HW4 Intelligent Systems

Fardin Abbasi 810199456

School of Electrical and Computer Engineering

College of Engineering

University of Tehran

Fall 2023

Questions

Q1: Neural Networks

A: Linear Regression

$$a^{(1)} = w^{(1)} x = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_1 x_1 + w_3 x_2 \\ w_2 x_1 + w_4 x_2 \end{bmatrix} \qquad y^{(1)} = \begin{bmatrix} c_1^{(1)} (w_1 x_1 + w_3 x_2) \\ c_2^{(1)} (w_2 x_1 + w_4 x_2) \end{bmatrix}$$

$$a^{(2)} = w^{(2)} y^{(1)} = [w_5 \; w_6] y^{(1)} \qquad y^{(2)} = c^{(2)} a^{(2)} = c^{(2)} \left[w_5 c_1^{(1)} (w_1 x_1 + w_3 x_2) + w_6 c_2^{(1)} (w_2 x_1 + w_4 x_2) \right]$$

$$= \delta_1 x_1 + \delta_2 x_2$$

where $\delta_1 = c^{(2)} (w_5 c_1^{(1)} w_1 + w_6 c_2^{(1)} w_2) \cancel{x_1}$ & $\delta_2 = c^{(2)} (w_5 c_1^{(1)} w_3 + w_6 c_2^{(1)} w_4)$

B: Binary classification

$$y^{(1)} = \begin{bmatrix} \text{sign}(\delta(w_1 + w_3 x_1 + w_5 x_2) - 0.5) \\ \text{sign}(\delta(w_2 + w_4 x_1 + w_6 x_2) - 0.5) \end{bmatrix}$$

$$y^{(2)} = c w_7 + c w_8 \, \text{sign}(\delta(w_1 + w_3 x_1 + w_5 x_2) - 0.5) + c w_9 \, \text{sign}(\delta(w_2 + w_4 x_1 + w_6 x_2) - 0.5)$$

$Decision\ boundry: y^{(2)} = 0$

There is no alternative neural network without hidden layers!

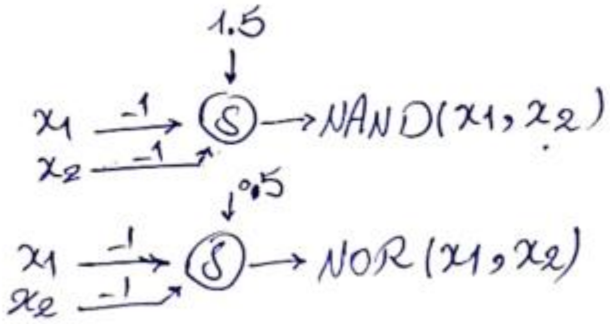
Q2: Neural networks

A: Multi-layer networks

1.

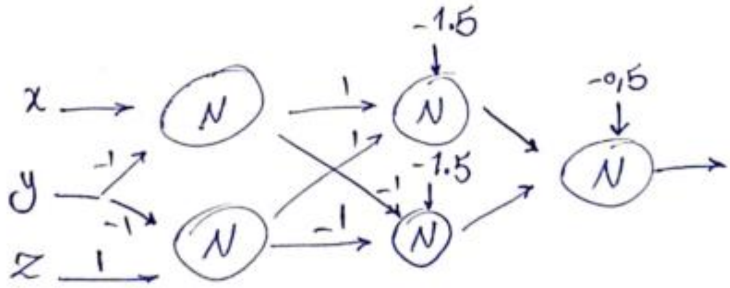
NAND: $\delta(1.5 - x_1 - x_2)$

NOR: $\delta(0.5 - x_1 - x_2)$



2.

$N := 2\,sign(\Sigma) - 1$



B: Regularization

1.

$$\frac{\partial E}{\partial w} = 2\left(y - \sum_i w_i x_i\right)\left(-\sum_i x_i\right) + 2\lambda\left(\sum_i w_i\right) \qquad w_{t+1} = w_t - \eta \frac{\partial E}{\partial w}$$

Since in each iteration new w is decayed with $2\lambda w$, L2 regularization is also called weight decay.

2.

$$\frac{\partial E}{\partial w} = -2\left(y - \sum_i w_i x_i\right)\left(\sum_i x_i\right) + \lambda \sum_i sign(w_i)$$

This regularization is called L1 regularization or lasso regression.

In this method, since the decay size is regardless of w, so it shrinks smaller wights to zero.

