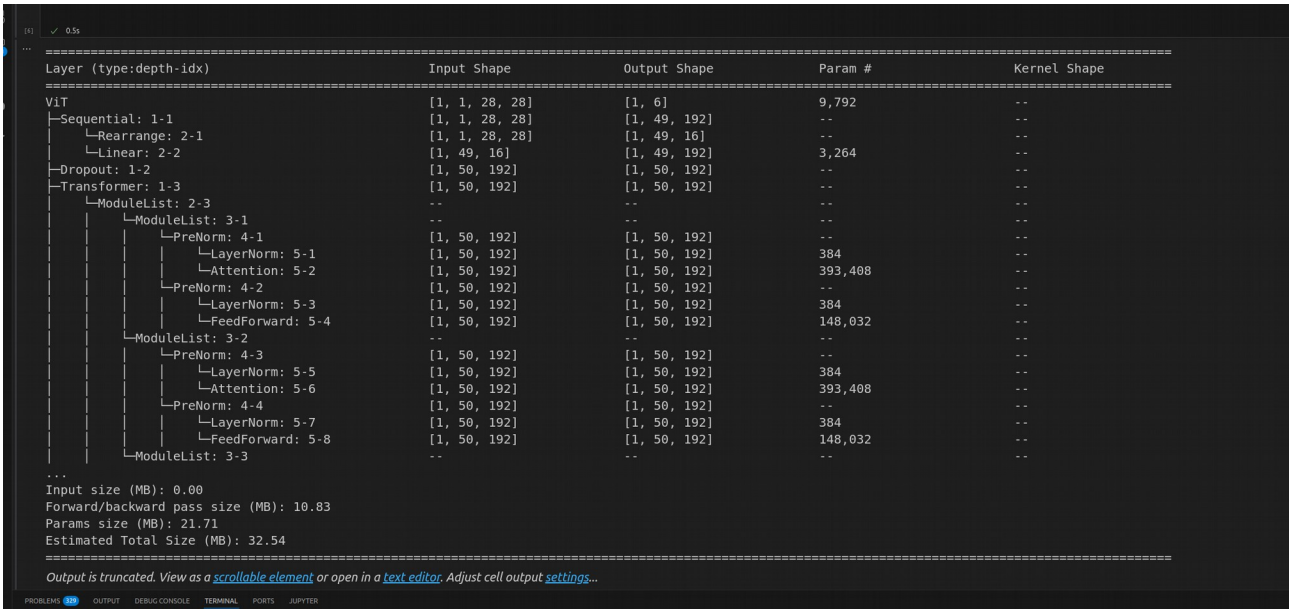
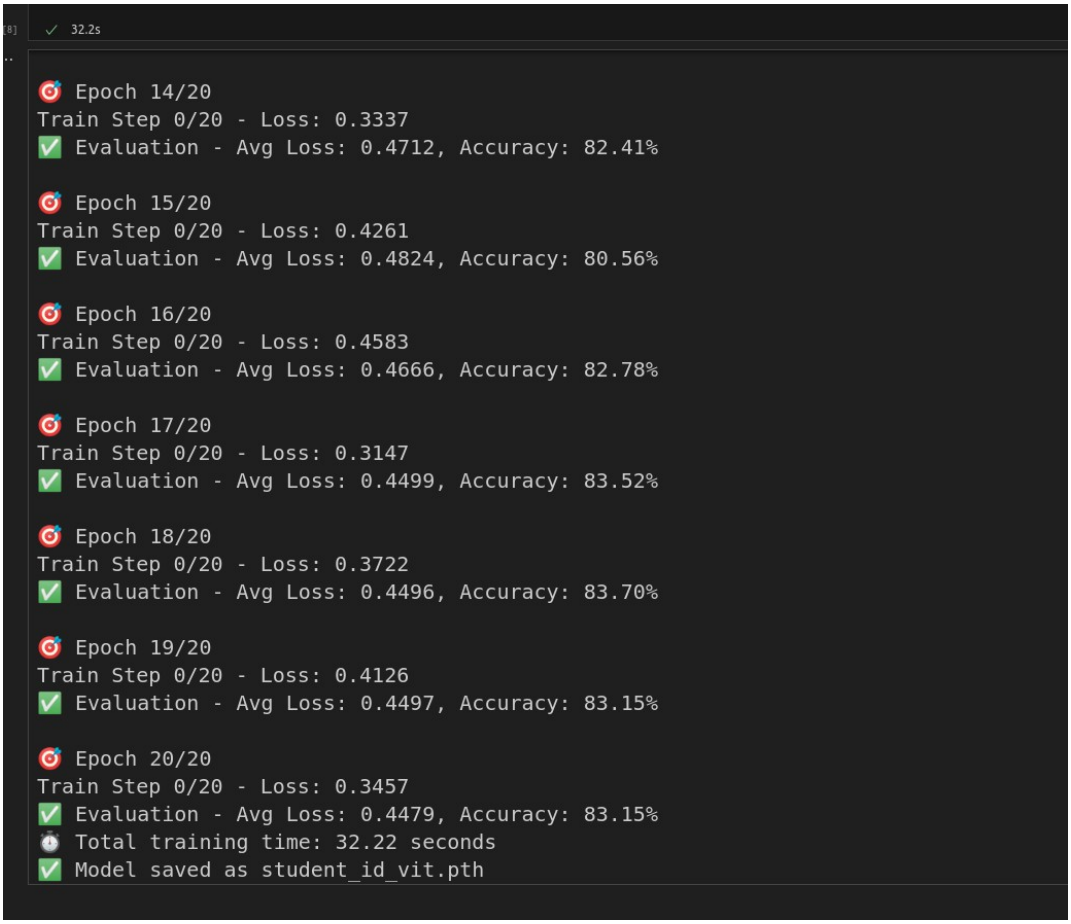
Lab8

