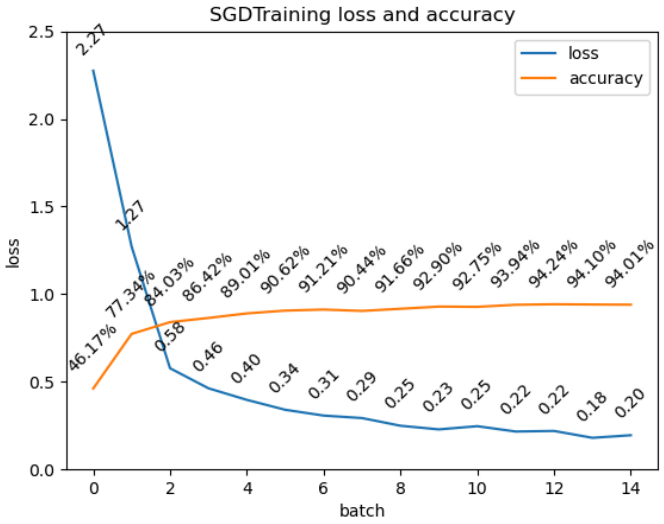
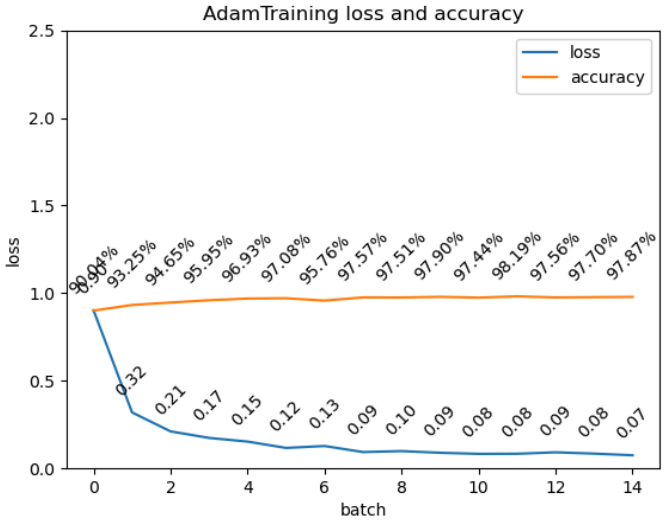
CV assignment 3
1.1 Answer the questions:

		Given a `batch_size=16`, what are the shapes of the input and output for the 7 specified layers?	How many trainable parameters does each layer contain?
0	Input image: 1x28x28	imput size 16*1*28*28	
1	Conv layer: ● kernel_size: 5x5 ● out_channels: 16 ● activation: ReLU	# convolutional layer 1 # output = 28 - 5 + 1 = 24 # (16, 1, 28, 28) -> (16, 6, 24, 24)	(1*5*5+1)*6 =156
2	Max pooling: ● kernel_size: 2x2	# pooling layer 1 (2,2) # output = 24/2 = 12 # (16, 6, 24, 24) -> (16, 6, 12, 12)	0
3	Conv layer: ● kernel_size: 3x3 ● out_channels: 32 ● activation: ReLU	# convolutional layer 2 # output = 12 - 3 + 1 = 10 # (16, 6, 12, 12) -> (16, 32, 10, 10)	(6*3*3+1)*32 = 1760
4	Max pooling: ● kernel_size: 2x2	#pooling layer 2 (2,2) # output = 10/2 = 5 # (16, 32, 10, 10) -> (16, 32, 5, 5)	0
5	Conv layer: ● kernel_size: 1x1 ● out_channels: 16 ● activation: ReLU	# convolutional layer 3 # output = 5 - 1 + 1 = 5 # (16, 32, 5, 5) -> (16, 16, 5, 5)	(32*1*1+1)*16 = 528
6	FC layer: ● in_features: **16 *5 *5 = 400** ● out_features: 64 ● activation: ReLU	FC layer1: input = (16,(16*5*5)) = (16,400) output = (16, 64)	400*64+64 = 25664
7	FC layer: ● out_features: **10** ● activation: None	FC layer2: input =(16, 64) output = (16, 10) (label have 10)	64*10+10 = 650