So, this works well for **feature selection** in case we have a huge number of features.

This regularization is more robust to overfitting and makes model more interpretable.

3.

$\boldsymbol{\|\|X_1\|\|_1 = 7}$	$\boldsymbol{\|\|X_2\|\|_1 = 7}$	$\boldsymbol{\|\|X_3\|\|_1 = 5}$
$\boldsymbol{\|\|X_1\|\|_2 = \sqrt{17}}$	$\|\|X_2\|\|_2 = 5$	$\|\|X_3\|\|_2 = 5$

$||X_3||_1$ has the smallest norm-1 in comparison to others which confirms that L1 regularization makes weights sparse.

C: Analytical

1.

Since the objective function is highly complicated and has low stochastic noise, if we have adequate dataset its better to estimate it with a complicated model, else we have to use a more simple model.

2.

Since logical functions can be simulated with one layer of “AND” gates and one layer of “OR” gates, so it can be simulated with maximum of 2 layers.

3.

Validation set is used for tuning hyperparameters while test set is used for evaluating model performance. Since the validation set is seen while training, there should be a separated dataset for testing the model.

Q3: Convolutional Neural Networks

Part 1: Classification with CNNs

A: Preprocessing

Training

Testing

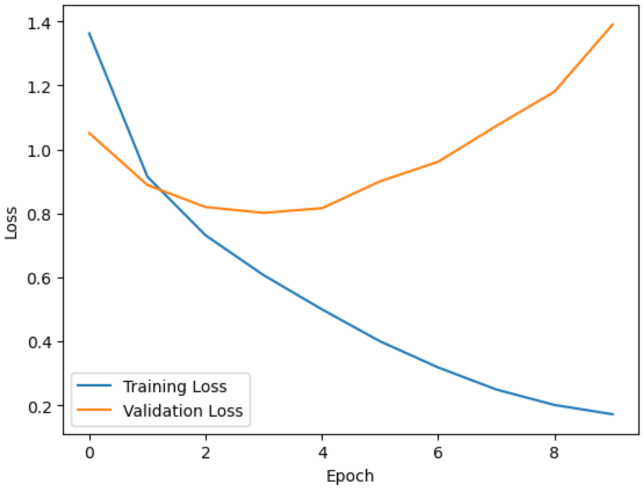
Validation

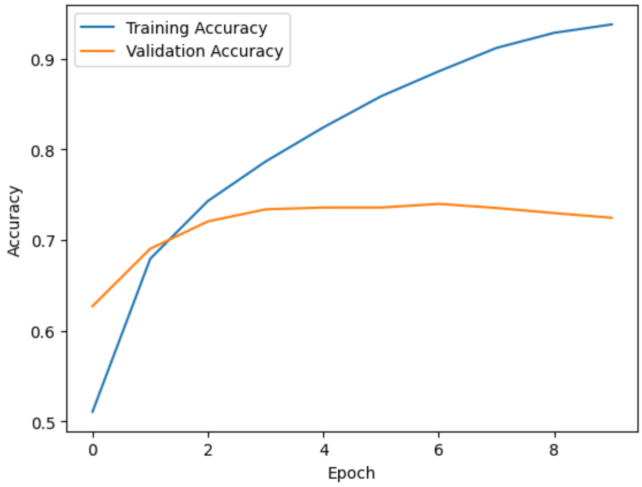
B: Implementation & setting hyperparameters

In last layer the chosen activation function has to be appropriate for classification tasks. Softmax activation function would map outputs to the probability of belonging to each class. Knowing, the labels are one-hot encoded and the task is classification we use cross entropy loss.

