Clab-2 Report

ENGN6528

Name

uxxxxxxx

dd/mm/yyyy

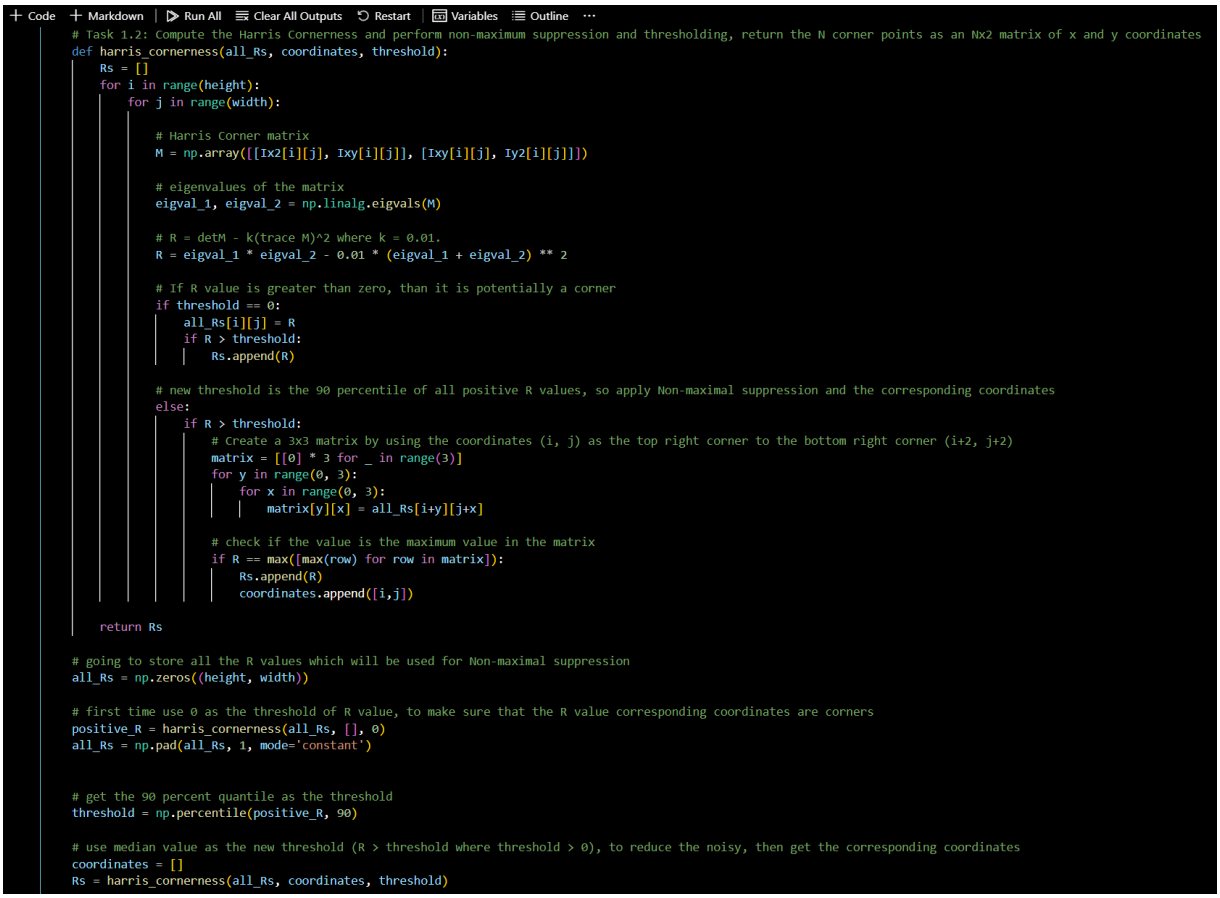
This is just a template for CLab-2. Please make sure you follow all the requirements in Clab-2 assignment file if it is not mentioned in this template.

You may need to refer to the assignment file for detailed requirements

Task 1: Harris Corner Detector (6 marks)

1. **Read and understand the corner detection code 'harris.m' in Fig 1.**

2. **Complete the missing parts, rewrite 'harris.m' (or harris.py) into a Matlab (or Python) function, and design appropriate function signature (1 mark).**



```
+ Code + Markdown | Run All  Clear All Outputs  Restart | Variables  Outline ···
# Task 1.2: Compute the Harris Cornerness and perform non-maximum suppression and thresholding, return the N corner points as an Nx2 matrix of x and y coordinates
def harris_cornerness(all_Rs, coordinates, threshold):
    Rs = []
    for i in range(height):
        for j in range(width):

            # Harris Corner matrix
            M = np.array([[Ix2[i][j], Ixy[i][j]], [Ixy[i][j], Iy2[i][j]]])

            # eigenvalues of the matrix
            eigval_1, eigval_2 = np.linalg.eigvals(M)

            # R = detM - k(trace M)^2 where k = 0.01.
            R = eigval_1 * eigval_2 - 0.01 * (eigval_1 + eigval_2) ** 2

            # If R value is greater than zero, than it is potentially a corner
            if threshold == 0:
                all_Rs[i][j] = R
                if R > threshold:
                    Rs.append(R)

            # new threshold is the 90 percentile of all positive R values, so apply Non-maximal suppression and the corresponding coordinates
            else:
                if R > threshold:
                    # Create a 3x3 matrix by using the coordinates (i, j) as the top right corner to the bottom right corner (i+2, j+2)
                    matrix = [[0] * 3 for _ in range(3)]
                    for y in range(0, 3):
                        for x in range(0, 3):
                            matrix[y][x] = all_Rs[i+y][j+x]

                    # check if the value is the maximum value in the matrix
                    if R == max([max(row) for row in matrix]):
                        Rs.append(R)
                        coordinates.append([i,j])

    return Rs

# going to store all the R values which will be used for Non-maximal suppression
all_Rs = np.zeros((height, width))

# first time use 0 as the threshold of R value, to make sure that the R value corresponding coordinates are corners
positive_R = harris_cornerness(all_Rs, [], 0)
all_Rs = np.pad(all_Rs, 1, mode='constant')


# get the 90 percent quantile as the threshold
threshold = np.percentile(positive_R, 90)

# use median value as the new threshold (R > threshold where threshold > 0), to reduce the noisy, then get the corresponding coordinates
coordinates = []
Rs = harris_cornerness(all_Rs, coordinates, threshold)
```

