Classify the Flower dataset

I. Network structure

The network structure is as follows:

Output layer: **Fully connected layer 2** with 5 output units.
Fully connected layer 1 with 256 output units and ReLU activation followed by BatchNorm1d and Dropout.
Convolutional layer 3 with kernel size (3,3) and 128 filters and ReLU activation followed by BatchNorm2d, MaxPool(kernel_size=2, stride=2) and Dropout.
Convolutional layer 2 with kernel size (3,3) and 64 filters and ReLU activation followed by BatchNorm2d, MaxPool(kernel_size=2, stride=2) and Dropout.
Convolutional layer 1 with kernel size (3,3) and 32 filters and ReLU activation followed by BatchNorm2d, MaxPool(kernel_size=2, stride=2) and Dropout.
Input layer

The loss function is crossentropy because this network is doing classification tasks. For the convolutional layers, they are applied because the inputs are images, and CNN is appropriate for extracting features in images. The max pooling layer down samples the outputs of the convolutional layer and extracts the representative features in a local area. The dropout layer is there to prevent the model from overfitting. Only three convolutional layers are used, which is much less than the VGG or ResNet models. Then, there is a fc layer with 256 units with ReLU activation. Another dropout layer after the first fc layer is added to enhance the performance on test data. Finally, a fc layer with 5 units is the output layer. Given the loss function, the softmax activation function is implicitly chosen because this represents the probability of belonging to each class well.

II. Representative weights

Conv1:

```
tensor([[-0.0988, -0.1107, -0.1187],
        [-0.1925, -0.1807,  0.2660],
        [-0.1711,  0.0439,  0.3947]],
       device='cuda:0', grad_fn=<SelectBackward>)
```

Conv2:

```
tensor([[-0.1021, -0.1621, -0.1276],
        [-0.1491, -0.0778, -0.0101],
        [-0.0599,  0.0673,  0.0987]],
       device='cuda:0', grad_fn=<SelectBackward>)
```

Conv3:

```
tensor([[ 0.0847, -0.0132, -0.0032],
        [ 0.0051, -0.0306, -0.0552],
        [ 0.0401, -0.0501, -0.0438]],
       device='cuda:0', grad_fn=<SelectBackward>)
```

FC1:

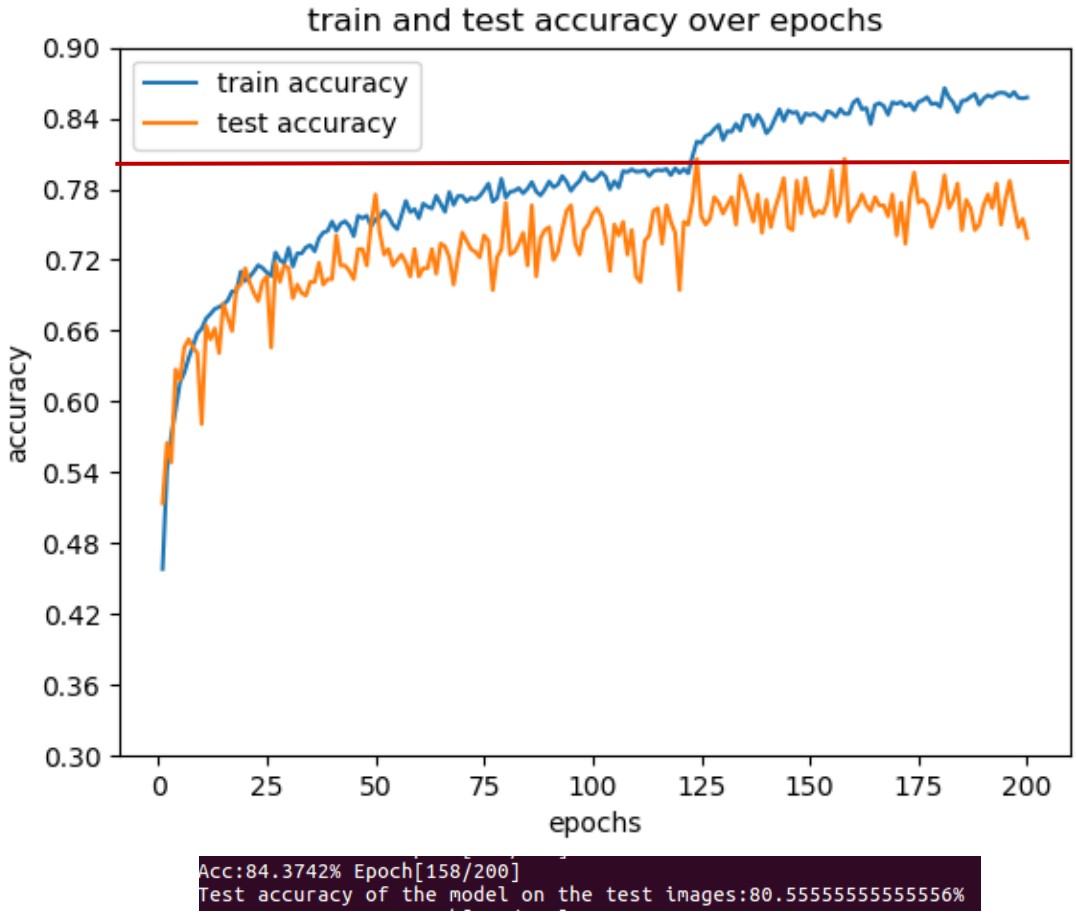
```
tensor([ 0.0044, -0.0177,  0.0088,  0.0151,  0.0067,  0.0337,  0.0069,  0.0189,
         0.0129,  0.0170,  0.0261,  0.0020, -0.0034,  0.0138, -0.0122,  0.0060,
         0.0124, -0.0073, -0.0025, -0.0123],
       device='cuda:0', grad_fn=<SliceBackward>)
```

FC2:

```
tensor([ 0.1487,  0.0437, -0.0115, -0.1392,  0.0908,  0.0975,  0.0445, -0.1449,
        -0.1437, -0.1621, -0.1156,  0.1272, -0.1528,  0.1249,  0.0015, -0.0868,
        -0.1743,  0.1571,  0.1169, -0.1927],
       device='cuda:0', grad_fn=<SliceBackward>)
```

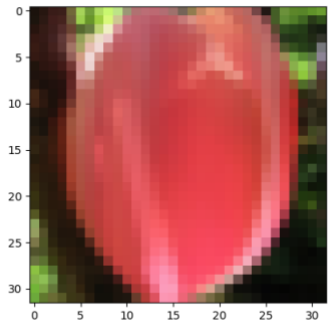
As we can see from the above prints, all of the weights are in the range [-1,1], which is a reasonable range. The weights are not tuned to be all zeros or identical, so they are likely to yield a good output.

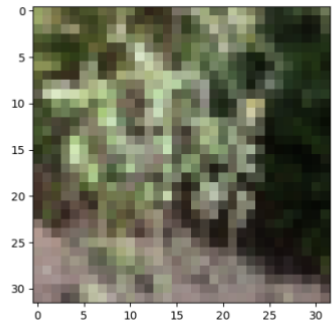
III. Training and testing accuracy vs epoch