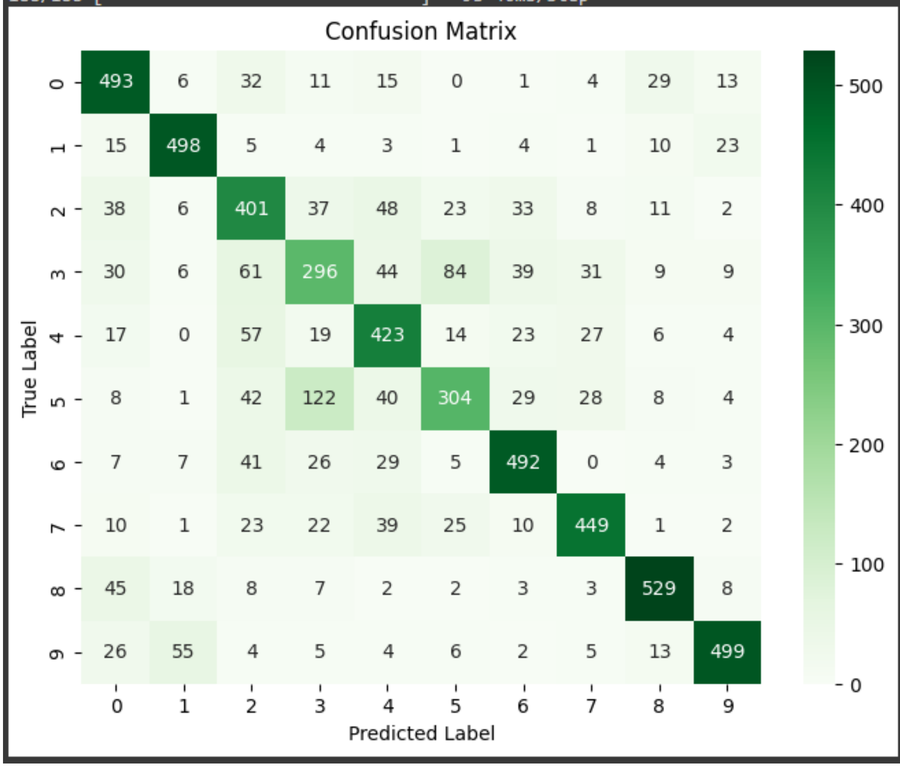
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0
 D)

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_1 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 128)               524416

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 591274 (2.26 MB)
Trainable params: 591274 (2.26 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

C: Training & evaluation



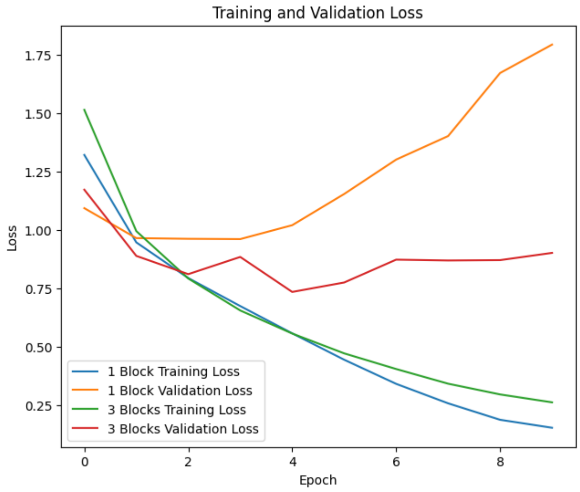


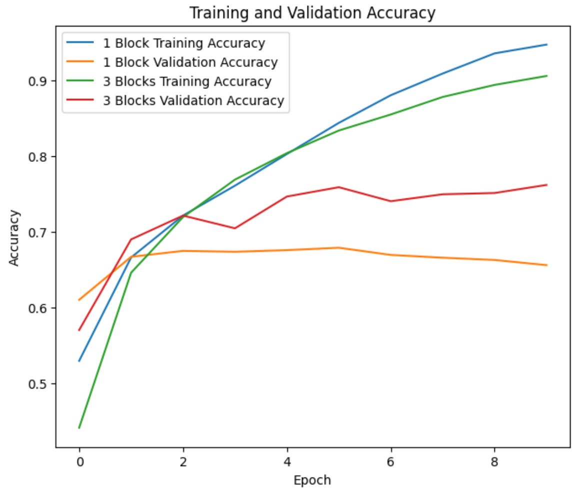

```
188/188 [==============================] - 8s 42ms/step - loss: 1.2776 - accuracy: 0.7307
Test Loss: 1.2776
Test Accuracy: 73.07%
188/188 [==============================] - 9s 46ms/step
```



