 1.560355).  Saving model …
Training loss: 1.5618987409273783
Validation loss decreased (inf --> 1.559565).  Saving model …
Training loss: 1.56094775279363
Validation loss decreased (inf --> 1.558723).  Saving model …
Training loss: 1.560025138060252
```

```
Validation loss decreased (inf --> 1.557862).  Saving model …
Training loss: 1.5591410636901855
Validation loss decreased (inf --> 1.557015).  Saving model …
Training loss: 1.5582989875475566
Validation loss decreased (inf --> 1.556195).  Saving model …
Training loss: 1.5574971230824788
Validation loss decreased (inf --> 1.555409).  Saving model …
Training loss: 1.5567300462722777
Validation loss decreased (inf --> 1.554663).  Saving model …
Training loss: 1.555992157459259
Validation loss decreased (inf --> 1.553954).  Saving model …
Training loss: 1.5552813307444255
Validation loss decreased (inf --> 1.553277).  Saving model …
Training loss: 1.554593575000763
Validation loss decreased (inf --> 1.552629).  Saving model …
Training loss: 1.5539258623123169
Validation loss decreased (inf --> 1.552005).  Saving model …
Training loss: 1.5532759483655294
Validation loss decreased (inf --> 1.551403).  Saving model …
Training loss: 1.552643248240153
Validation loss decreased (inf --> 1.550822).  Saving model …
Training loss: 1.5520249064763387
Validation loss decreased (inf --> 1.550266).  Saving model …
```



```
1.5520249064763387
1.5502662658691406
```

[ ]:
```python
#Test loss
model5.eval()
test_loss = 0.0
num_test = 0.0
t_running_loss = 0.0
for t_image, t_labels in testLoader:
    t_image = t_image.view(t_image.shape[0], -1)
    outputs_t = model5(t_image)
    num_test+=1
    loss_t = criterion(outputs_t.squeeze(), t_labels.squeeze())
    t_running_loss += loss_t
test_loss = t_running_loss/num_test
print('Test Loss: {:.6f}\n'.format(test_loss))
```

```
Test Loss: 1.553015
```

Compared to the one-layercase, the graph tends to be looking like logistic function with more overfitted datas.

6 Task V - Dropout (30 marks):

[ ]:
```python
class SNN6(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(28*28*3, 1000)
        self.fc2 = nn.Linear(1000, 10)
        self.relu = nn.ReLU()
        self.softmax = nn.Softmax(dim=1)
        self.dropout = nn.Dropout(0.5)


    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.fc2(x)
        x = self.dropout(x)

        x = self.softmax(x)
        return x
```

[ ]:
```
def load_model6(lr):
    net = SNN6()
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(net.parameters(), lr)
    return net, criterion, optimizer
```

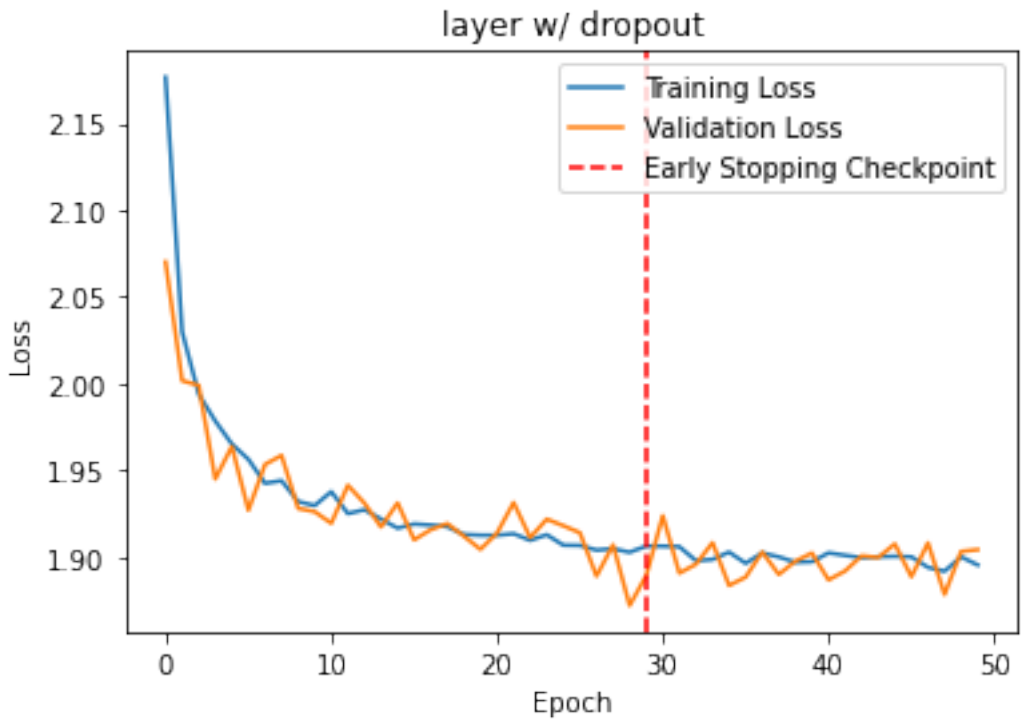
[ ]:
```
alpha = 0.01
epochs = 50
seed = 0
model6, criterion, optimizer = load_model6(alpha)
lossRec = []
varlossRec = []
nRec = []
timeRec = []

