1 Fashion-MNIST Classification

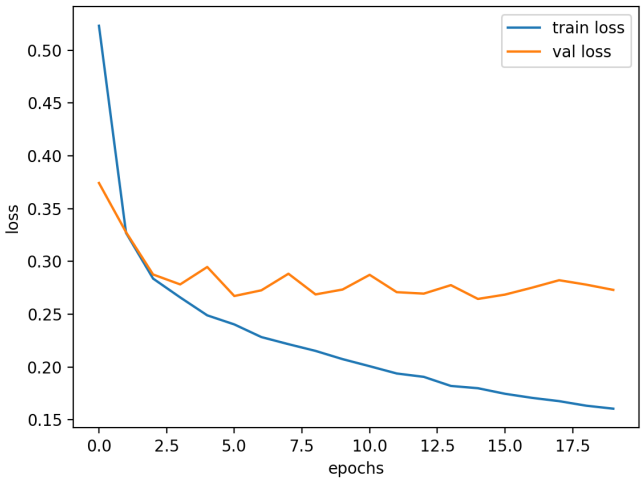
Submit a program which trains with your best combination of model architecture, optimizer and training parameters, and evaluates on the test set to report an accuracy at the end.

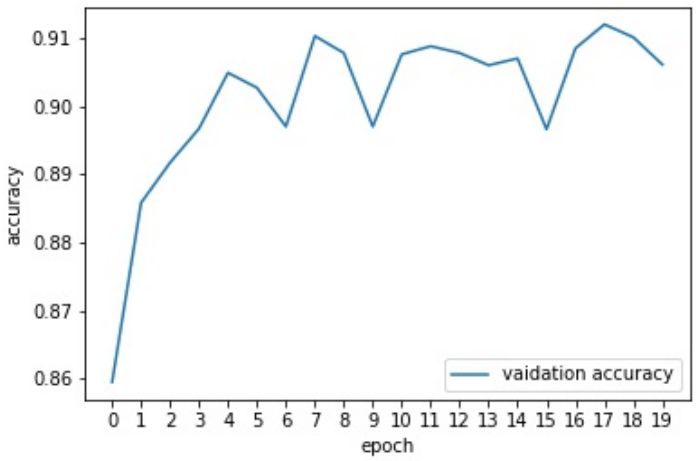
Submitted in CANVAS

Report the detailed architecture of your best model. Include information on hyperparameters chosen for training and a plot showing both training and validation loss across iterations. (10 pts)

-Hyperparameters
learning rate = 1e-3
weight_decay = 1e-4
number of epoch = 20
batch size = 64
optimizer = adam
Model architecture:
model was composed of convolutional layers followed by reLu max pool and linear layers.

```
nn.Conv2d(1, 12, kernel_size=3, stride=1, padding=1),
nn.ReLU(),
nn.MaxPool2d(kernel_size=2, stride=2)
nn.Conv2d(12, 24, kernel_size=3, stride=1, padding=1),
nn.ReLU(),
nn.MaxPool2d(kernel_size=2, stride=2))
nn.Dropout()
nn.Linear(7 * 7 * 24, 500)
nn.Linear(500, 10)
```





one can observe form that plot that both losses decreases as every iteration and validation accuracies increases as number of epochs increases

Report the accuracy of your best model on the test set. We expect you to achieve over 90%. (10 pts)

accuracy = 90.05 %

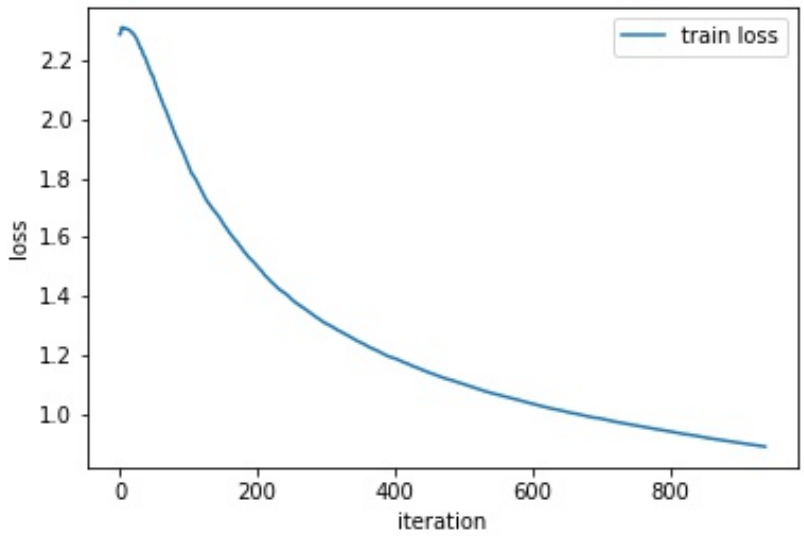
2 Activation Visualization

Report the detailed architecture of your self.base module. Include information on hyperparameters chosen for training, and the accuracy on the test set. To make the visualization look nice, you should achieve over 80% on the test set.

