Assignment 1

Adnan Altukleh
Course: DV2586

Part 1:

Q-1) Consider the following function with two parameters:

$$f(x,y) = x^2 + y^2 + x(y+2) + \cos(3x)$$

a) Take the derivative of the function with respect to x

$2x + y + 2 - 3sin(3x)$

b) Take the derivative of the function with respect to y

$2y + x$

c) Implement and use gradient descent and find the minimum of this function.
You can use Python.

min_x: -1.088156592554671 min_y: 0.5440286008422788 functions_value: -2.2807097055009007

d) Plot the gradient descent results.

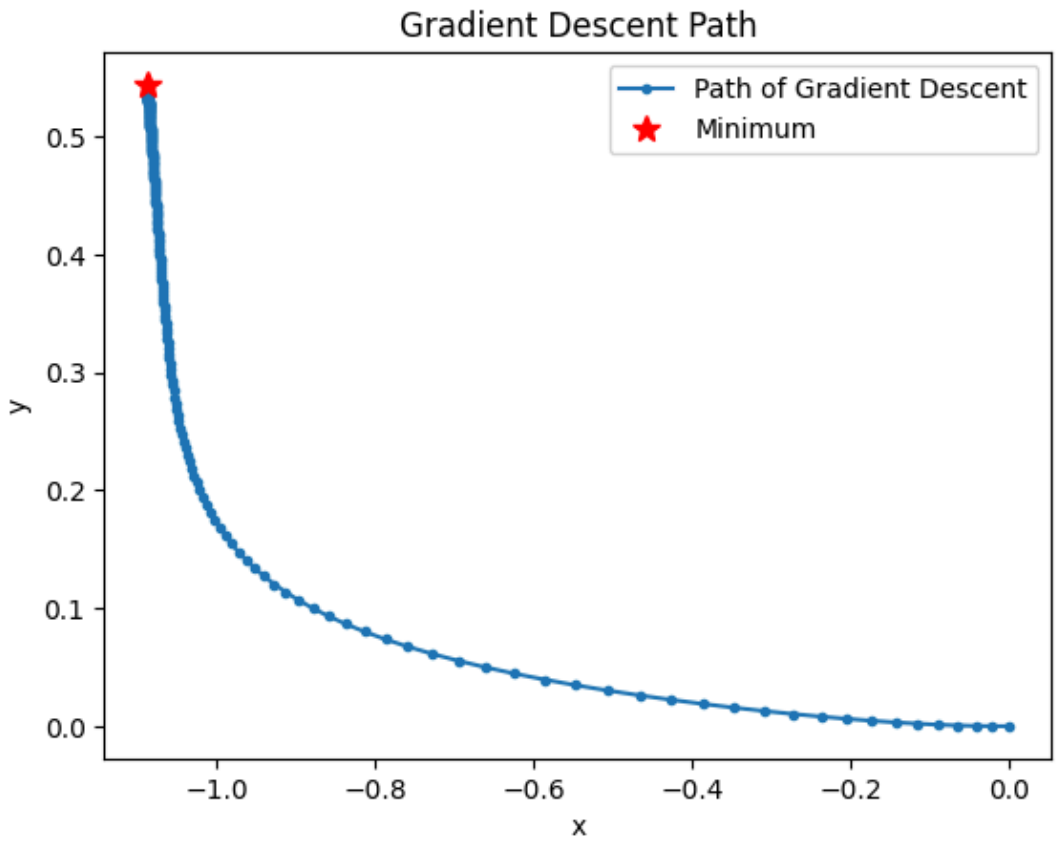

figure 1: the gradient descent results

Part 2:

Models structure:

VGG19:

For this assignment, I am building the VGG19 in the file vgg19.py and training it, instead of using a pre-trained model. In the training process there is some customization that can be applied. Ex: It allows customization such as specifying whether to include a top layer (which consists of fully connected layers), specifying weights, and adjusting input configurations. The model architecture consists of sequential convolutional blocks.

ResNet-50:

In my Resnet-50 experiment, I am using a pre-trained Resnet-50 model. I am not including the fully connected layers nor the weights that are from imagenet. Instead, I choose to create my own fully connected layers and specify the input shape to 32x32x3.

