Lab 6: Flower Classification using CNN

Spring 2021

BRAE 428

Dr. Bo Liu

Emma Gray, Chandler Jones, Gracely Speth

Abstract

Convolutional Neural Network (CNN) is a deep learning technique based on image classification and recognition. In this lab students use MATLAB to create a CNN to identify flowers. This contains similar methods to the California Polytechnic University's BRAE-428 Lab 4 regarding training a network. In this report, the CNN used was AlexNet in the Deep Network Designer of MATLAB's Deep Learning Toolbox. AlexNet is 8 layers deep and was pretrained on more than 1 million images from the ImageNet database. AlexNet has the capability to characterize information into 1000 categories. In this lab we limited our training to 10 types of flowers and were able to successfully categorize types of flowers from the internet. With the use of AlexNet Network in the Deep Learning Toolbox we were able to adjust this network and identify flowers through real time.

Introduction

In this lab students will create a CNN to classify flowers. CNN models take the input image and run it through a series of hidden layers and classify the images probability of being in a category. A computer classifies the image using convolutional layers, each adding a new filter to the image to better classify the image. Kernels are used to identify certain features of the input image, and the weights of the kernels are adjusted through the training process. Finally, the model is ready to test, and in this lab, we tested the model to see how well it can identify different flowers in real time. We believe that will a well, but not overtrained, CNN we will be able to successfully identify images of flowers in real time using a webcam. Overtraining is something to be cautious about. If a network is overtrained, it learns and associates the noise within the examples to its outputs and will perform poorly when tested in the real world.

Materials and Methods

Materials and Methods: This part should include experimental procedures, subjects, equipment, mathematical equations and statistical analysis in a sentence form. You should write this section so that other people with similar academic background can duplicate the experiment without any extra instructions. (FROM RUBERIC in syllabus)

Materials:

Computer

MathWork's MATLAB

Deep Learning Toolbox

AlexNet

Webcam

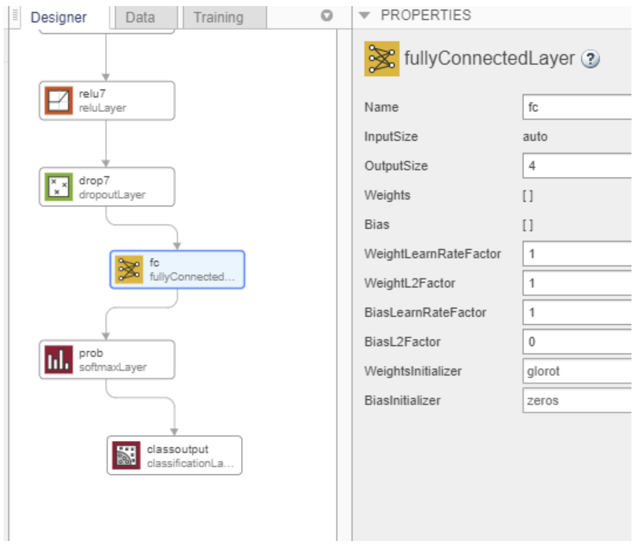
Flower Images

Methods:

1. Open MATLAB
2. Download the train folder and load it into your MATLAB.
3. Pick 5 flowers out of the folder and move them to your workspace.
4. Download some other images of those flower from the internet.
5. Open the image directory and upload your train photos into the program.
6. Import AlexNet (you may have to install the toolbox onto your computer)
7. Go to the APPS tab in MATLAB and select the Deep Network Designer APP, import the AlexNet CNN model from the workspace.
8. Make some modifications to the existing AlexNet processer by deleting the third to last layer and replacing it with a fully connected layer. Delete the last layer and replace it with a new classification Layer.
9. Export the modified Alex model back into the workspace in order to run it.
10. Use code to train the new CNN model you just created.
11. Test your trained system using the downloaded images.
12. When your system is fully trained and identifying your 5 flowers download the other 5 flower folders from the train folder.
13. Retrain your program using all 10 flower types.
14. Test your new CNN with other internet images of the new floweres.

Results:

Our results will in part be a discussion of how we followed the instructions of the lab to better understand the processes we followed. As seen in **Figure 1.** below, we correctly modified the two sections of AlexNet so that we can correctly classify our flower selections. For the first section of of the lab, it was necessary to change the classifiacation size to ‘4’.