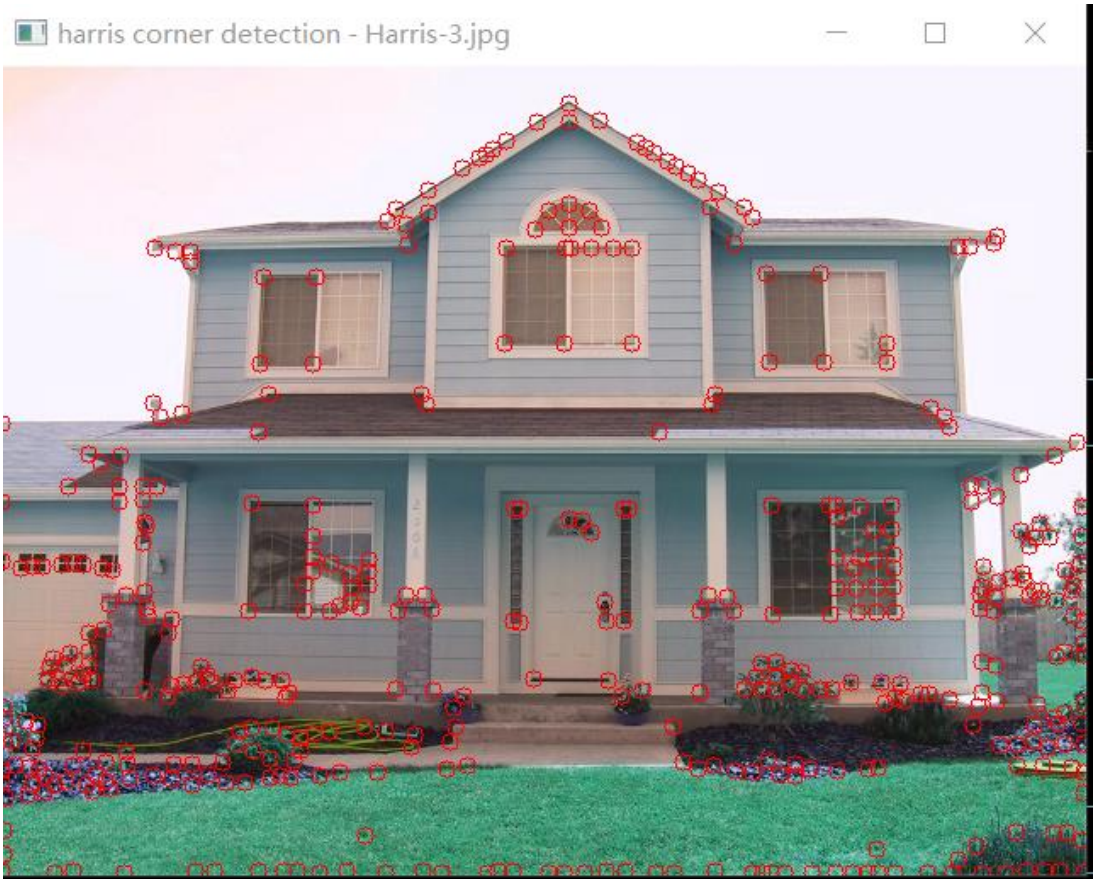
3. **Please provide comments on line … and every line of your solution after line … (0.5 mark). Specifically, you need to provide short comments on your code, which should make your code readable.**

```
# Task1.3
# generate Gaussian filter for a specific size, where height and width are [max(1, np.floor(3 * sigma) * 2 + 1)], [max(1, np.floor(3 * sigma) * 2 + 1)] respectively
# "np.floor(3 * sigma)" calculates the size of filter based on sigma,
# the max() function is to make sure the least size is 1,
# np.floor(3 * sigma) * 2 + 1) make sure the size is always odd
g = fspecial((max(1, np.floor(3 * sigma) * 2 + 1), max(1, np.floor(3 * sigma) * 2 + 1)), sigma)
Iy2 = conv2(np.power(Iy, 2), g)
Ix2 = conv2(np.power(Ix, 2), g)
Ixy = conv2(Ix * Iy, g)
```

The comments for block 7 is provided in task 1.2 screenshot.