DenseNet-121:

The DenseNet-121 experiment is also a pre-trained model similar to the Resnet-50 i am not including the fully connected layers nor weights.

My Model:

My model is created in the file my_model.py and it is designed to extract as many useful features from images as possible. The setup includes groups, or "blocks," where each block is made up of pairs of layers that process the image data. Each of these pairs includes two layers, and after processing, the outputs from each pair are stuck together "concatenated." This approach helps the model capture a broader range of information from the image by combining different details from multiple pairs of layers.

To prevent overfitting, I am using techniques like average pooling and batch normalization. Average pooling reduces the amount of information the model deals with, allowing it to focus only on the most important parts. Batch normalization makes the data more uniform, which stabilizes the learning process and enhances the overall performance of the model.

The architecture starts and ends with max pooling, which focuses the model's attention on the most crucial features of the image by reducing the size of the image. This simplification helps the model identify patterns more effectively.

At the end of the model, there are two fully connected layers. Each of these layers is followed by a dropout layer to reduce overfitting. Due to limited computational resources, the images were resized to (32x32).

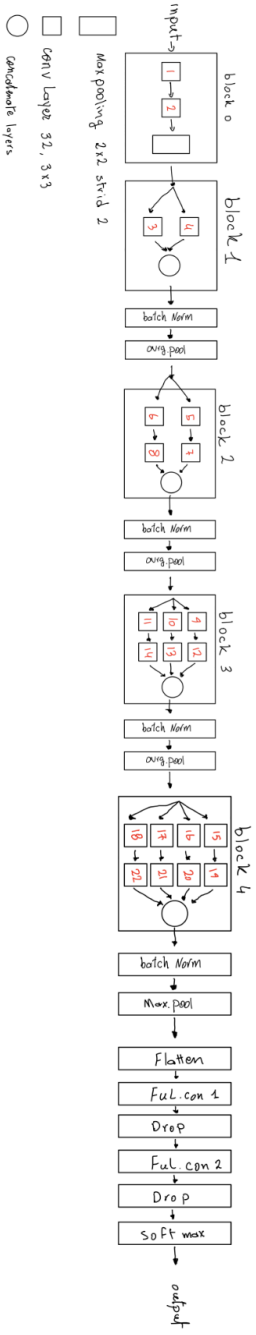

figure 2: The customized model architecture

Results:

VGG19:

```
Epoch 1/5
1265/1265 - 695s - 549ms/step - accuracy: 0.8700 - loss: 0.3743 - val_accuracy: 0.9716 - val_loss: 0.1027
Epoch 2/5
1265/1265 - 692s - 547ms/step - accuracy: 0.9832 - loss: 0.0617 - val_accuracy: 0.9826 - val_loss: 0.0686
Epoch 3/5
1265/1265 - 694s - 549ms/step - accuracy: 0.9894 - loss: 0.0426 - val_accuracy: 0.9814 - val_loss: 0.0740
Epoch 4/5
1265/1265 - 691s - 546ms/step - accuracy: 0.9908 - loss: 0.0378 - val_accuracy: 0.9791 - val_loss: 0.0811
Epoch 5/5
1265/1265 - 689s - 545ms/step - accuracy: 0.9931 - loss: 0.0296 - val_accuracy: 0.9831 - val_loss: 0.0714
Epoch 5: early stopping
Restoring model weights from the end of the best epoch: 2.
```