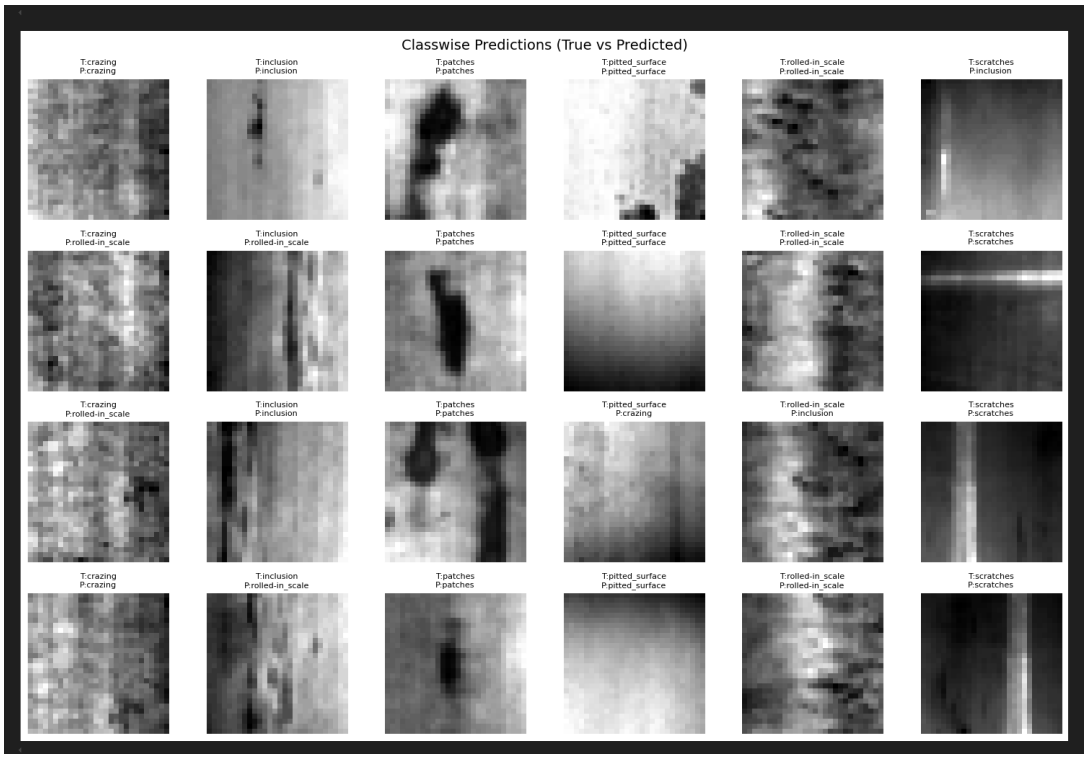
Results:

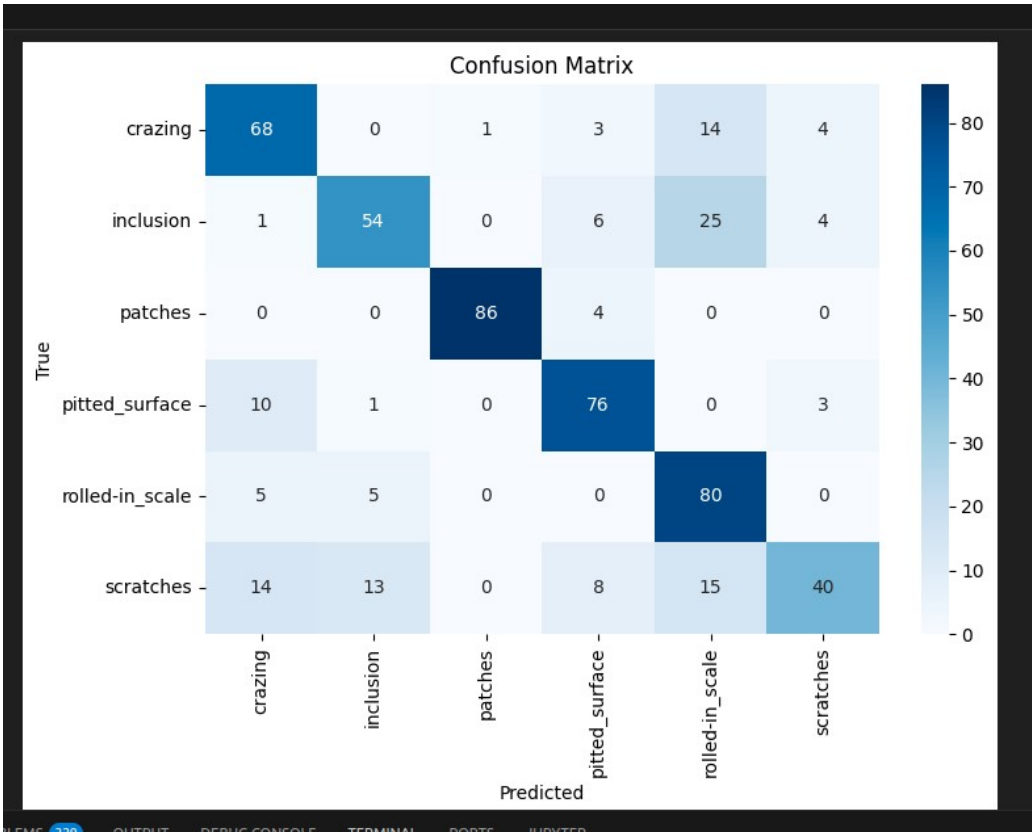


```
[6]  ✓ 0.5s
...
==========================================================================================================================================
Layer (type:depth-idx)                                  Input Shape               Output Shape              Param #                   Kernel Shape
==========================================================================================================================================
ViT                                                     [1, 1, 28, 28]            [1, 6]                    9,792                     --
├─Sequential: 1-1                                       [1, 1, 28, 28]            [1, 49, 192]              --                        --
│    └─Rearrange: 2-1                                   [1, 1, 28, 28]            [1, 49, 16]               --                        --
│    └─Linear: 2-2                                      [1, 49, 16]               [1, 49, 192]              3,264                     --
├─Dropout: 1-2                                          [1, 50, 192]              [1, 50, 192]              --                        --
├─Transformer: 1-3                                      [1, 50, 192]              [1, 50, 192]              --                        --
│    └─ModuleList: 2-3                                  --                        --                        --                        --
│    │    └─ModuleList: 3-1                             --                        --                        --                        --
│    │    │    └─PreNorm: 4-1                           [1, 50, 192]              [1, 50, 192]              --                        --
│    │    │    │    └─LayerNorm: 5-1                    [1, 50, 192]              [1, 50, 192]              384                       --
│    │    │    │    └─Attention: 5-2                    [1, 50, 192]              [1, 50, 192]              393,408                   --
│    │    │    └─PreNorm: 4-2                           [1, 50, 192]              [1, 50, 192]              --                        --
│    │    │    │    └─LayerNorm: 5-3                    [1, 50, 192]              [1, 50, 192]              384                       --