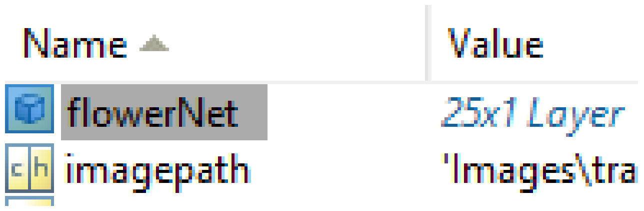


fullyConnectedLayer

Name	fc
InputSize	auto
OutputSize	4
Weights	[]
Bias	[]
WeightLearnRateFactor	1
WeightL2Factor	1
BiasLearnRateFactor	1
BiasL2Factor	0
WeightsInitializer	glorot
BiasInitializer	zeros

Figure 1. Team Modified AlexNet to have an output size of 4

In **Figure 2.** below, the exported model can be seen with the correct name. This will allow us to train on our image dataset.



Name	Value
flowerNet	25x1 Layer
imagepath	'Images\tra

Figure 2. Team exported New CNN as ‘flowerNet’

In order to train on our 4 classification model, we opted to temporarily delete the excess subfolders and limit our classification to the first four flower groups, Bluebell, Daffodil, Daisy, and Fritillary. This subfloder structure can be seen on the next page in **Figure 3.**

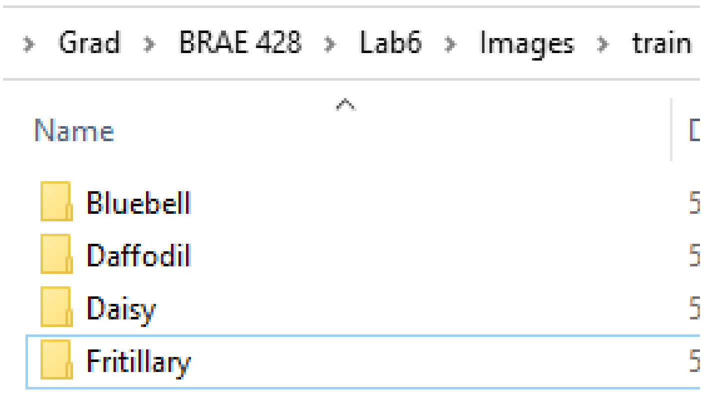


Figure 3. Team chose the top 4 subfolders to train on

After correctly organizing our model, we trained using our internal CPU. For this small network, it did not take much time at all and was suitable. We achieved supreme accuracy and were willing to test as is without modification. The results of our training can be seen below in **Figure 5.**

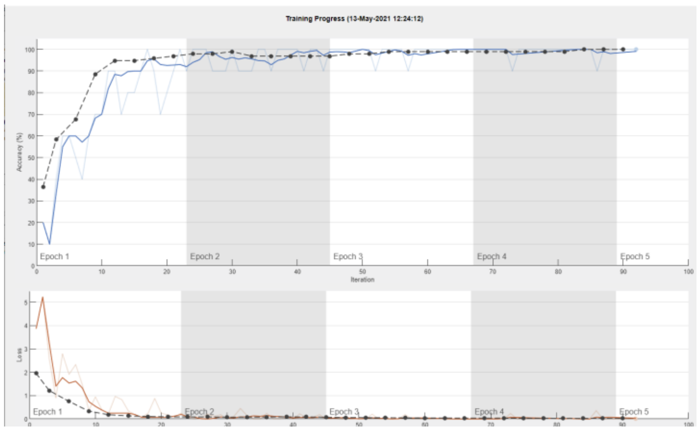


Figure 4. Team Trained ‘flowerNet’ on the 4 types of flowers

On the next page we will overview the test of our model.

In **Figure 6.** a test image is shown of a white daisy. This image was found online via duckduckgo. We saved this image into our matlab folder, and after importing, resized it to the proper 227x227 pixels required for our neural net.