figure 3: Accuracy and Loss Metrics Across Epochs

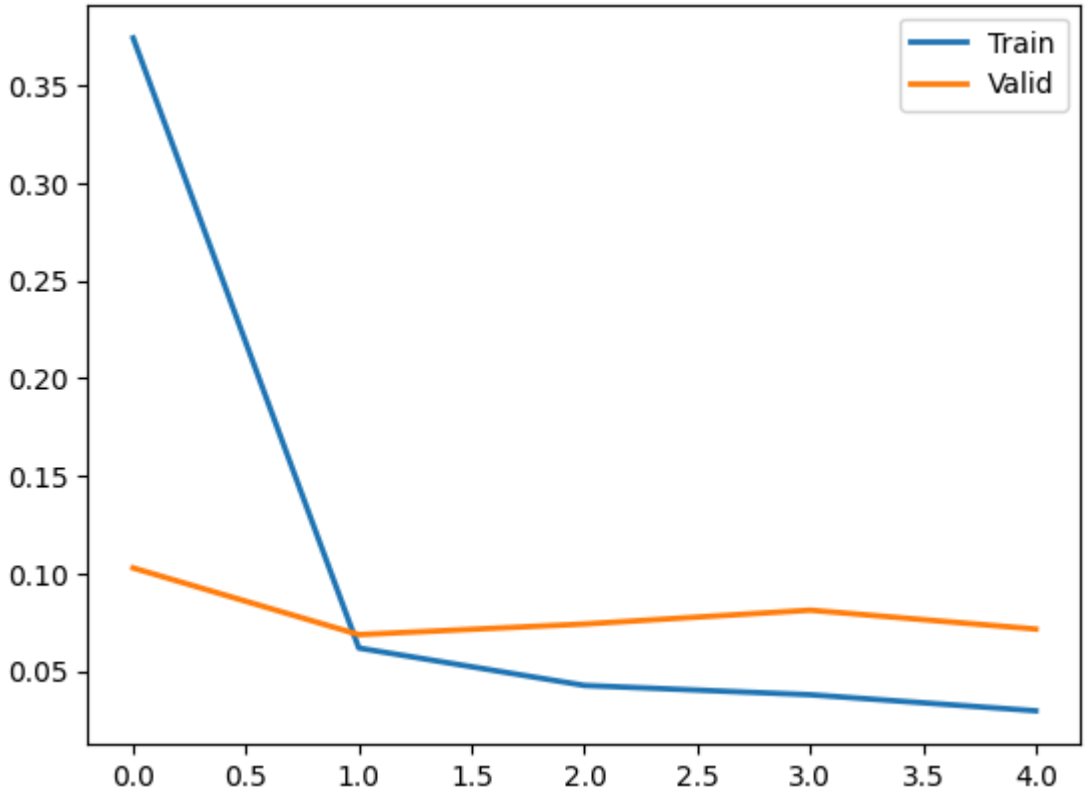

figure 4: Convergence of Training and Validation Loss During Model Training

Classification repot for VGG19:

	precision	recall	f1-score	support
0	1.00	0.99	0.99	5008
1	0.99	0.98	0.99	5001
2	0.99	0.99	0.99	5031
3	0.99	0.99	0.99	5010
4	0.99	0.99	0.99	5082
5	0.97	1.00	0.98	5138
6	0.99	0.98	0.99	5142
7	0.97	0.99	0.98	5069
8	0.99	0.98	0.98	5073
9	0.98	0.98	0.98	5022
accuracy			0.99	50576
macro avg	0.99	0.99	0.99	50576
weighted avg	0.99	0.99	0.99	50576

figure 5: Classification report for VGG-19

```
[[4959,    3,    1,    5,    1,    3,    7,    2,   16,   11],
 [   9, 4899,   10,    2,   21,    0,   14,   37,    2,    7],
 [   1,    0, 4975,    6,    2,    4,    0,   11,    7,   25],
 [   0,    0,    4, 4946,    8,   24,    0,   10,    4,   14],
 [   3,   12,    5,    1, 5018,    5,    4,   16,    2,   16],
 [   0,    0,    0,   14,    0, 5116,    3,    1,    3,    1],
 [   4,    5,    0,    0,    0,   74, 5045,    2,   12,    0],
 [   1,    6,    1,    1,    5,    7,    4, 5030,    3,   11],
 [   2,    0,   26,   23,    3,   44,    7,    5, 4954,    9],
 [   1,    2,   16,    4,    3,    2,    1,   73,    1, 4919]]
```

figure 6: Confusion Matrix for VGG-19

ResNet-50:

```
Epoch 1/5
1265/1265 - 510s - 403ms/step - accuracy: 0.8582 - loss: 0.4656 - val_accuracy: 0.8881 - val_loss: 0.3523
Epoch 2/5
1265/1265 - 495s - 392ms/step - accuracy: 0.9732 - loss: 0.0880 - val_accuracy: 0.9438 - val_loss: 0.1815
Epoch 3/5
1265/1265 - 504s - 398ms/step - accuracy: 0.9802 - loss: 0.0684 - val_accuracy: 0.9619 - val_loss: 0.1363
Epoch 4/5
1265/1265 - 514s - 407ms/step - accuracy: 0.9840 - loss: 0.0572 - val_accuracy: 0.9108 - val_loss: 0.3155
Epoch 5/5
1265/1265 - 495s - 391ms/step - accuracy: 0.9852 - loss: 0.0534 - val_accuracy: 0.9431 - val_loss: 0.2293
```