│    │    │    │    └─FeedForward: 5-4                  [1, 50, 192]              [1, 50, 192]              148,032                   --
│    │    └─ModuleList: 3-2                             --                        --                        --                        --
│    │    │    └─PreNorm: 4-3                           [1, 50, 192]              [1, 50, 192]              --                        --
│    │    │    │    └─LayerNorm: 5-5                    [1, 50, 192]              [1, 50, 192]              384                       --
│    │    │    │    └─Attention: 5-6                    [1, 50, 192]              [1, 50, 192]              393,408                   --
│    │    │    └─PreNorm: 4-4                           [1, 50, 192]              [1, 50, 192]              --                        --
│    │    │    │    └─LayerNorm: 5-7                    [1, 50, 192]              [1, 50, 192]              384                       --
│    │    │    │    └─FeedForward: 5-8                  [1, 50, 192]              [1, 50, 192]              148,032                   --
│    │    └─ModuleList: 3-3                             --                        --                        --                        --
...
Input size (MB): 0.00
Forward/backward pass size (MB): 10.83
Params size (MB): 21.71
Estimated Total Size (MB): 32.54
==========================================================================================================================================
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
PROBLEMS 329  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  JUPYTER
```



```
[8]  ✓ 32.2s
...
🎯 Epoch 14/20
Train Step 0/20 - Loss: 0.3337
✅ Evaluation - Avg Loss: 0.4712, Accuracy: 82.41%

🎯 Epoch 15/20
Train Step 0/20 - Loss: 0.4261
✅ Evaluation - Avg Loss: 0.4824, Accuracy: 80.56%

🎯 Epoch 16/20
Train Step 0/20 - Loss: 0.4583
✅ Evaluation - Avg Loss: 0.4666, Accuracy: 82.78%

🎯 Epoch 17/20
Train Step 0/20 - Loss: 0.3147
✅ Evaluation - Avg Loss: 0.4499, Accuracy: 83.52%

🎯 Epoch 18/20
Train Step 0/20 - Loss: 0.3722
✅ Evaluation - Avg Loss: 0.4496, Accuracy: 83.70%

🎯 Epoch 19/20
Train Step 0/20 - Loss: 0.4126
✅ Evaluation - Avg Loss: 0.4497, Accuracy: 83.15%

🎯 Epoch 20/20
Train Step 0/20 - Loss: 0.3457
✅ Evaluation - Avg Loss: 0.4479, Accuracy: 83.15%
⏱️ Total training time: 32.22 seconds
✅ Model saved as student_id_vit.pth
```