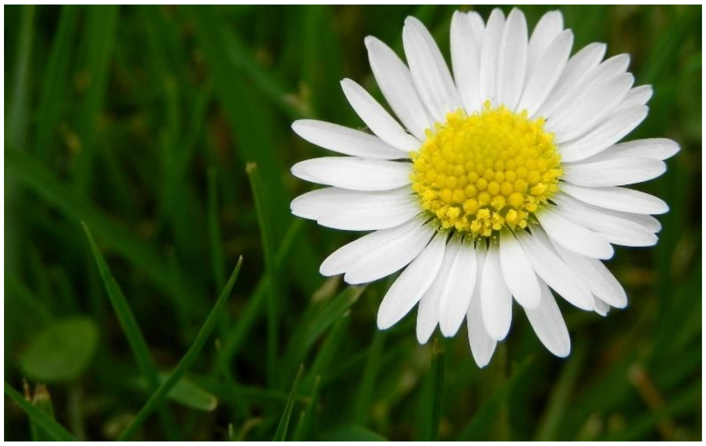
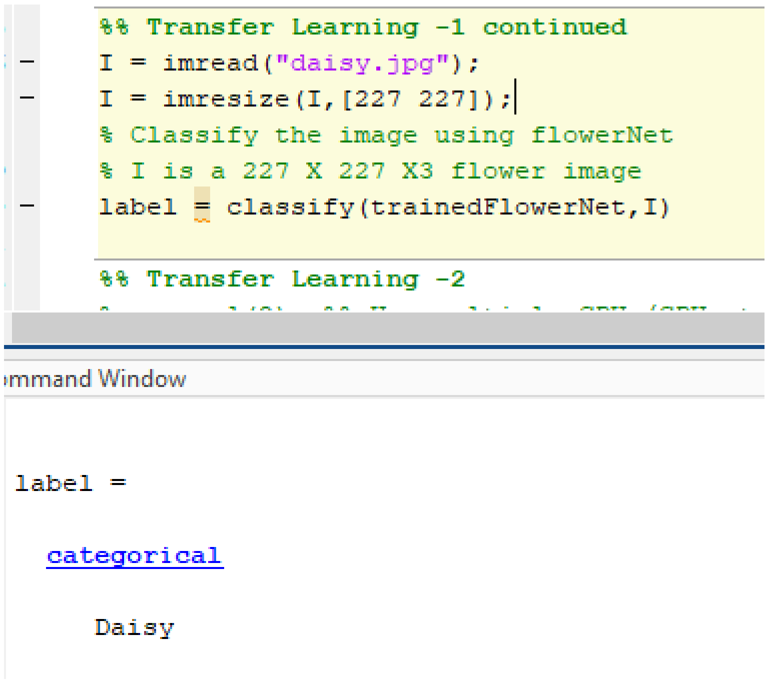


Figure 5. Imported image of Daisy from the Web

The newly found image, not part of our training set, was ran through the minimal CNN and the results found that it correctly classified the image as a Daisy! This is shown in **Figure 6.** below.



```
%% Transfer Learning -1 continued
I = imread("daisy.jpg");
I = imresize(I,[227 227]);
% Classify the image using flowerNet
% I is a 227 X 227 X3 flower image
label = classify(trainedFlowerNet,I)

%% Transfer Learning -2
```

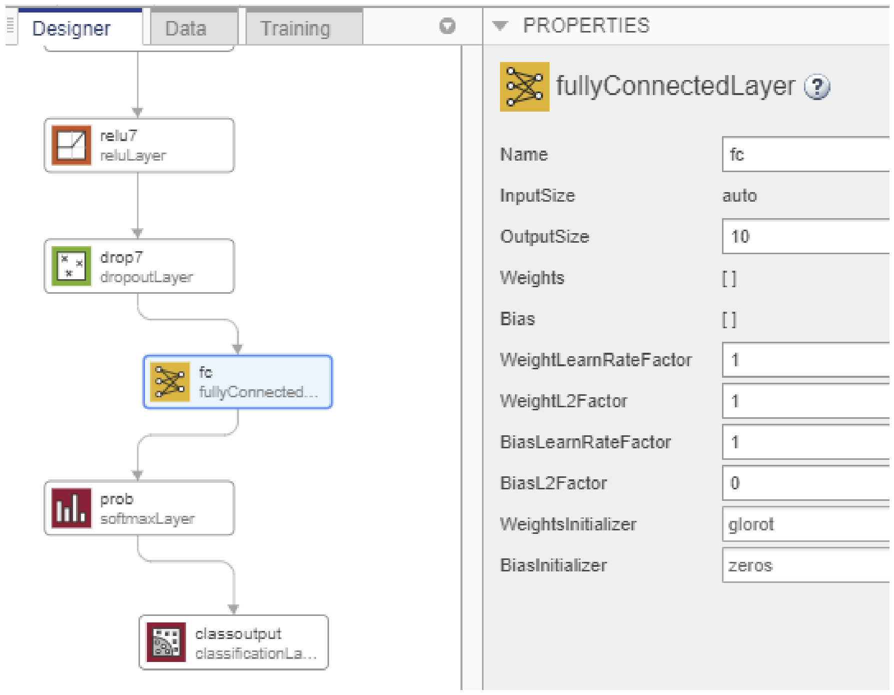
```
label =

  categorical

     Daisy
```