4. **Test this function on the provided four test images (Harris-[1,2,3,4].jpg, they can be downloaded from Wattle). Display your results by marking the detected corners on the input images (using circles or crosses, etc) (0.5 mark for each image, (2 marks) in total).**


harris corner detection - Harris-2.jpg

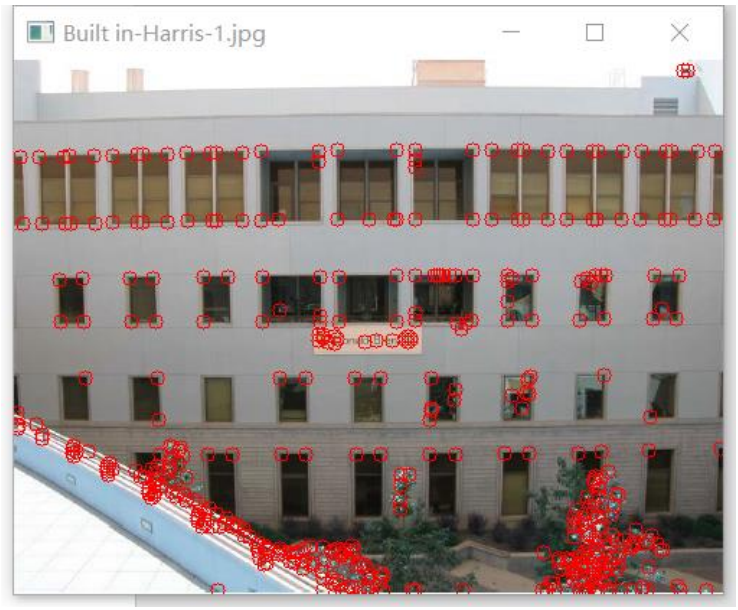

harris corner detection - Harris-3.jpg


harris corner - Harris-4.jpg

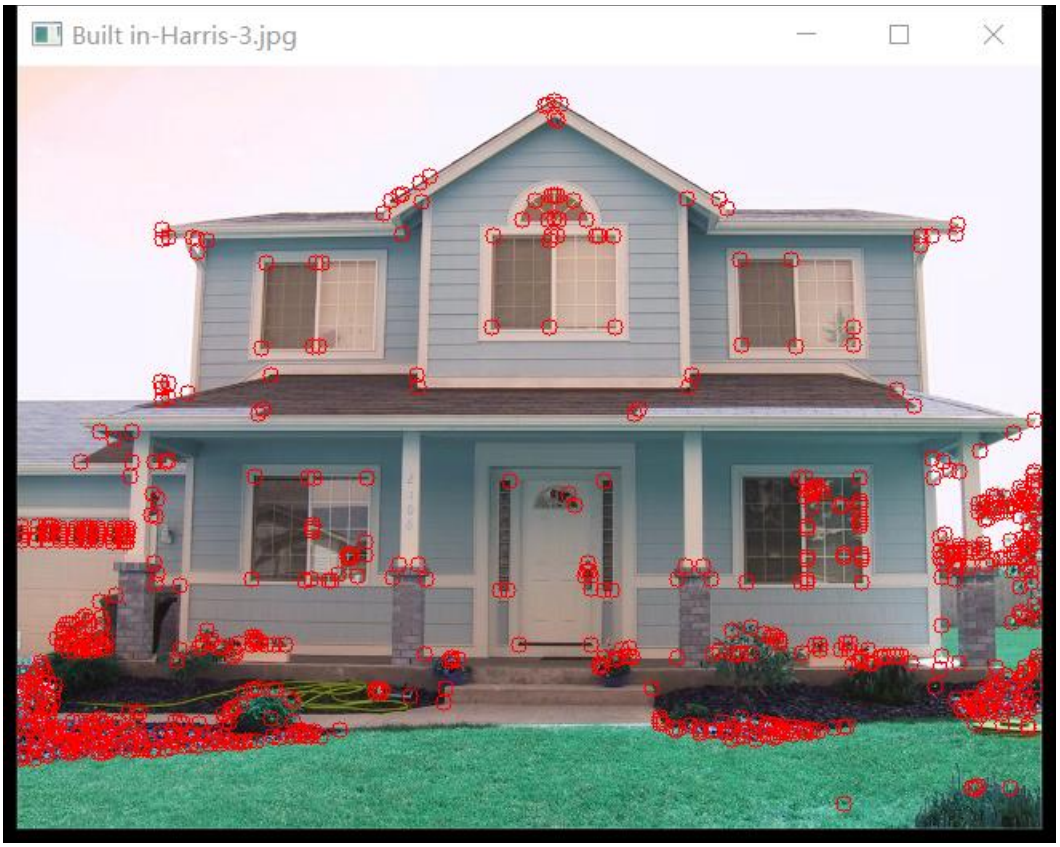
5. **Compare your results with that from built-in function used to detect corcers (0.5 mark), and discuss the factors that affect the performance of Harris corner detection (1 mark) .**

```python
import cv2
import numpy as np

# Task 1.5

# Convert to grayscale
if len(img.shape) == 3:
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
if len(img.shape) == 2:
    gray = img

height, width = gray.shape

# Apply Harris corner detection
block_size = 3
ksize = 3
k = 0.01
corners = cv2.cornerHarris(gray, block_size, ksize, k)

length = 0
# Threshold output
threshold = np.percentile(corners, 98)

for i in range(1, height-1):
    for j in range(1, width-1):
        if corners[i][j] > threshold:

            # Create a 3x3 matrix by using the coordinates (i, j) as the center corner, create matrix from (i-1, j-1) to (i+1, j+1)
            matrix = [[0] * 3 for _ in range(3)]
            for y in range(-1, 2, 1):
                for x in range(-1, 2, 1):
                    matrix[y][x] = corners[i+y][j+x]

            # check if the value is the maximum value in the matrix
            if corners[i][j] == max([max(row) for row in matrix]):
                img_cpy2 = cv2.circle(img_cpy2, (j, i), 5, (0, 0, 255), 1)
                length += 1

print(length)
# Display output
cv2.imshow('Built in-Harris-2.jpg', img_cpy2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The above code is my built-in function.




Built in-Harris-2.jpg


Built in-Harris-3.jpg

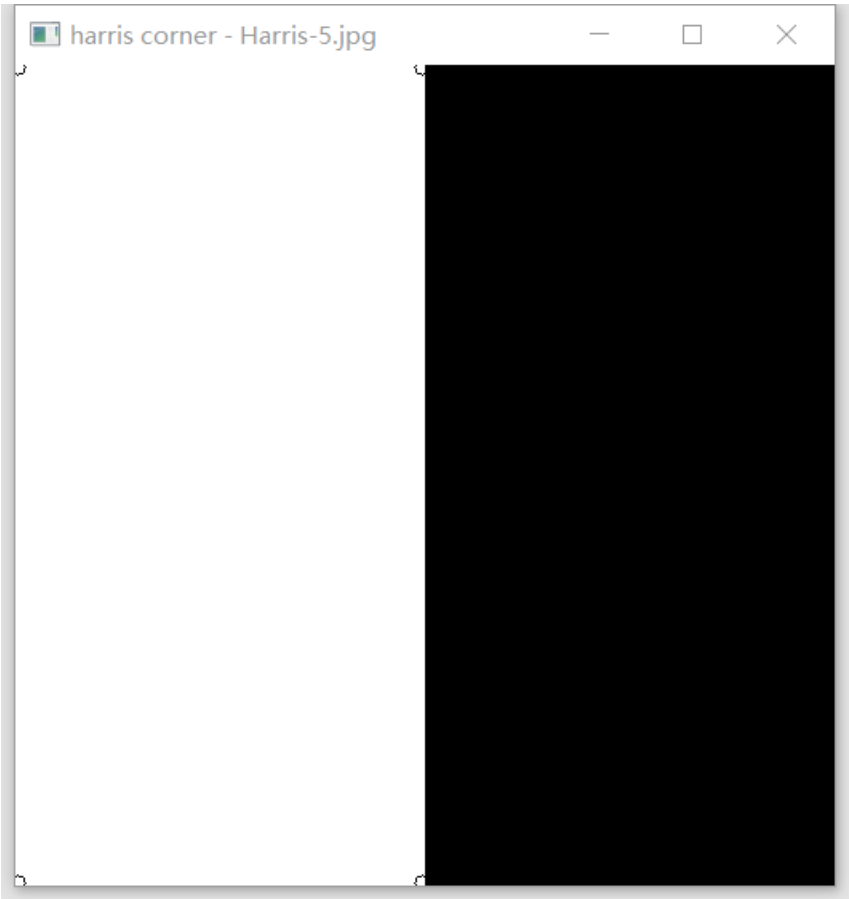
The built in function can actually detect more true corners than mine, but actually it also has more false corners to be detected as well.

Discuss at least two factors that affect the performances of Harris corner detection

Image noise could be one of the factor as the noise might be recognized as a corner by Harris corner detection.

Another factor is the threshold value. We need a suitable threshold to make sure that the edges will not be detected as corners, as well as avoid some true corners being discarded.