Classwise Predictions (True vs Predicted)
T:crazing P:crazing
T:inclusion P:inclusion
T:patches P:patches
T:pitted_surface P:pitted_surface
T:rolled-in_scale P:rolled-in_scale
T:scratches P:inclusion
T:crazing P:rolled-in_scale
T:inclusion P:rolled-in_scale
T:patches P:patches
T:pitted_surface P:pitted_surface
T:rolled-in_scale P:rolled-in_scale
T:scratches P:scratches
T:crazing P:rolled-in_scale
T:inclusion P:inclusion
T:patches P:patches
T:pitted_surface P:crazing
T:rolled-in_scale P:inclusion
T:scratches P:scratches
T:crazing P:crazing
T:inclusion P:rolled-in_scale
T:patches P:patches
T:pitted_surface P:pitted_surface
T:rolled-in_scale P:rolled-in_scale
T:scratches P:scratches


Confusion Matrix
crazing
inclusion
patches
pitted_surface
rolled-in_scale
scratches
True
Predicted
68 0 1 3 14 4
1 54 0 6 25 4
0 0 86 4 0 0
10 1 0 76 0 3
5 5 0 0 80 0
14 13 0 8 15 40
80
70
60
50
40
30
20
10
0

QA:

6. Briefly explain the role of patch embedding and positional encoding in ViT. You may include

In Vision Transformer, Patch Embedding cuts the image into small blocks (patches) and converts each patch into a vector through linear projection (Linear layer). This step is like the convolution kernel feature extraction of CNN.

Positional Encoding complements the Transformer's sensitivity to "sequential information" because it does not have a fixed spatial structure like CNN and the position information of each patch must be manually added.

7.Describe which hyperparameters you tuned, why you chose them, and how they affected your final accuracy.

dim, mlp_dim: Increasing the embedding dimension and number of layers can enhance the model's learning ability, but it is also necessary to balance memory and overfitting

Increasing depth :it has benefits about

1.Deeper feature learning capabilities

2.Stronger modeling capabilities

3.Enhanced model capacity

Increasing heads (Attention heads):

1.Capturing diverse relational features: Each head learns different types of dependencies.

2.Improve model expressiveness: Helps understand the interaction between different regions in the image.

3.Improving generalization capabilities: Learning multi-scale and multi-view attention.

8.Compare ViT and CNN for image classification: what are the main differences, and when might one be preferred over the other using the provided dataset?

	CNN	**ViT**
Feature extraction method	Convolution (translation invariant)	After patch segmentation, use linear layers and attention to learn features

	CNN	ViT
Spatial structure	Built-in receptive field to enhance local features	No built-in spatial structure, need to rely on Positional Encoding
Training material requirements	A small amount of information can also be used for stable learning	Usually requires a lot of data to learn stable

This project may be better using CNN because of mini dataset

9.Report the final achieved test accuracy; explain whether you reached the >60% baseline — if not, describe what you tried and why it might have failed; if you did, explain how you achieved it.

My final test accuracy was about 80%, which was well above the baseline of 60%.

training 20 epochs and increasing larger dim, mlp_dim has effect

10. Based on the paper you referred to (Dosovitskiy et al., An Image is Worth 16x16 Words), please brief explain: You may include parts of your Step 3 model code to support your explanation.

a.How does the Vision Transformer process input images from start to finish?

1.Divide the image into patches (e.g. 28x28 into 14x14 2x2 patches)

2.Each patch is linearly projected into a vector

3.Adding learnable positional encoding

4.Feed into the multi-layer Transformer Encoder

5.Take out the [CLS] token or average vector for classification

b.How are the image patches divided and transformed into input sequences?

ViT uses patches of fixed size (e.g. 2x2), each patch is flattened and then linearly projected and treated as an element of the sequence.

c.How does the multi-head self-attention mechanism operate within the Transformer encoder?

After each token generates query, key, and value:

$\text{Attention} = \text{softmax}(QK^T / \sqrt{d}) \cdot V$

Each head can learn different attention aspects (color, edge, shape, etc.)

Multiple heads are connected in series and then linearly transformed before being sent to the MLP module.

d.How does the model use the [CLS] token (or final output) to produce the final image classification?

[CLS] is a trainable token that is initially placed first in the sequence. After passing through the Transformer Encoder layer, it summarizes the information of the entire image and finally serves as the basis for classification output