torch.manual_seed(seed)

for e in range(epochs):
    running_loss = 0
    v_running_loss = 0
    batch_cnt = 0
    v_batch_cnt = 0

    early_stopping = EarlyStopping(patience = 7, verbose = True)

    for images, labels in trainLoader:
        # Flatten MNIST images into a 784 long vector
        images = images.view(images.shape[0], -1)
        batch_cnt+=1
        # Training pass
        optimizer.zero_grad()
        output = model6(images)
        loss = criterion(output.squeeze(), labels.squeeze())
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    else:
        print(f"Training loss: {running_loss/len(trainLoader)}")

    with torch.no_grad():
        for v_image, v_labels in validationLoader:
            v_image = v_image.view(v_image.shape[0], -1)
            v_batch_cnt += 1
            optimizer.zero_grad()
            outputs_v = model6(v_image)
            loss_v = criterion(outputs_v.squeeze(), v_labels.squeeze())
```

```
            v_running_loss += loss_v.item()


    nRec.append(e)
    lossRec.append(running_loss/batch_cnt)
    varlossRec.append(v_running_loss/v_batch_cnt)
    timeRec.append(time.time())

    early_stopping(v_running_loss/v_batch_cnt, model6)
    if early_stopping.early_stop:
            print("Early stopping")
            break

plt.plot(nRec, lossRec, label="Training Loss")
plt.plot(nRec, varlossRec, label="Validation Loss")

minposs = varlossRec.index(min(varlossRec))+1
plt.axvline(minposs, linestyle='--', color='r',label='Early Stopping␣
 ↪Checkpoint')

plt.title("layer w/ dropout")
plt.legend()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
print(lossRec[-1]) #Train loss
print(varlossRec[-1]) #validation loss
```

```
Training loss: 2.1766559267044068
Validation loss decreased (inf --> 2.070105).  Saving model …
Training loss: 2.0298939124743143
Validation loss decreased (inf --> 2.001570).  Saving model …
Training loss: 1.9937926038106282
Validation loss decreased (inf --> 1.998897).  Saving model …
Training loss: 1.9782878931363423
Validation loss decreased (inf --> 1.944852).  Saving model …
Training loss: 1.965030992825826
Validation loss decreased (inf --> 1.963637).  Saving model …
Training loss: 1.9561847201983134
Validation loss decreased (inf --> 1.926845).  Saving model …
Training loss: 1.9423652561505635
Validation loss decreased (inf --> 1.953238).  Saving model …
Training loss: 1.9438869428634644
Validation loss decreased (inf --> 1.958441).  Saving model …
Training loss: 1.9317857313156128
Validation loss decreased (inf --> 1.927873).  Saving model …
Training loss: 1.92953684091568
```