-Hyperparameters
learning rate = 0.001
number of epoch = 10
batch size = 64
optimizer = adam
Model architecture (self.base):
Model architecture was composed of convolution layers ReLu activation followed by maxpool and dropout. The exact dimensions are as below:

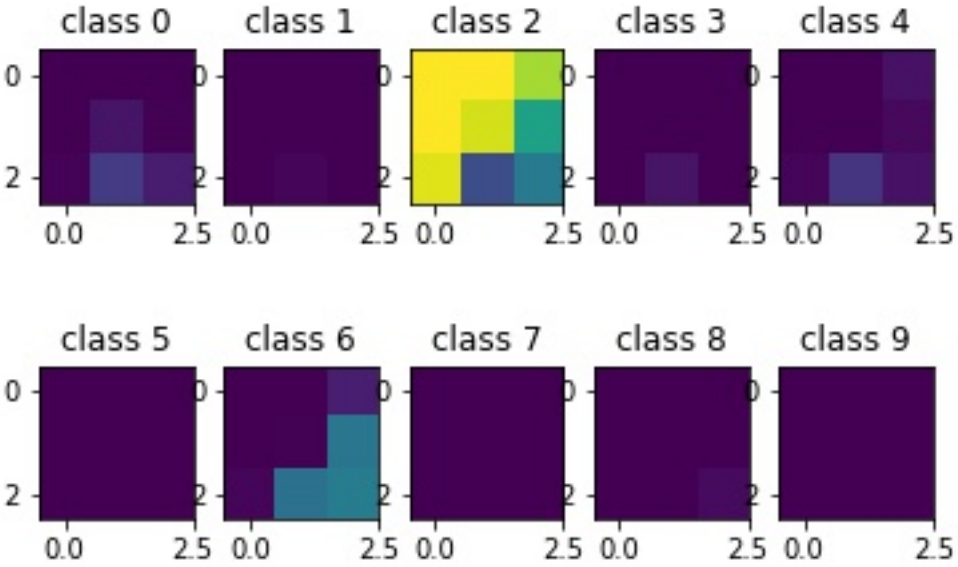
```
nn.Conv2d(1, 12, kernel_size=5, stride=1, padding=2),
nn.ReLU(),
nn.MaxPool2d(kernel_size=2, stride=2),

nn.Conv2d(12, 48, kernel_size=8, stride=2, padding=3),
nn.ReLU(),

nn.Conv2d(48, 24, kernel_size=8, stride=2, padding=3),
nn.ReLU(),

nn.Conv2d(24, out_channel, kernel_size=5, stride=1, padding=2),
nn.ReLU(),

nn.MaxPool2d(kernel_size=2, stride=2),
nn.Dropout()
```

-**Output accuracy: 81%**



Choose a correctly classified image from the evaluation on test set. Report its index in the test set and include plots of both the image and the activation maps of all classes. (10 pts)

index in test test: 2
As shown in the below figure, the activation map of the classified label is activated the most. Also the activation is the highest where the fashion minist image is, instead of other number mnist image



Submit a program which contains your best combination of self.base mudule, optimizer and training parameters, along with the code to select a correctly classified image and to visualize the results. (15 pts)

Submitted in CANVAS

3 Semantic Segmentation [30 pts]

Report the detailed architecture of your model. Include information on hyperparameters chosen for training and a plot showing both training and validation loss across iterations