Figure 6. ‘flowerNet’ Correctly identifies image as a Daisy

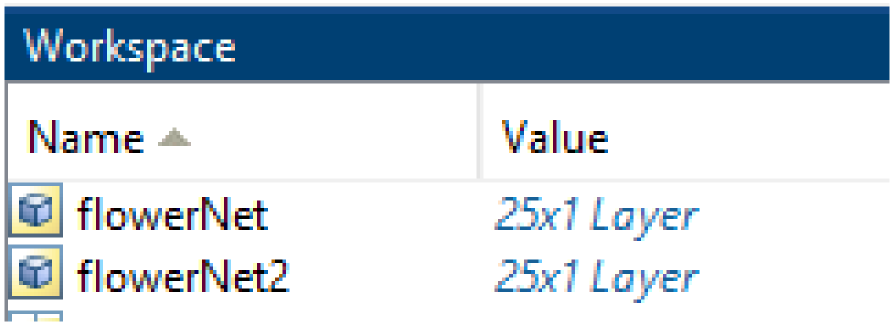
In the second step of the lab, the team reorganized and recreated our CNN to allow for all 10 classifications. This is shown below in **Figure 7.**



Name	fc
InputSize	auto
OutputSize	10
Weights	[ ]
Bias	[ ]
WeightLearnRateFactor	1
WeightL2Factor	1
BiasLearnRateFactor	1
BiasL2Factor	0
WeightsInitializer	glorot
BiasInitializer	zeros

Figure 7. Team modified CNN to output all 10 flower types

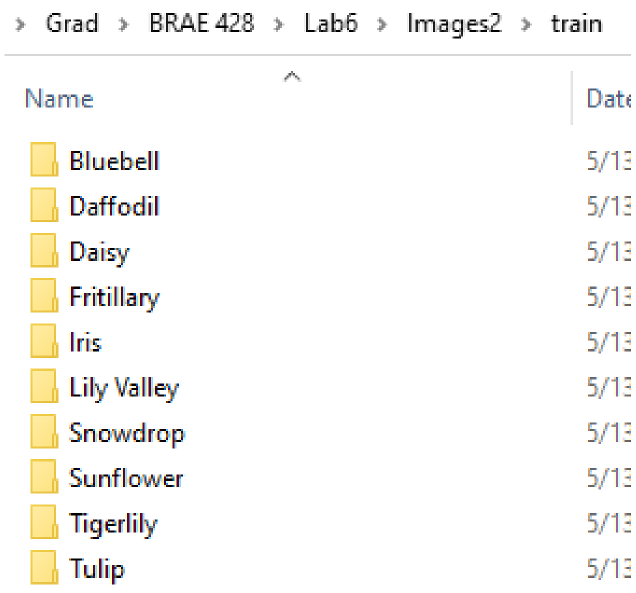
We retrained this model, and renamed it 'flowerNet2' to distinguish it from our first model. Our workspace is shown below in **Figure 8.**



Name	Value
flowerNet	25x1 Layer
flowerNet2	25x1 Layer

Figure 8. Team exported new NeuralNet as 'flowerNet2' with all 10 categories

Similarly, the team then recaptured the training images in a newsubfolder titled 'Image2' This distinguises our training images from the initial set, and is shown on the next page in **Figure 9.**