1.2 Training a CNN to Recognize Hand-written Digits in the MNIST Dataset.
b.Experiment with different optimizers such as Adam in addition to the provided SGD optimizer, and **discuss** their impact on the final results. (1 point)





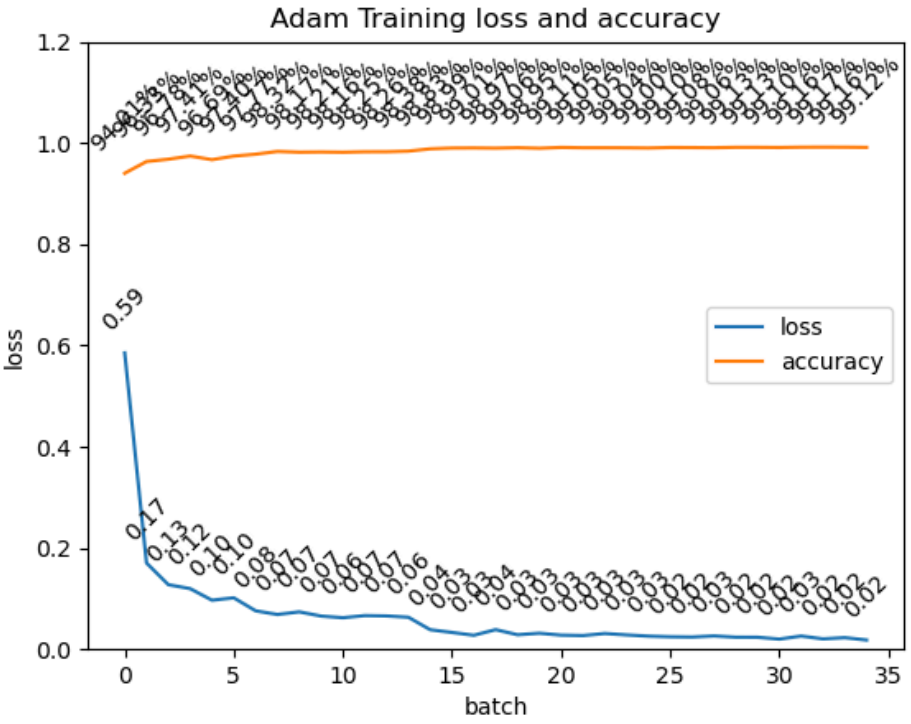
SGD Optimizer: Final loss: 0.20 Final accuracy: 94.01%

Adam Optimizer: Final loss: 0.07 Final accuracy: 97.87%

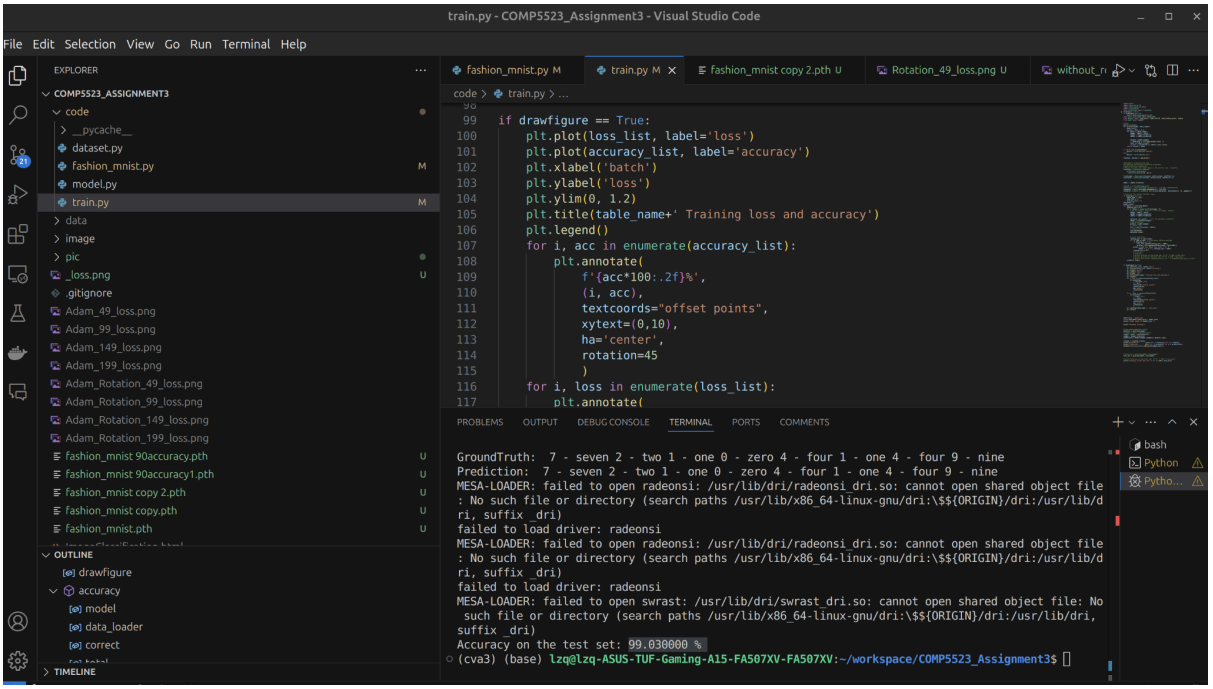
Adam computes adaptive learning rates for each parameter. It is generally considered to be more effective than SGD for many deep learning tasks.
The Adam optimizer outperforms the SGD optimizer in this case. **The final loss is lower, and the final accuracy is higher** when using the Adam optimizer.
This is due to the adaptive nature of the Adam optimizer, which adjusts the learning rate for each parameter based on the estimates of the first and second moments of the gradients. This can lead to faster convergence and better performance, especially for complex models or tasks with noisy gradients.