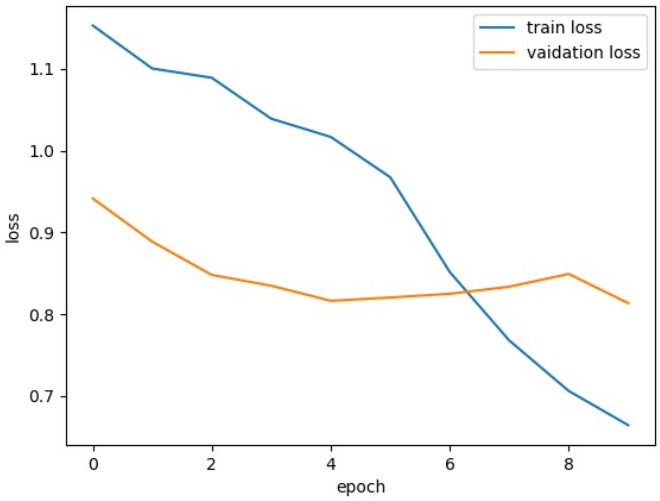
Hyperparameters
learning rate = 0.001
number of epoch = 10
batch size = 1
optimizer = adam
weight decay = 1e-4
Model architecture :
The model architecture was designed in a form of autoencoder, inspired by U-Net. Instead of a deep architecture which U-net has, I implemented a light-weighted version of authoencoder where it has only two down and up sampling each. Each downsampling blocks were composed of pooling, convolution followed by batch norm and RELU. Each upsampling blocks were composed of upsampling followed by convolution. The following is the detailed architecture:

```
self.conv0= nn.Conv2d(3, 64, 3, padding=1)

# down 1
self.pool1=nn.MaxPool2d(2)
self.conv1=nn.Conv2d(64, 128, 3, padding=1)
self.batchnorm1 =  nn.BatchNorm2d(128)
self.relu=nn.ReLU()

# down 2
self.pool2=nn.MaxPool2d(2)
self.conv2=nn.Conv2d(128, 128, 3, padding=1)
self.batchnorm2 =  nn.BatchNorm2d(128)
self.relu=nn.ReLU()

# up 1
self.up1=nn.Upsample(scale_factor=2, mode='bilinear', align_corners=True)
# self.conv3 = nn.Conv2d(192, 128, 3, padding=1)
self.conv_up1 = nn.Conv2d(256, 64, 3, padding=1)

# up 2
self.up2=nn.Upsample(scale_factor=2, mode='bilinear', align_corners=True)
self.conv_up2 = nn.Conv2d(128, 64, 3, padding=1)
self.outconv=nn.Conv2d(64, self.n_class, kernel_size=3, padding=1)
```

Forward pass

```
x0 = self.conv0(x) # torch.Size([1, 64, 256, 256])

# down1
x1 = self.pool1(x0)
x1 = self.conv1(x1)
x1 = self.batchnorm1(x1)
```

```
x1 = self.relu(x1) # torch.Size([1, 128, 128, 128])

# down2
x2 = self.pool2(x1) #torch.Size([1, 128, 64, 64])
x2 = self.conv2(x2) #torch.Size([1, 128, 64, 64])
x2 = self.batchnorm2(x2)
x2 = self.relu(x2)

# up1
x = self.up1(x2) #torch.Size([1, 128, 128, 128])
x = torch.cat([x,x1],dim=1) #torch.Size([1, 256, 128, 128])
x = self.conv_up1 (x) #torch.Size([1, 64, 128, 128])

# up2
x = self.up2(x) #torch.Size([1, 64, 256, 256])
x = torch.cat([x,x0],dim=1) #torch.Size([1, 128, 256, 256])
x = self.conv_up2(x)

logit = self.outconv(x)
```



Training loss and validation losses were plotted every epochs. As we can observe from the graph, overall the loss decreases as the number of epochs increases

Report the average precision on the test set. You can use provided function to calculate AP on the test set. You should only evaluate your model on the test set once. All hyperparameter tuning should be done on the validation set. We expect you to to achieve 0.45 AP on the test set. (10 pts)

AP = 0.5513151002328199
AP = 0.6898169392685897
AP = 0.15048516956690153
AP = 0.7987713311174914
AP = 0.33718163492871445
average AP = 50.2

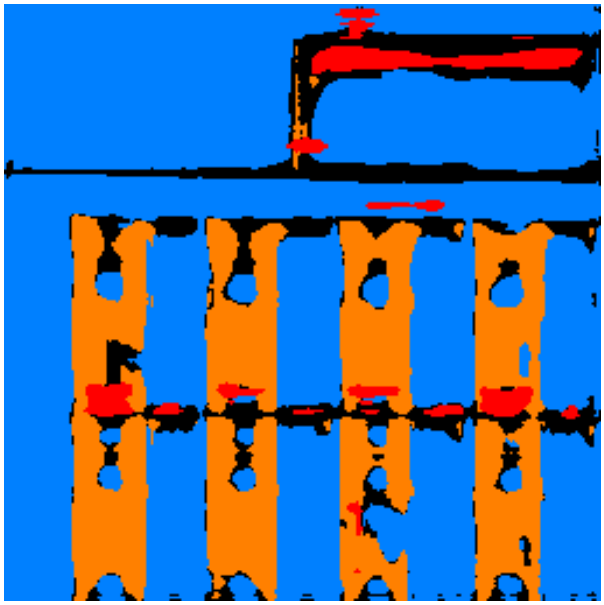
Submit a program which contains your best combination of self.base module, optimizer and training parameters.

Submitted in CANVAS

Take a photo of a building in UM, preprocess it as you like and input it to your best trained model. Plot the output labels and comment qualitatively on why it works or doesn't work. Submit the image as input.jpg and the output labels as output.png. (5 pts) Like HW1 Part3, we will have a contest where every student can vote for the best-labeled images

Input RGB image

output prediction



As one can see in the images above, the prediction (segmentation) is, in general, working. However, the segmentation fails in some area, for example in one of the windows the light is on the room. It misclassify as a facade as shown in the image. Thus this network is also sensitive to the color, not only to the shape.