figure 7: Accuracy and Loss Metrics Across Epochs for ResNet-50

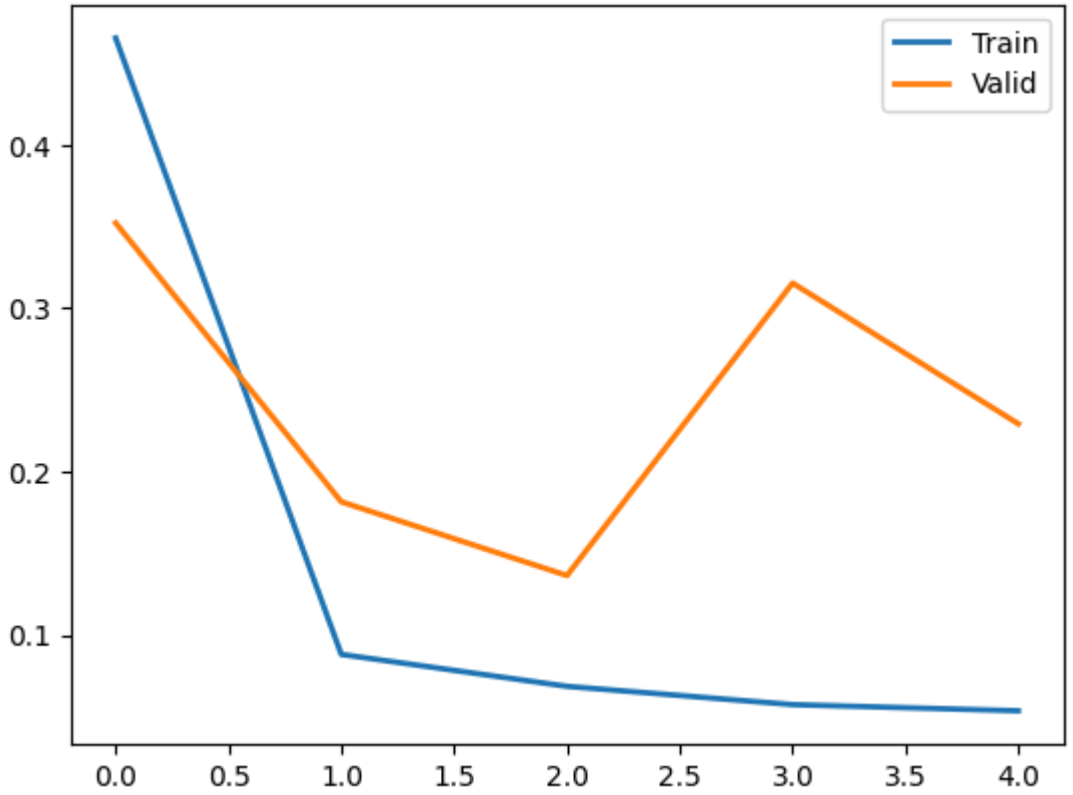

figure 8: Convergence of Training and Validation Loss During Model Training for ResNet-50

Classification repot for ResNet50:

	precision	recall	f1-score	support
0	0.99	0.98	0.98	5008
1	0.96	0.97	0.97	5001
2	1.00	0.87	0.93	5031
3	0.98	0.90	0.94	5010
4	0.94	0.98	0.96	5082
5	0.97	0.98	0.98	5138
6	0.95	0.99	0.97	5142
7	0.90	0.97	0.94	5069
8	0.98	0.92	0.95	5073
9	0.90	0.97	0.93	5022
accuracy			0.95	50576
macro avg	0.96	0.95	0.95	50576
weighted avg	0.96	0.95	0.95	50576