d: Set appropriate parameters (e.g., modify the optimizer, learning rate, epochs, etc.) to ensure that the model achieves an accuracy higher than 95% on the test set. Print the training accuracy, testing accuracy and learning rate every 1000 mini-batches. Save the model to the file `model.pth`. (2 points)

the final result reached 99.12%

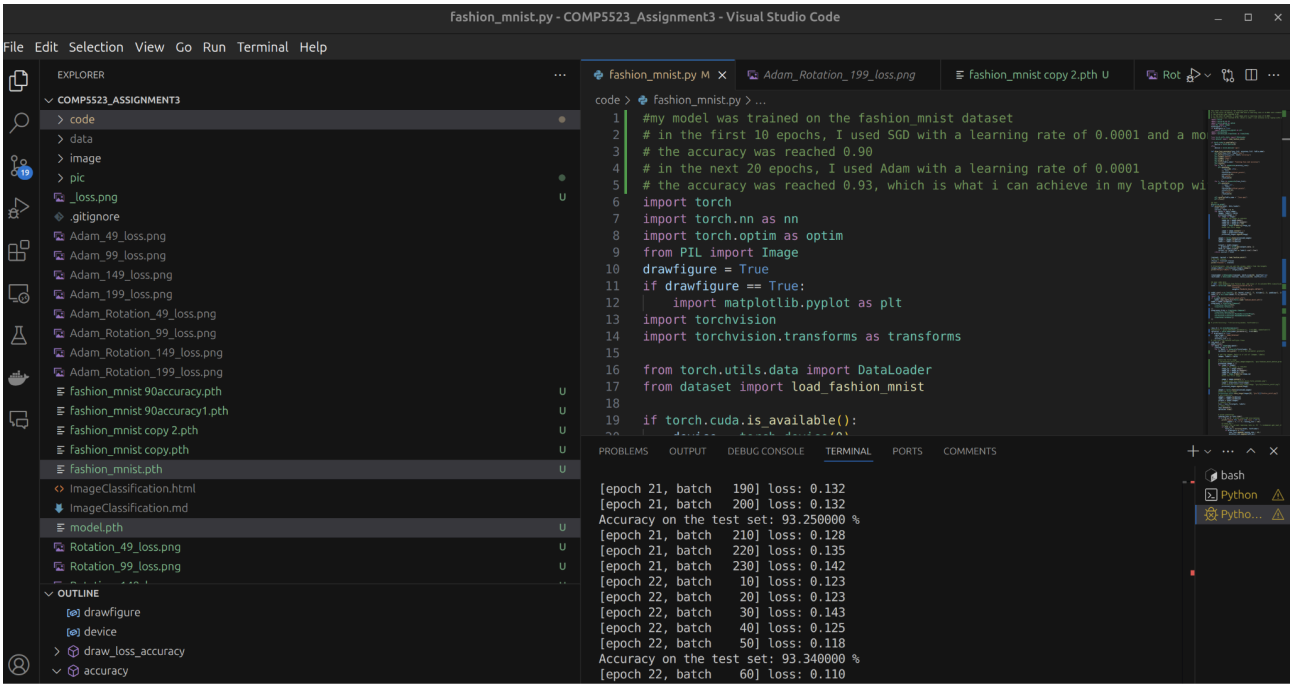


screenshot



Bonus

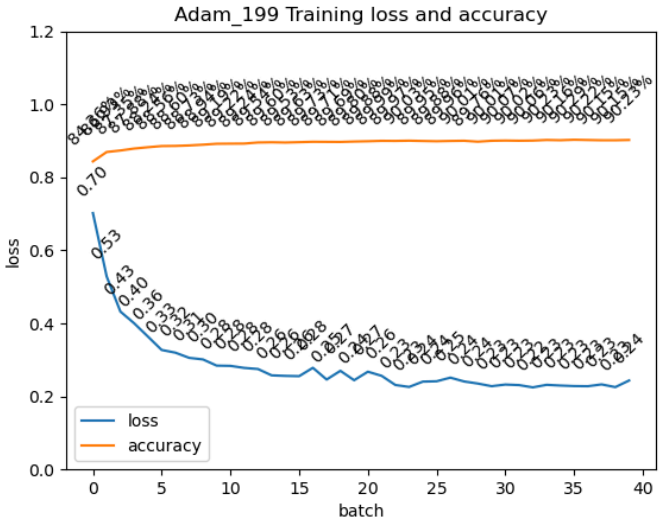
my model was trained on the fashion_mnist dataset, in the first 10 epochs, I used SGD with a learning rate of 0.0001 and a momentum of 0.9,the accuracy was reached 0.90, in the next 20 epochs, I used Adam with a learning rate of 0.0001, the accuracy was reached 0.93



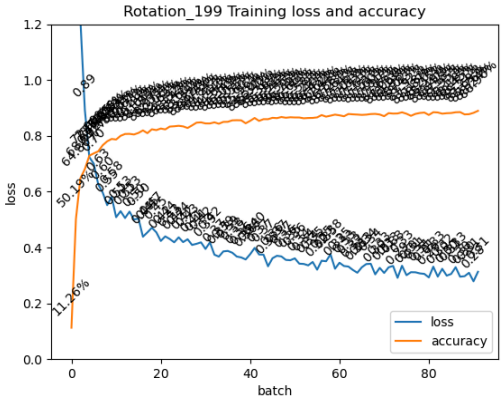
Bonus 2.2-2

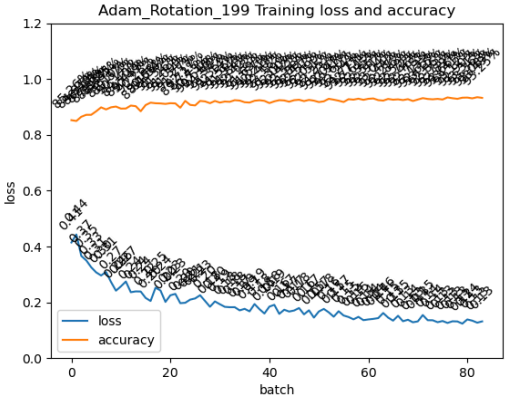
Experiment with data augmentation techniques such as `torchvision.transforms.RandomHorizontalFlip` and `torchvision.transforms.RandomRotation`, **discuss** the impact of these data augmentation parameters on the final

classification accuracy. (0.5 point)



without rotation





with rotation train phase1 and train phase 2

from the image we can find that, without random rotation and random mirror, the model can reach 0.90 in only 40 batches (256 image for each batch) however, with the rotation and mirror, we need 90(in phase1, above left)+20(in phase2, above right)=110 batches to reach similar accuracy. **So, the rotation and mirror will slow the training compared to similar accuracy.**

But with the rotation, the model have potential to be trained a better performance, With 60 more batches, the accuracy will reach 93%, and still have potential to reach higher if the following train would be applied.

generally, it can provide a more general model despite it requiring more training consumption.