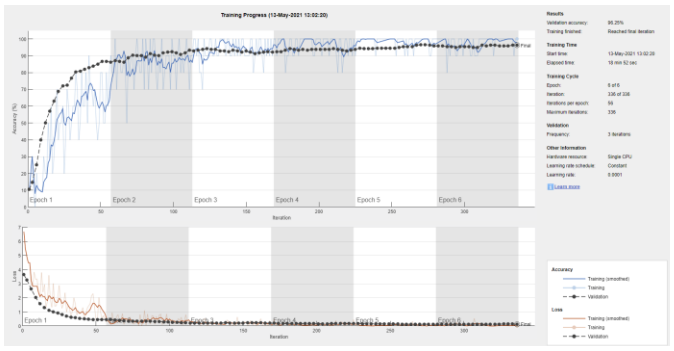

› Grad › BRAE 428 › Lab6 › Images2 › train

Name	Date
Bluebell	5/13
Daffodil	5/13
Daisy	5/13
Fritillary	5/13
Iris	5/13
Lily Valley	5/13
Snowdrop	5/13
Sunflower	5/13
Tigerlily	5/13
Tulip	5/13

Figure 9. Team created second folder with all 10 categories

At this point, the team attempted to train the second model. It is noted that this larger set takes significantly more computing resources. It was attempted to modify the code using the 'Parallel Computing Toolbox' to implement a GPU to train the system. Unfortunately, the team's desktop uses an RX 580 GPU which does not have the CUDA architecture required to train neural networks. We opted to take the time to train via onboard CPU instead, and will look into acquiring the GPU necessary to train CNNs in the interim.

Using a Ryzen 1950 Threadripper, the training time was 19 minutes. Since the loss trended to zero after 2 epochs, it is hypothesized that the model would not be harmed by reducing the training significantly. However, the accuracy reached >90% after the end of the 2nd epoch, and trended much more slowly upwards to 96%+ after 5 epochs. From our prior knowledge, we understand that 85-95% accuracy is the sweet spot for training CNN to avoid overfitting to the training set. Our training results are shown below in **Figure 10.**



Results

Validation accuracy:	96.25%
Training finished:	Reached final iteration

Training Time

Start time:	13-May-2021 13:02:20
Elapsed time:	18 min 52 sec

Training Cycle

Epoch:	6 of 6
Iteration:	336 of 336
Iterations per epoch:	56
Maximum iterations:	336

Validation

Frequency:	3 iterations

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Learn more

Figure 10. Training Results of 'flowerNet2'; all 10 classifications

With a tested model, the team tested multiple different types of flowers. At first we tested using images of tulips from the web. Unfortunately, both tulip pictures were chosen by the neural network to be Fritillary flowers.

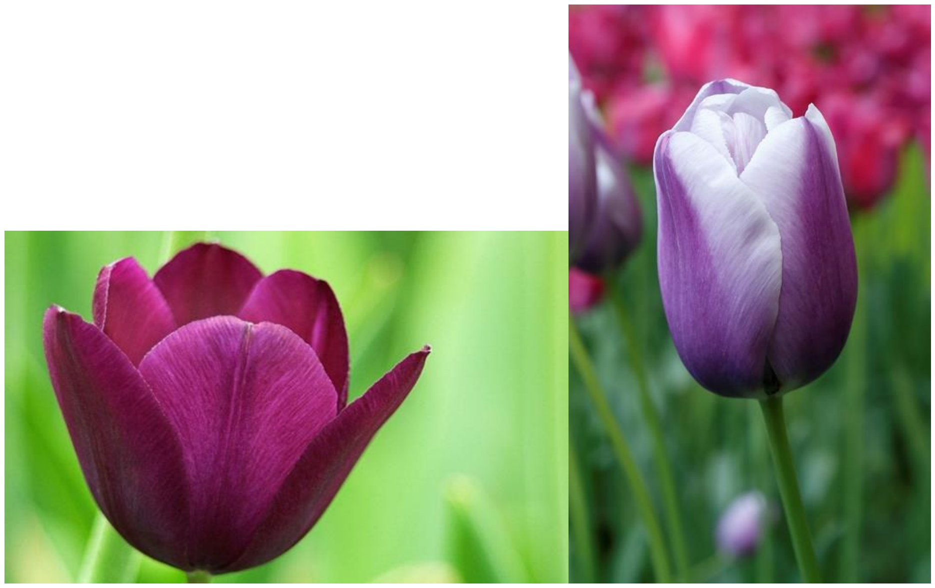


Figure 11. Tested images of Tulips; Guessed as Fritillary flowers

Fearing the worst, the team switched to test a sunflower shown below in **Figure 12.** Using this image, we were able to correctly classify the image as a sunflower.