figure 9: Classification report for ResNet-50

```
[[4910,   15,    0,    3,   10,    5,   14,   15,    7,   29],
 [   7, 4873,    7,    0,   36,    1,   26,   45,    0,    6],
 [   4,   40, 4374,    9,  113,    8,   19,  122,   77,  265],
 [  14,   12,    1, 4524,   87,   69,    4,  151,    9,  139],
 [   1,   20,    2,    0, 4993,    3,    4,   45,    4,   10],
 [   0,   12,    0,   26,    7, 5035,   24,   18,    4,   12],
 [  23,   10,    0,    0,    1,   28, 5077,    1,    2,    0],
 [   0,   38,    0,    0,   21,    4,   21, 4938,    2,   45],
 [  20,   24,    1,   57,   38,   26,  166,   51, 4672,   18],
 [   5,   13,    3,    7,   29,    1,    2,  103,    9, 4850]]
```

figure 10: Confusion Matrix for ResNet-50

DenseNet-121:

```
Epoch 1/5
1265/1265 - 319s - 252ms/step - accuracy: 0.9314 - loss: 0.2119 - val_accuracy: 0.9344 - val_loss: 0.2089
Epoch 2/5
1265/1265 - 285s - 225ms/step - accuracy: 0.9833 - loss: 0.0579 - val_accuracy: 0.9585 - val_loss: 0.1429
Epoch 3/5
1265/1265 - 282s - 223ms/step - accuracy: 0.9876 - loss: 0.0434 - val_accuracy: 0.9275 - val_loss: 0.2333
Epoch 4/5
1265/1265 - 287s - 227ms/step - accuracy: 0.9895 - loss: 0.0374 - val_accuracy: 0.9735 - val_loss: 0.0977
Epoch 5/5
1265/1265 - 287s - 227ms/step - accuracy: 0.9912 - loss: 0.0310 - val_accuracy: 0.9779 - val_loss: 0.0854
```