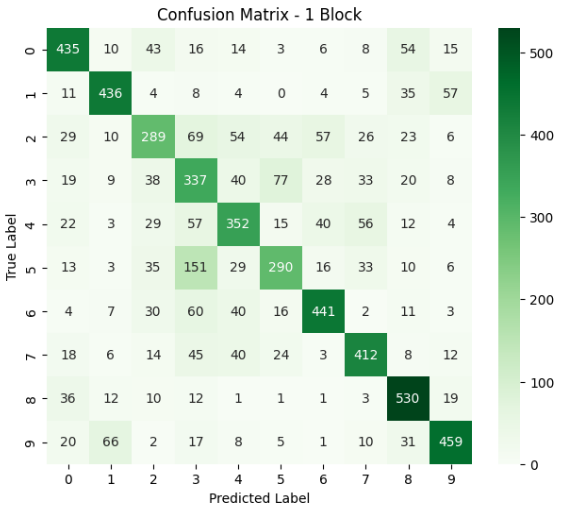
D: Improving the model

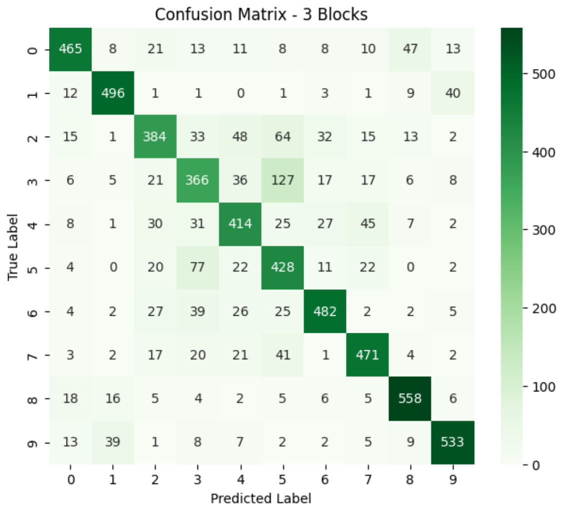
Testing 1 and 3 blocks






```
188/188 [==============================] - 4s 22ms/step - loss: 1.7773 - accuracy: 0.6635
188/188 [==============================] - 14s 76ms/step - loss: 0.8868 - accuracy: 0.7662

Model with 1 Block:
Test Loss: 1.7773
Test Accuracy: 66.35%

Model with 3 Blocks:
Test Loss: 0.8868
Test Accuracy: 76.62%
```