```
Validation loss decreased (inf --> 1.926002).  Saving model …
Training loss: 1.9375301567713419
Validation loss decreased (inf --> 1.919280).  Saving model …
Training loss: 1.9248945260047912
Validation loss decreased (inf --> 1.941250).  Saving model …
Training loss: 1.927096548875173
Validation loss decreased (inf --> 1.930908).  Saving model …
Training loss: 1.9217398142814637
Validation loss decreased (inf --> 1.917167).  Saving model …
Training loss: 1.9164837940533956
Validation loss decreased (inf --> 1.931051).  Saving model …
Training loss: 1.918897409439087
Validation loss decreased (inf --> 1.909738).  Saving model …
Training loss: 1.918248545328776
Validation loss decreased (inf --> 1.915612).  Saving model …
Training loss: 1.91731707572937
Validation loss decreased (inf --> 1.919113).  Saving model …
Training loss: 1.912848379611969
Validation loss decreased (inf --> 1.911863).  Saving model …
Training loss: 1.912419023513794
Validation loss decreased (inf --> 1.904261).  Saving model …
Training loss: 1.9123784923553466
Validation loss decreased (inf --> 1.913606).  Saving model …
Training loss: 1.913260099887848
Validation loss decreased (inf --> 1.931235).  Saving model …
Training loss: 1.9094761617978413
Validation loss decreased (inf --> 1.911001).  Saving model …
Training loss: 1.9127055795987447
Validation loss decreased (inf --> 1.921668).  Saving model …
Training loss: 1.906659870147705
Validation loss decreased (inf --> 1.917897).  Saving model …
Training loss: 1.9065086356798808
Validation loss decreased (inf --> 1.913857).  Saving model …
Training loss: 1.9038555685679117
Validation loss decreased (inf --> 1.888821).  Saving model …
Training loss: 1.9045845675468445
Validation loss decreased (inf --> 1.906872).  Saving model …
Training loss: 1.9025906538963318
Validation loss decreased (inf --> 1.871878).  Saving model …
Training loss: 1.9061221408843994
Validation loss decreased (inf --> 1.889077).  Saving model …
Training loss: 1.906087376276652
Validation loss decreased (inf --> 1.923677).  Saving model …
Training loss: 1.9058279132843017
Validation loss decreased (inf --> 1.890558).  Saving model …
Training loss: 1.8976996692021688
Validation loss decreased (inf --> 1.895676).  Saving model …
Training loss: 1.898456827799479
```

```
Validation loss decreased (inf --> 1.908113).  Saving model …
Training loss: 1.9026797127723694
Validation loss decreased (inf --> 1.883417).  Saving model …
Training loss: 1.896220326423645
Validation loss decreased (inf --> 1.888254).  Saving model …
Training loss: 1.9021527727444967
Validation loss decreased (inf --> 1.902707).  Saving model …
Training loss: 1.8998921012878418
Validation loss decreased (inf --> 1.889807).  Saving model …
Training loss: 1.8968381182352703
Validation loss decreased (inf --> 1.897429).  Saving model …
Training loss: 1.8971299648284912
Validation loss decreased (inf --> 1.902095).  Saving model …
Training loss: 1.902225205898285
Validation loss decreased (inf --> 1.886629).  Saving model …
Training loss: 1.9007794149716695
Validation loss decreased (inf --> 1.891799).  Saving model …
Training loss: 1.8994683949152629
Validation loss decreased (inf --> 1.900516).  Saving model …
Training loss: 1.9000127204259236
Validation loss decreased (inf --> 1.899561).  Saving model …
Training loss: 1.9002482787768047
Validation loss decreased (inf --> 1.907384).  Saving model …
Training loss: 1.899950065612793
Validation loss decreased (inf --> 1.888455).  Saving model …
Training loss: 1.893859206835429
Validation loss decreased (inf --> 1.907986).  Saving model …
Training loss: 1.8916439747810363
Validation loss decreased (inf --> 1.878220).  Saving model …
Training loss: 1.8999462683995565
Validation loss decreased (inf --> 1.903047).  Saving model …
Training loss: 1.8952134235699971
Validation loss decreased (inf --> 1.904067).  Saving model …
```




```
1.8952134235699971
1.9040672779083252
```

As seen from the result graph of using dropout, it definitely reaches the minimum loss within small epoch compared to any of the previous network in this assignment without the use of dropout.