figure 11: Accuracy and Loss Metrics Across Epochs for Deneneet-121

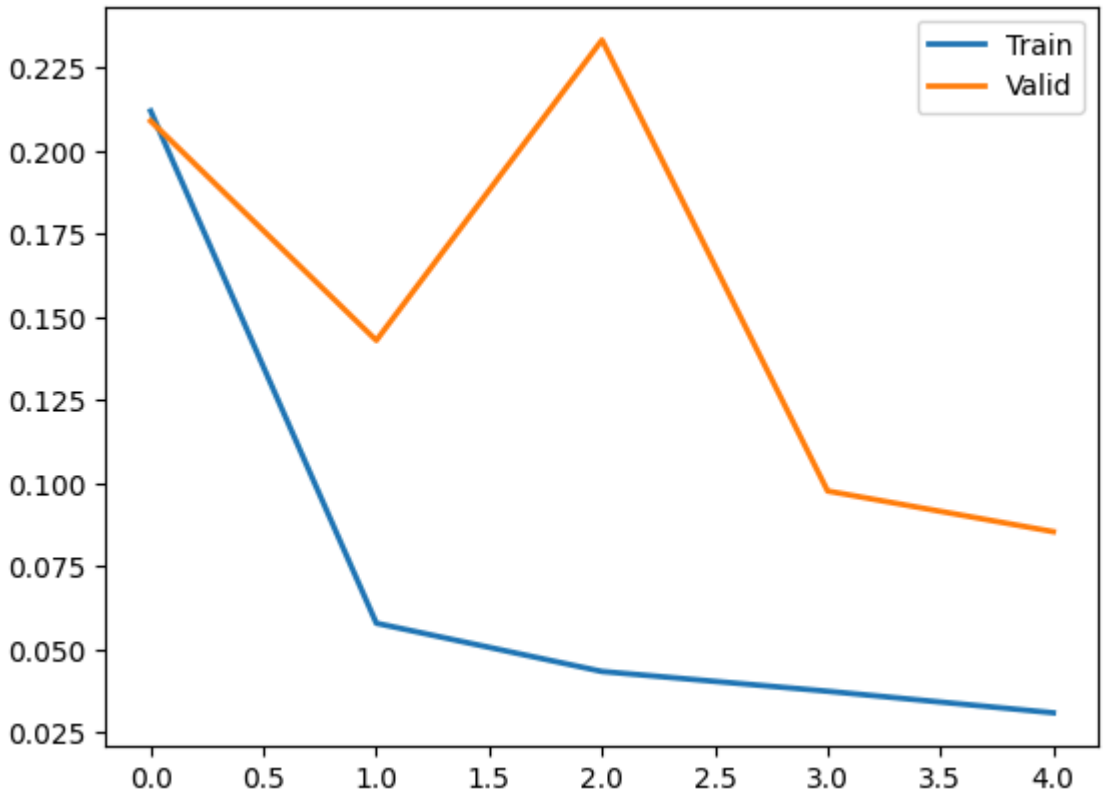

figure 12: Convergence of Training and Validation Loss During Model Training for DeseNet-121

```
Classification repot for DenseNet_121:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      5008
           1       0.98      0.99      0.99      5001
           2       0.99      0.97      0.98      5031
           3       1.00      0.94      0.97      5010
           4       0.99      0.99      0.99      5082
           5       0.98      0.99      0.99      5138
           6       0.99      0.99      0.99      5142
           7       0.97      0.99      0.98      5069
           8       0.99      0.99      0.99      5073
           9       0.93      0.99      0.96      5022

    accuracy                           0.98     50576
   macro avg       0.98      0.98      0.98     50576
weighted avg       0.98      0.98      0.98     50576
```

figure 13: Classification report for DeseNet-121

```
[[4928,    4,    2,    0,    2,    2,   16,    1,   10,   43],
 [   4, 4932,    8,    0,   27,    0,   13,   12,    2,    3],
 [   0,    8, 4888,    1,    1,    1,    1,   18,    4,  109],
 [   2,   14,   10, 4732,    5,   48,    1,   49,   25,  124],
 [   0,   11,    7,    0, 5014,    4,    3,   17,    5,   21],
 [   0,    4,    0,    5,    2, 5099,   15,    4,    1,    8],
 [   4,    2,    0,    0,    1,   15, 5110,    1,    9,    0],
 [   1,   21,    0,    1,    8,    1,    6, 5007,    3,   21],
 [   3,    1,    7,    2,    0,   16,    6,    3, 5015,   20],
 [   1,   12,    2,    0,    2,    0,    0,   31,    4, 4970]]
```

figure 14: Confusion Matrix for DeseNet-121

My Model:

```
Epoch 1/10
1265/1265 - 117s - 92ms/step - accuracy: 0.8611 - loss: 0.4104 - val_accuracy: 0.9044 - val_loss: 0.2985
Epoch 2/10
1265/1265 - 113s - 89ms/step - accuracy: 0.9650 - loss: 0.1095 - val_accuracy: 0.9404 - val_loss: 0.1845
Epoch 3/10
1265/1265 - 161s - 127ms/step - accuracy: 0.9764 - loss: 0.0736 - val_accuracy: 0.8751 - val_loss: 0.3953
Epoch 4/10
1265/1265 - 165s - 131ms/step - accuracy: 0.9822 - loss: 0.0557 - val_accuracy: 0.9647 - val_loss: 0.1127
Epoch 5/10
1265/1265 - 113s - 89ms/step - accuracy: 0.9864 - loss: 0.0430 - val_accuracy: 0.9740 - val_loss: 0.0868
Epoch 6/10
1265/1265 - 113s - 89ms/step - accuracy: 0.9892 - loss: 0.0347 - val_accuracy: 0.9724 - val_loss: 0.0908
Epoch 7/10
1265/1265 - 114s - 90ms/step - accuracy: 0.9905 - loss: 0.0296 - val_accuracy: 0.9710 - val_loss: 0.1009
Epoch 8/10
1265/1265 - 113s - 89ms/step - accuracy: 0.9920 - loss: 0.0244 - val_accuracy: 0.9742 - val_loss: 0.0918
Epoch 8: early stopping
Restoring model weights from the end of the best epoch: 5.
```