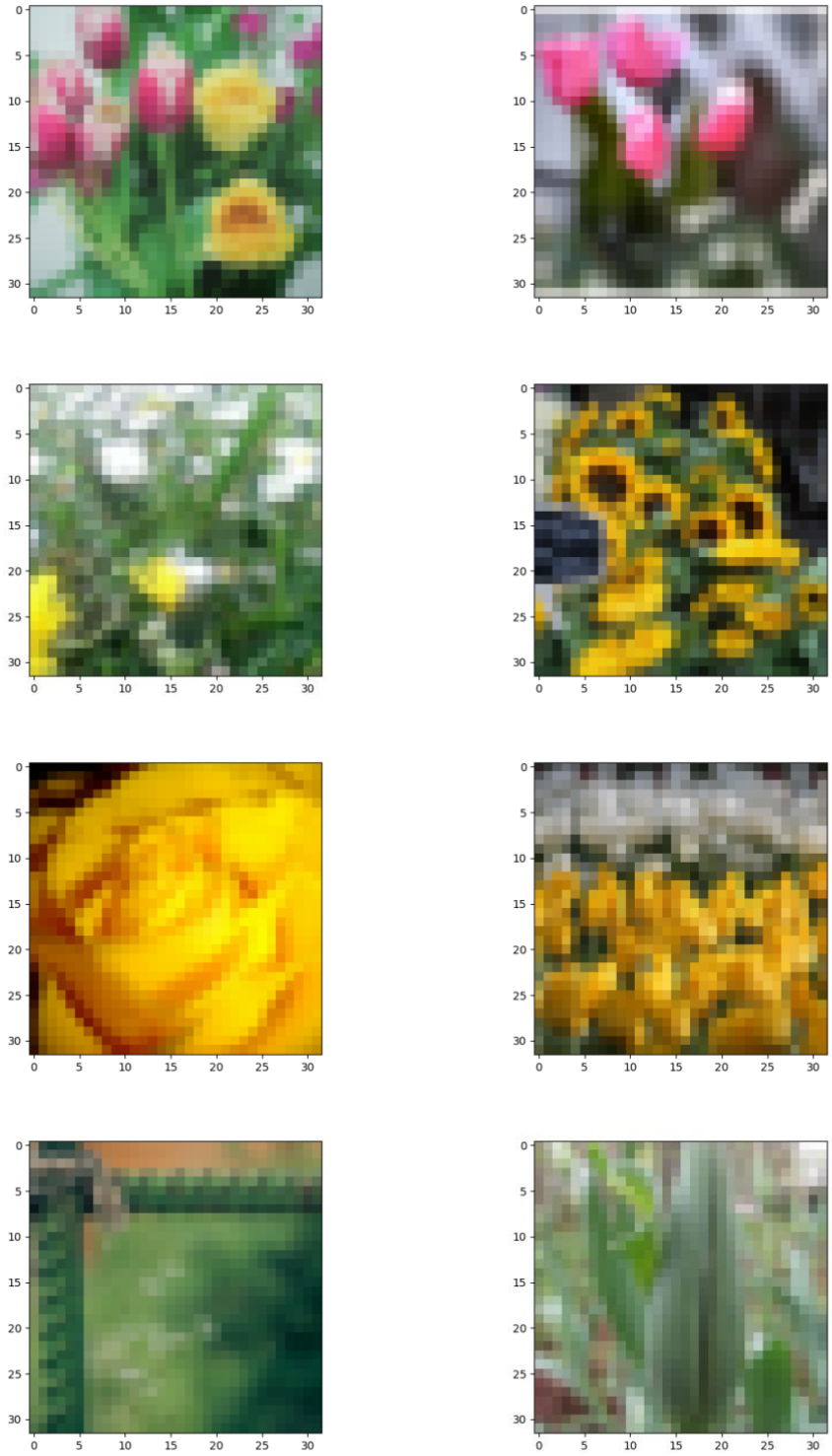


The training accuracy is increasing with the number of epochs, which is expected. Because of the dropouts, the model generalizes well so the test accuracy is also increasing at the first several epochs. However, the test accuracy cannot be improved in later epochs because of the overfitting. During the training process, there was a model that has an accuracy over 80% on test set. And in the later epochs, the test accuracies are higher than 75% in general.

IV. Data points that fail in the model







As we can see from the above misclassified flowers, some of them are very similar to both daisy and tulip, and some are too vague for us to distinguish. It is reasonable that the network finds it difficult to classify them.