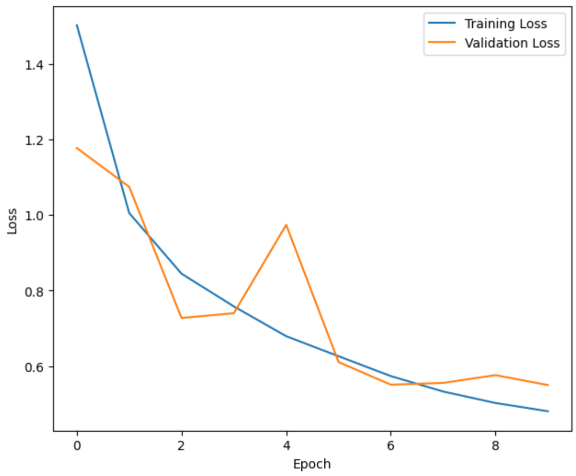


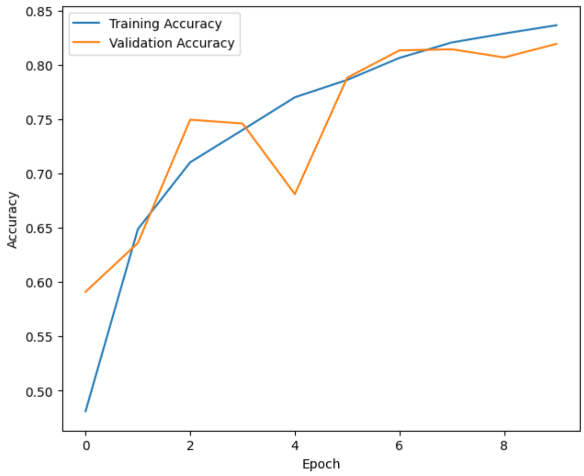


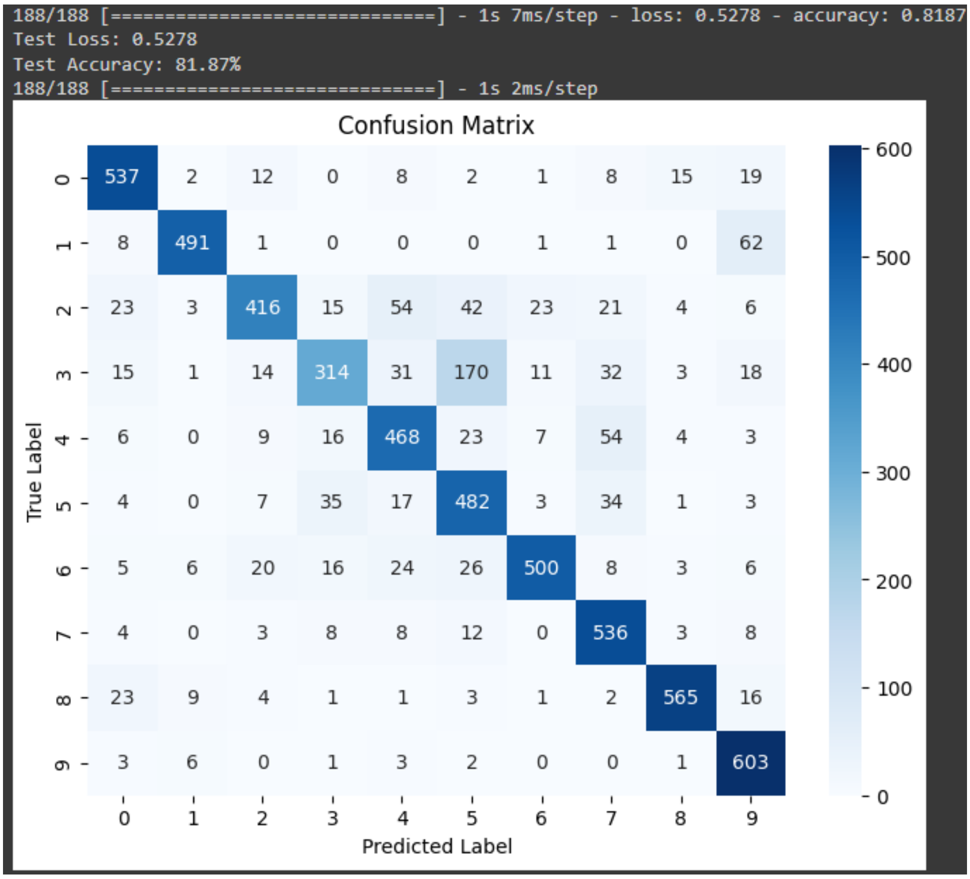
It is shown, the model with 3 blocks performs better.

Adding BatchNormalization and Dropout

```
Layer (type)                Output Shape              Param #
=================================================================
conv2d (Conv2D)             (None, 32, 32, 32)        896

batch_normalization (Batch  (None, 32, 32, 32)        128
Normalization)

conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

batch_normalization_1 (Bat  (None, 32, 32, 32)        128
chNormalization)

max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0
D)

dropout (Dropout)           (None, 16, 16, 32)        0

conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496

batch_normalization_2 (Bat  (None, 16, 16, 64)        256
chNormalization)

conv2d_3 (Conv2D)           (None, 16, 16, 64)        36928

batch_normalization_3 (Bat  (None, 16, 16, 64)        256
chNormalization)

max_pooling2d_1 (MaxPoolin  (None, 8, 8, 64)          0
g2D)

dropout_1 (Dropout)         (None, 8, 8, 64)          0

conv2d_4 (Conv2D)           (None, 8, 8, 128)         73856

batch_normalization_4 (Bat  (None, 8, 8, 128)         512
chNormalization)

conv2d_5 (Conv2D)           (None, 8, 8, 128)         147584

batch_normalization_5 (Bat  (None, 8, 8, 128)         512
chNormalization)

max_pooling2d_2 (MaxPoolin  (None, 4, 4, 128)         0
g2D)

dropout_2 (Dropout)         (None, 4, 4, 128)         0

flatten (Flatten)           (None, 2048)              0

dense (Dense)               (None, 128)               262272

batch_normalization_6 (Bat  (None, 128)               512
chNormalization)

dropout_3 (Dropout)         (None, 128)               0

dense_1 (Dense)             (None, 10)                1290
```