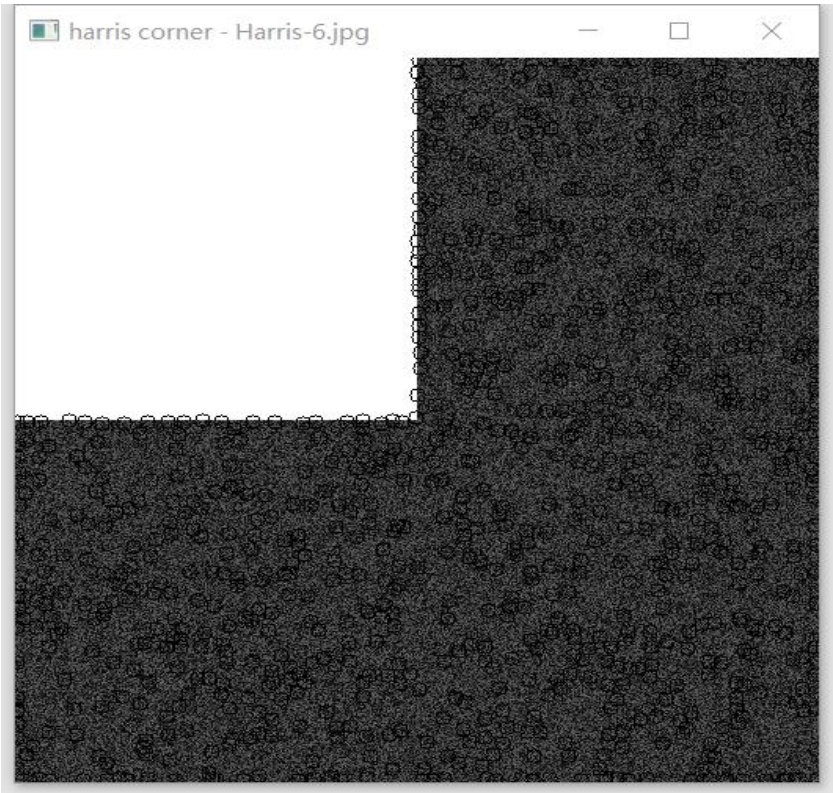
6. **Test your built function on 'Harris-5.jpg' (Harris-5.jpg can be downloaded from wattle.). Analyse the results why we cannot get corners by discussing and visualising your corner response scores of the image. (0.5mark)**



Analyse the results why we cannot get corners by discussing and visualising your corner response scores of the image (0.4 marks)

the maximum value of each row is zero, which probably means if value is too small, it will be eliminate by the in-built Harris Corner function. However, my function could still detect 4 corners as shown above, I guess it is probably because my threshold is set to the 90 percentiles of the all-positive data, so these points are being reserved as corners on the image.

7. **Test your built function on 'Harris-6.jpg' (Harris-6.jpg can be downloaded from wattle.). Plot your harris corner detector results in your report and propose a solution to obtain the salient corners which is robust to noise. (0.5mark)**



Propose a solution to obtain the salient corners which is robust to noise (0.4 marks)

Before applying the Harris corner detector, maybe pre-process the image using noise filtering techniques to reduce the noise in the image. This can improve the accuracy of the corner detection and make the resulting salient corners more robust to noise.

Task 2 - Deep Learning Classification (10 marks)

In this lab, we will train a CNN with the Fashion-MNIST using the PyTorch[1] deep learning frame- work. The Fashion-MNIST dataset contains 70000 images: 60000 training images and validation images, 10000 testing images[2] for fashion classification task. Note that we have 60000 training images with labels. You are required to split the 60k training data further by yourself to reserve 1000 images for validation (they should be from 10 classes). Images are of size 28×28 Greyscale, for 10 classes:

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

1. **Download the Fashion-MNIST dataset from the following git link**

2. **After loading the data using numpy, normalize the data to the range between (-1, 1). Also perform the following data augmentation when training (1 mark):**
   - **randomly flip the image left and right.**
   - **zero-pad 4 pixels on each side of the input image and randomly crop 28x28 as input to the network.**