figure 15: Accuracy and Loss Metrics Across Epochs for My Model

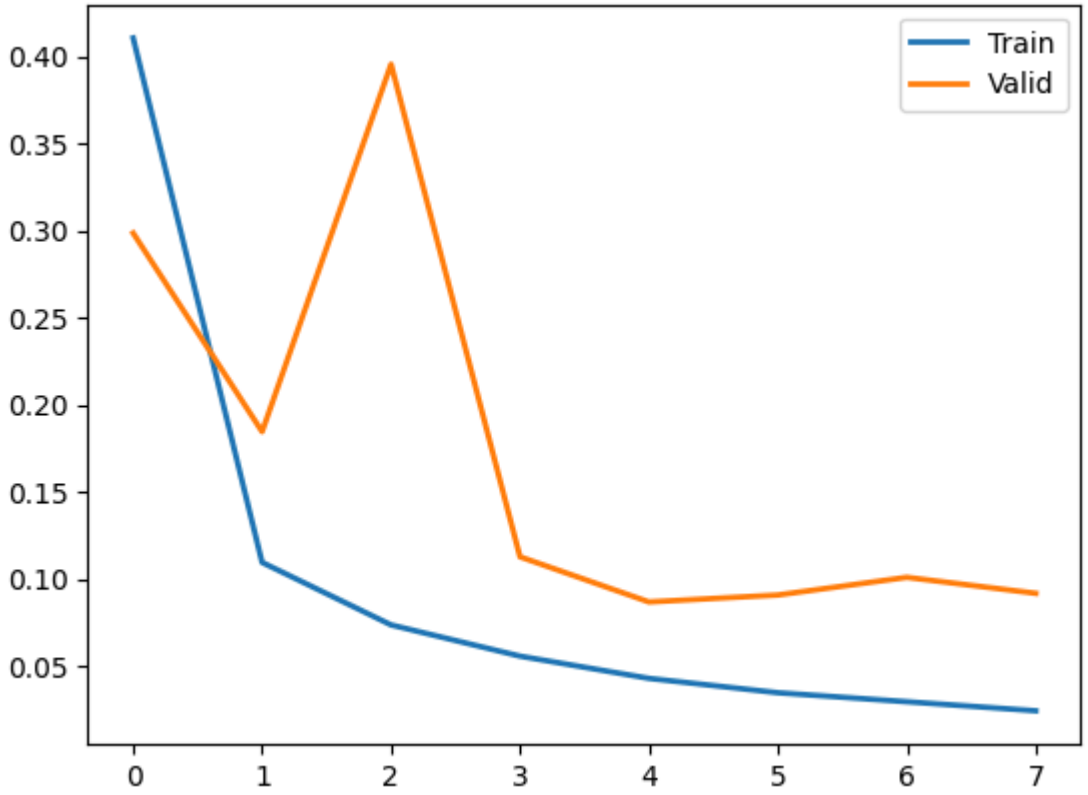

figure 16: Convergence of Training and Validation Loss During Model Training for My Model

Classification repot for My model:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	5008
1	0.98	0.99	0.98	5001
2	0.98	0.98	0.98	5031
3	0.98	0.99	0.98	5010
4	0.98	0.98	0.98	5082
5	1.00	0.95	0.97	5138
6	0.97	0.99	0.98	5142
7	0.99	0.95	0.97	5069
8	0.97	0.98	0.97	5073
9	0.96	0.97	0.96	5022
accuracy			0.98	50576
macro avg	0.98	0.98	0.98	50576
weighted avg	0.98	0.98	0.98	50576

figure 17: Classification report for My Model

```
[[4966,    3,    3,    4,    2,    2,   16,    0,    4,    8],
 [  10, 4935,    5,    0,   25,    0,   10,   10,    1,    5],
 [   3,    8, 4948,   15,    8,    0,    7,    4,   19,   19],
 [   2,    2,    8, 4961,    5,    5,    2,    3,   14,    8],
 [   2,   28,    7,    8, 4989,    3,    9,    4,   19,   13],
 [  16,    5,    7,   38,    9, 4880,  105,    5,   55,   18],
 [  13,    6,    0,    1,    1,    6, 5095,    1,   19,    0],
 [   3,   43,    3,    8,   33,    2,    7, 4812,   21,  137],
 [  15,    2,   15,   20,    4,    5,   27,    1, 4974,   10],
 [   3,   20,   40,   17,   11,    0,    0,   24,   23, 4884]]
```

figure 18: Confusion Matrix for My Model

Discussion:

The VGG-19 had the best performance result compared to the others models,The customized model and DenseNet had similar performances, possibly due to the similarities in their architectures. ResNet might suffer from poor parameters like epoch and batch size, leading to underperformance compared to models like VGG and the custom model. ResNet may require more epochs to perform optimally.