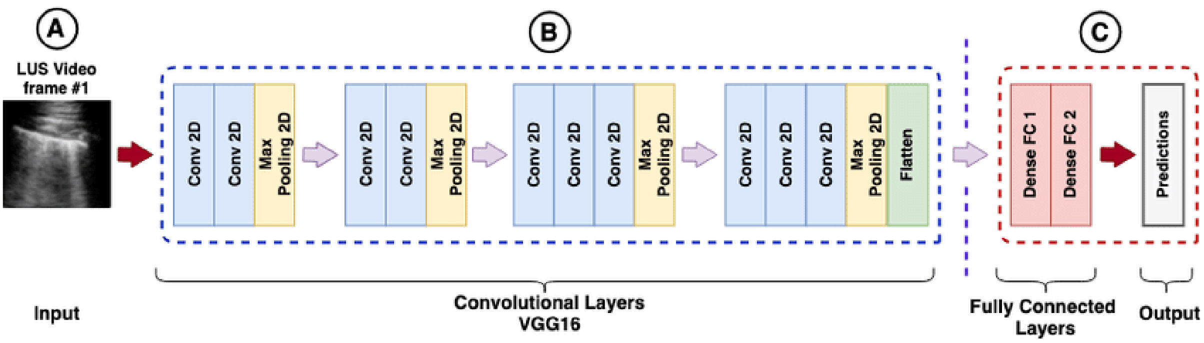




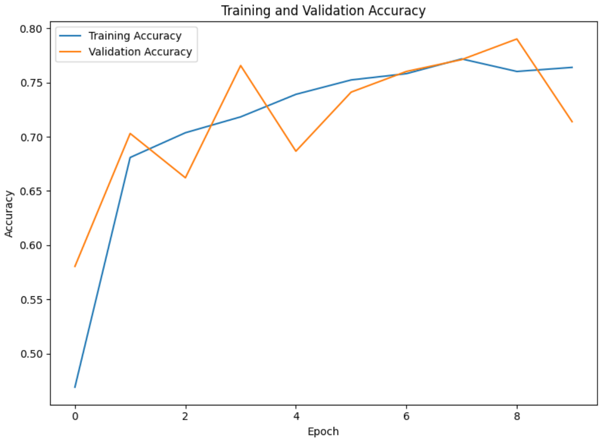
With this new architecture, the loss function is reduced on test data and performance is improved on test data.

Part 2: Transform learning

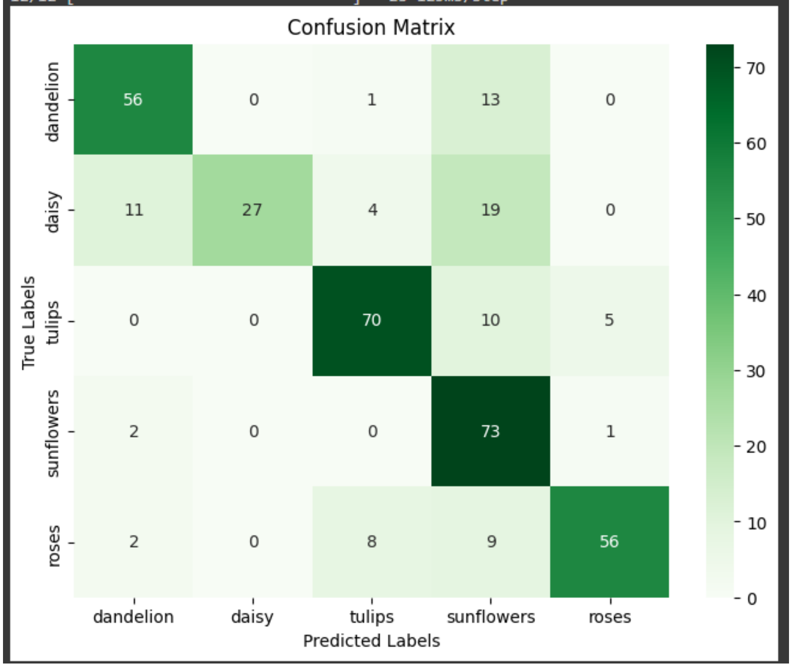
```
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 global_average_pooling2d (  (None, 512)               0
 GlobalAveragePooling2D)

 dense_2 (Dense)             (None, 1024)              525312

 dense_3 (Dense)             (None, 5)                 5125

=================================================================
```



In VGG16, part B is for feature extraction and part c is for classification.




```
12/12 [==============================] - 2s 119ms/step - loss: 0.6532 - accuracy: 0.
Test Loss: 0.6532
Test Accuracy: 0.7684
12/12 [==============================] - 2s 123ms/step
```



	precision	recall	f1-score	support
dandelion	0.79	0.80	0.79	70
daisy	1.00	0.44	0.61	61
tulips	0.84	0.82	0.83	85
sunflowers	0.59	0.96	0.73	76
roses	0.90	0.75	0.82	75
accuracy			0.77	367
macro avg	0.82	0.75	0.76	367
weighted avg	0.82	0.77	0.76	367