```
train_transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize(0.5, 0.5), # normalize the data to the range (-1, 1)
        transforms.RandomRotation(20),
        transforms.RandomVerticalFlip(0.5),
        transforms.RandomHorizontalFlip(0.5), # the image will be flipped with 50% probability left and right
        transforms.Pad(padding=4, fill=0, padding_mode='constant'), # zero padded 4 pixels
        transforms.RandomCrop(size=28), #randomly crop 28x28 as input
        ])
```

3. **Build a CNN with the following architecture (1 mark):**
    - **5×5 Convolutional Layer with 32 filters, stride 1 and padding 2.**
    - **ReLU Activation Layer.**
    - **2×2 Max Pooling Layer with a stride of 2.**
    - **3×3 Convolutional Layer with 64 filters, stride 1 and padding 1.**
    - **ReLU Activation Layer.**
    - **2×2 Max Pooling Layer with a stride of 2.**
    - **Fully-connected layer with 1024 output units.**
    - **ReLU Activation Layer.**
    - **Fully-connected layer with 10 output units.**

```
[24] import torch.nn as nn
     import torch.nn.functional as F

     class MyModel(nn.Module):    # inherit from nn.Module class
             def __init__(self):
                     super().__init__()
                     self.conv1 = nn.Conv2d(in_channels=1, out_channels=32, kernel_size=5, stride=1, padding=2) # stride means how many pixels need to move
                     self.relu = nn.ReLU()
                     self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2)
                     self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=1) # outchannel in conv1 is the inchannel in conv2
                     self.relu = nn.ReLU()
                     self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)
                     self.fc1 = nn.Linear(in_features=64*7*7, out_features=1024)
                     self.relu = nn.ReLU()
                     self.fc2 = nn.Linear(in_features=1024, out_features=10)


             def forward(self, x):
                     # x shape [batch_size, channel, height, width]   [B, 3, 32, 32]
                     x = self.conv1(x)
                     x = F.relu(x)
                     x = self.pool1(x)
                     x = self.conv2(x)
                     x = F.relu(x)
                     x = self.pool2(x) #[1024, 64, 7, 7]
                     x = x.view(-1, 64 * 7 * 7)
                     x = self.fc1(x)
                     x = F.relu(x)
                     x = self.fc2(x)
                     return x