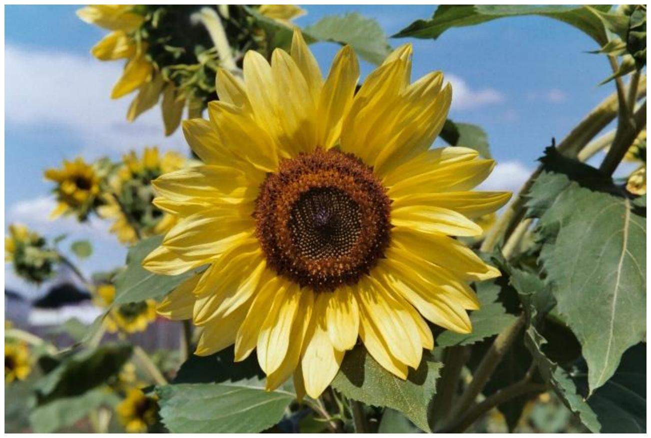


Figure 12. Tested images of Sunflower

Discussion and Conclusion

Our results show that as the number of classifications increases, the accuracy of our CNN goes down. Additionally it is increasingly difficult to train larger datasets, so it is hypothesized that finding the smallest classifications per model possible will both increase accuracy and reduce computational intensity. This idea of chunking is one that works well for humans and more than likely machines as well.

It was worthwhile to attempt to train using a GPU, and important to note that the required GPUs are very expensive, upwards of 10k dollars per unit. This pricepoint pushes out the retail trainer and requires the use of cloud based networks to train larger datasets.

Attached Code

```
%% Transfer Learning -1
% parpool(2); %% Use multiple CPUs/GPUs to train the model
% load image data from the folder Images/train
imagepath = fullfile('Images', 'train');
% use the image folder names as the classification labels
imds = imageDatastore(imagepath, 'IncludeSubfolders', true,'LabelSource',
'FolderNames');
% Anonymous Functions create simple functions without having to% create a
file for them
% Resize the input images to meet the input image size% requirements
imds.ReadFcn = @(loc)imresize(imread(loc),[227,227]);
% Split inputs into training and validation sets
[trainDS, valDS] = splitEachLabel (imds, 0.7, 0.3,'randomized');
% train the network
options = trainingOptions('sgdm', ...
    'MiniBatchSize',10, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',1e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData', valDS, ...
    'ValidationFrequency',3,...
    'Verbose',false, ...
    'Plots','training-progress');
%'ExecutionEnvironment', 'parallel';
trainedFlowerNet = trainNetwork(trainDS,flowerNet,options);

%% Transfer Learning -1 continued
I = imread("daisy.jpg");
I = imresize(I,[227 227]);
% Classify the image using flowerNet
% I is a 227 X 227 X3 flower image
label = classify(trainedFlowerNet,I)

%% Transfer Learning -2
% parpool(2); %% Use multiple CPUs/GPUs to train the model
% load image data from the folder Images/train
```

```
imagepath = fullfile('Images2', 'train');
% use the image folder names as the classification labels
imds =
imageDatastore(imagepath,'IncludeSubfolders',true,'LabelSource','FolderNames'
);
% Anonymous Functions create simple functions without having to
% create a file for them% Resize theinput images to meet the input image size
% requirements
imds.ReadFcn = @(loc)imresize(imread(loc),[227,227]);
% Split inputs into training and validation sets
[trainDS, valDS] = splitEachLabel (imds, 0.7, 0.3,'randomized');
% train the network
options = trainingOptions('sgdm', ...
    'MiniBatchSize',10, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',1e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData', valDS, ...
    'ValidationFrequency',3,...
    'Verbose',false, ...
    'Plots','training-progress');%, ...
    %'ExecutionEnvironment', 'gpu');
trainedFlowerNet = trainNetwork(trainDS,flowerNet2,options);

%% Transfer Learning -1 continued
I = imread("sunflower.jpg");
I = imresize(I,[227 227]);
% Classify the image using flowerNet
% I is a 227 X 227 X3 flower image
label = classify(trainedFlowerNet,I)
```