     model = MyModel() # instantiate a model
     print(model)
```

4. **Set up cross-entropy loss. (0.5 mark)**

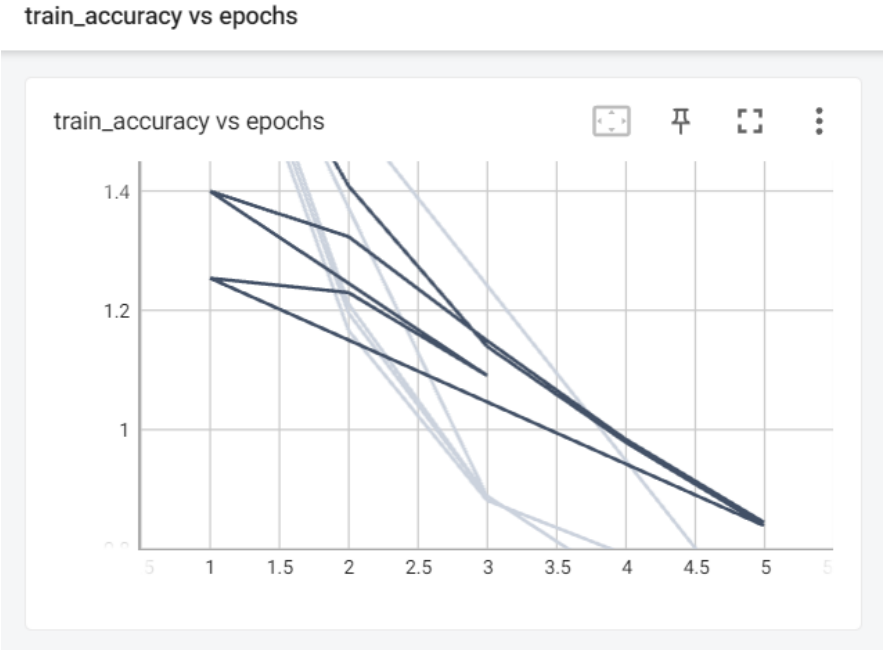
    loss_fn = nn.CrossEntropyLoss()

5. **Set up Adam optimizer, with 1e-3 learning rate and betas=(0.9, 0.999). (0.5 mark)**

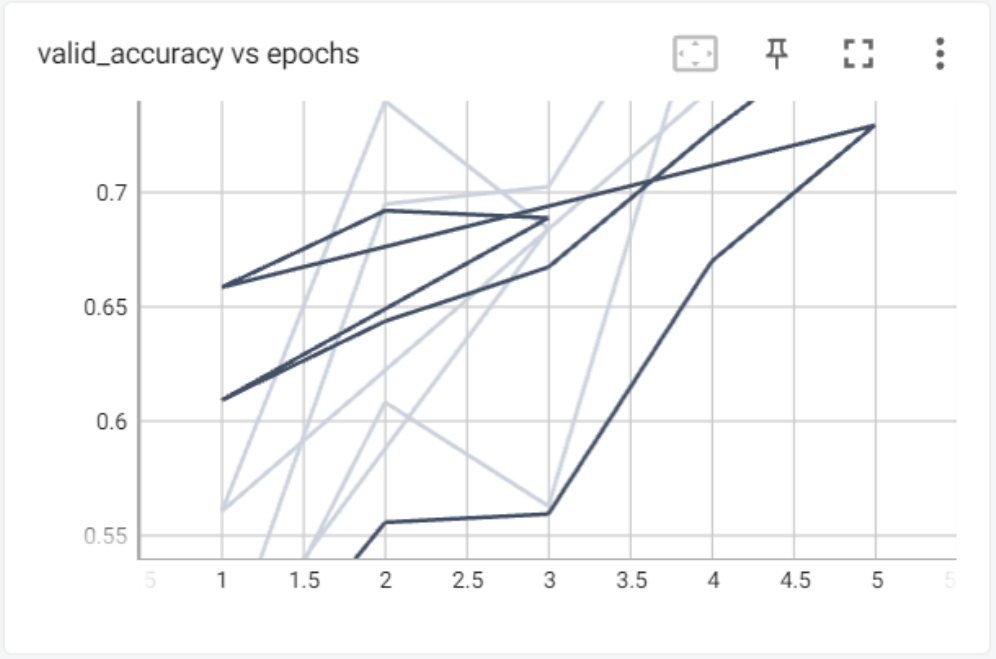
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate, betas=(0.9, 0.999))

6. **Train your model. Draw the following plots:**
   - **Training loss vs. epochs.**
   - **Training accuracy vs. epochs.**
   - **Validation loss vs. epochs.**
   - **Validation accuracy vs. epochs.**

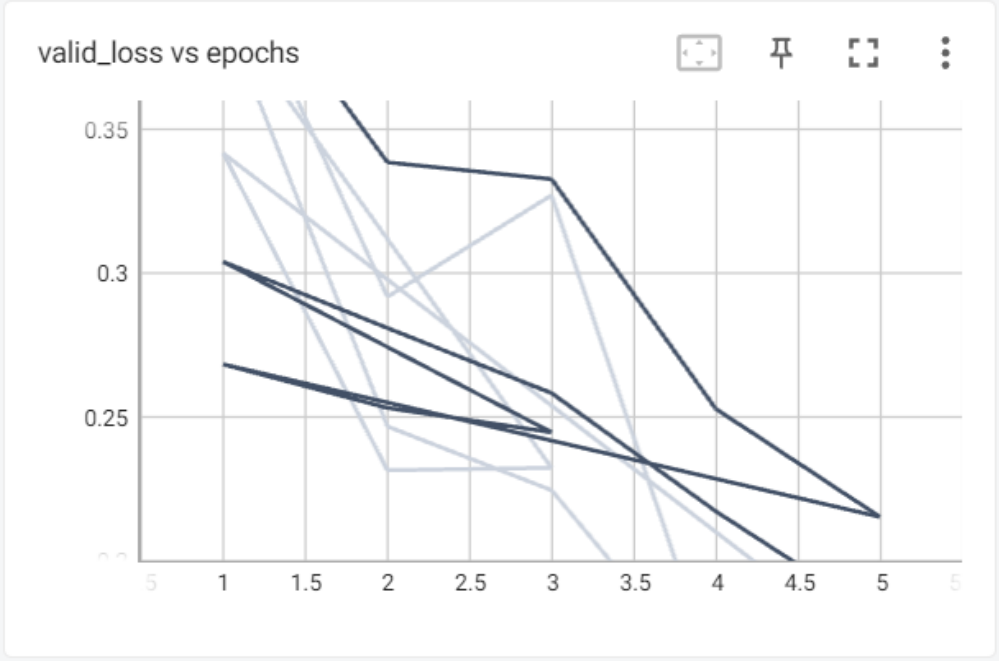
   You can either use Tensorboard to draw the plots or you can save the data (e.g. in a dictio- nary) then use Matplotlib to plot the curve. (2 marks)



valid_accuracy vs epochs



valid_loss vs epochs



7. **Train a good model. Marks will be awarded for high performance and high efficiency in training time and parameters (there may be a trade-off), good design, and your discussion. You are not allowed to use a pre-trained model, you should train the model yourself. You need to describe what exactly you did to the base model to improve your results in your report, and your motivation for your approach (no more that 1 page of text). Please include plots as above for training and validation loss and accuracy vs. epochs, as well as the final accuracy on the test set. Please submit the code and your trained model for this. Your performance will be verified. Please ensure you follow the test/train splits as provided. You also must show your training and validation accuracy vs. epochs for this model in your report. Note that you may be asked to run training of your model to demonstrate its training and performance. (4 marks)**

   # Display the accuracy of your modified model with reasonable interpretations/ motivations and four plots as required (3 mark)

   # Good design/ high performance/ high model efficiency/ good discussion (1 mark)

   (Listing your modifications and corresponding motivations by dots are preferred)

   I have increased the number of filters from 32 to 64 for the first convolution layer (64 to 128 for the second convolution layer).

   I have added a Batch Normalization which allows me to use much higher learning rates, which further increases the speed at which networks train.

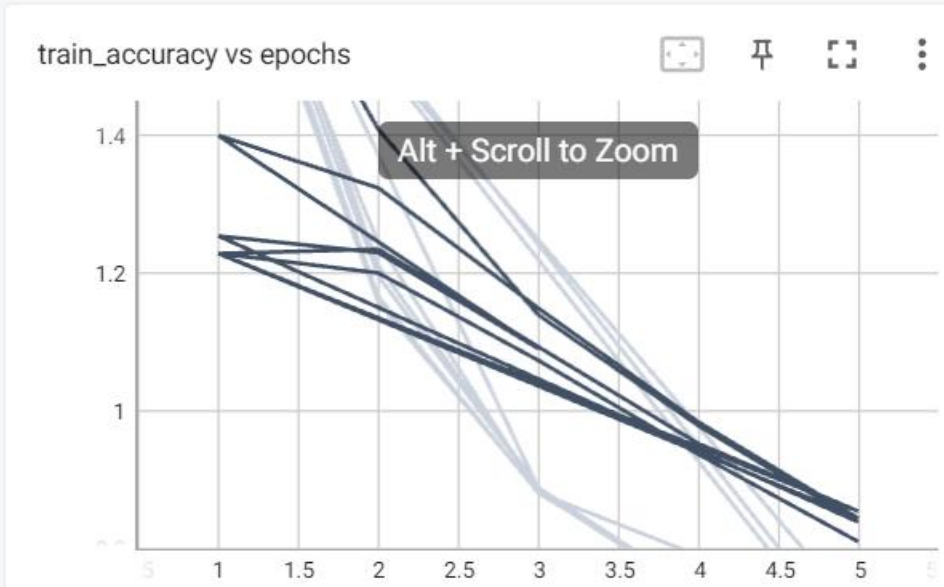
   I have increased the number of fully connected layers in a neural network can potentially lead to an increase in the model's capacity to learn more complex patterns and representations from the input data.

   I have used dropout to the fully connected layers in order to help in reducing overfitting.

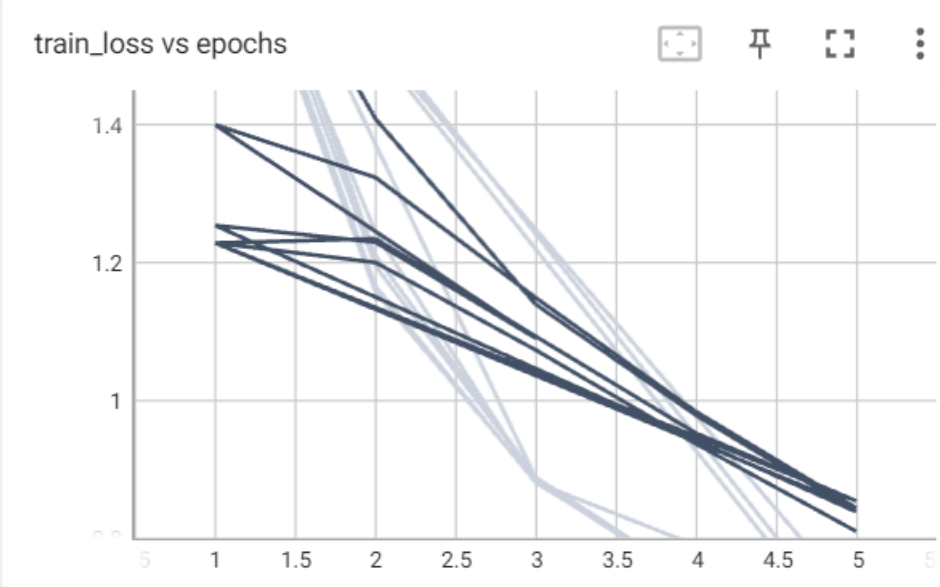
   As a result, the accuracy of my second model for the first Epoch is increased by more than 10%, but the accuracy of first fifth Epoch is just the same as the first model (see the graphs below). Due to the time limitation, I have no time to train maybe 10 or more epochs, but I think it can increase the performance than the previous model.

```
evaluating data...
Test Error:
   Accuracy: 54.09, Avg loss: 0.3357

current epoch is:  2
training data...
evaluating data...
Test Error:
   Accuracy: 66.64, Avg loss: 0.2698

current epoch is:  3
training data...
evaluating data...
Test Error:
   Accuracy: 76.23, Avg loss: 0.2037

current epoch is:  4
training data...
evaluating data...
Test Error:
   Accuracy: 71.19, Avg loss: 0.2367

current epoch is:  5
training data...
evaluating data...
Test Error:
   Accuracy: 84.68, Avg loss: 0.1300

Finished Training.
```

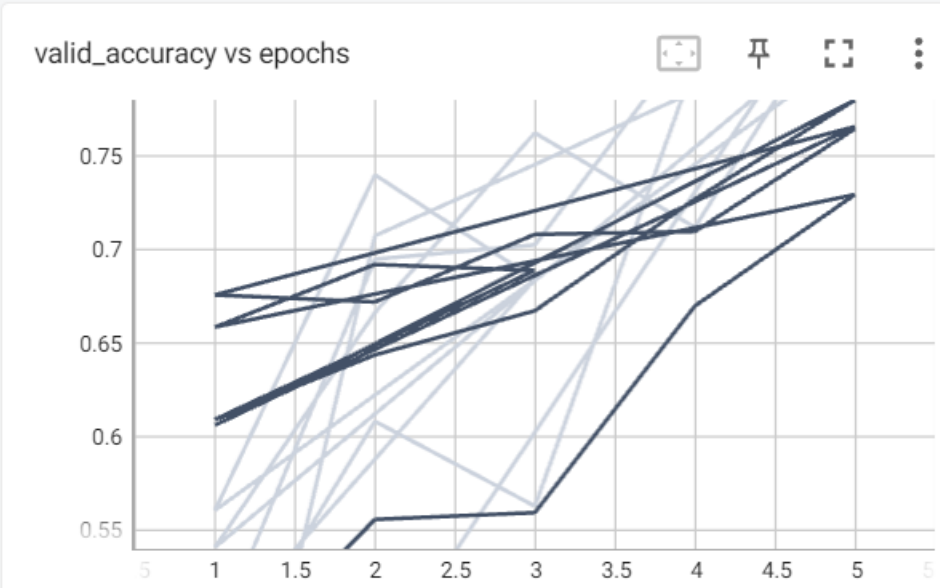
train_accuracy vs epochs



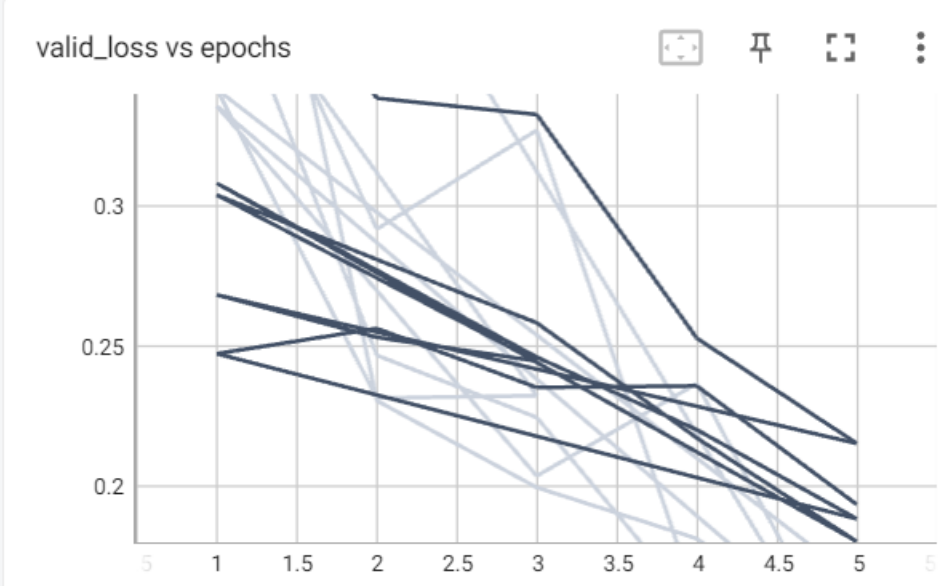
train_loss vs epochs



valid_accuracy vs epochs



valid_loss vs epochs



8. **The main dataset site on github (https://github.com/zalandoresearch/fashion-mnist) in- cludes a series of results for other network models (under Benchmark). How does your model compare? Explain why the ResNet18 model may produce better results than yours (or the other way around if this is the case).(1 mark).**

   # Compare your model with others. Also provide a comparison in words (0.2 marks)
   My model has a similar accuracy to the other models.

   # Discuss why the ResNet18 model may produce better results than yours (or the other way around if this is the case) (0.8 marks)

   ResNet18 model has a better accuracy than mine.
   From the internet, ResNet18 has 18 layers and consists of several convolutional layers, followed by a global average pooling layer, and then a fully connected layer for the final classification. The key innovation of ResNet18 is the use of residual blocks, which are designed to address the problem of vanishing gradients in deep neural networks.

   From my perspective, ResNet18 has a higher accuracy is mainly because of the residual connections. The residual connections in ResNet18 allow the gradient to flow directly through the network, bypassing multiple layers, which helps to prevent the gradients from becoming too small. This makes it easier for the model to learn, and allows it to train much deeper networks (18 layers) than other models like mine.