CS63 Spring 2022
Using a Neural Network to Recognise type of Flowers

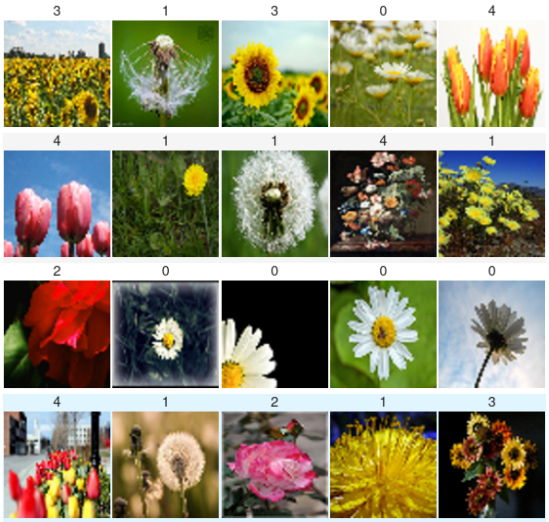
Delaney Hawkins, Yael Borger

1 Introduction

For this project we will be using a Neural Network to determine if an image containing a flower is a daisy, dandelion, rose, sunflower, or tulip. We will specifically be using a neural network made with many layers of varying types to create specialized feature detectors that are able to recognise key aspects of each type of flower so that it can be labeled appropriately.

2 Methods

We sourced our data set from Kaggle. The data set contained 764 daisy images, 1052 dandelion images, 784 rose images, 733 sunflower images,and 984 tulip images, all in folder labeled with the flowers. The data set contained a wide variety of flower images including black and white images, drawings of flowers, images in which only part of the flower was visible, images where the flower was obstructed by bugs, people holding flowers and some images that were seemingly included by accident without visible flowers in them. After they were imported we assigned each flower type to a numeric value accordingly:



0-Daisy
1-Dandelion
2-Rose
3-Sunflower
4-Tulip

Initially, the image data had to be downloaded onto the CS machines in Scratch, and then the images had to be imported themselves. Each image contained a different amount of pixel data, to address this we imported each image to be 56x56 pixels so the network could successfully train on data of the same dimension. During importing, we also created a parallel list with the correct labels added as the images were added to guarantee correct order and labeling to corresponding images. We then normalized these images by diving each pixel value by 255, the range of the image pixel colors, and sorted them into testing and training data to ensure the network was learning to generalize and not just memorize all of the data. We did this by adding 20 percent of the images from each type of flower to testing and the other remaining 80 percent into training.

The specifics for how the neural network was structured are included below:

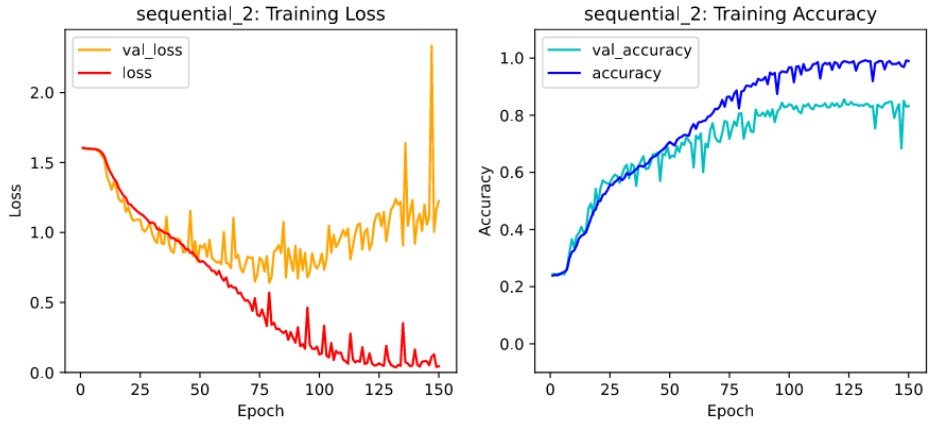
Layer (type)	Output Shape	Param #
conv1 (Conv2D)	(None, 53, 53, 30)	1470
pool1 (MaxPooling2D)	(None, 17, 17, 30)	0
conv2 (Conv2D)	(None, 17, 17, 40)	76840
dropout_12 (Dropout)	(None, 17, 17, 40)	0
conv3 (Conv2D)	(None, 17, 17, 40)	160040
pool2 (MaxPooling2D)	(None, 5, 5, 40)	0
conv4 (Conv2D)	(None, 5, 5, 20)	320020
dropout_13 (Dropout)	(None, 5, 5, 20)	0
conv5 (Conv2D)	(None, 5, 5, 20)	360020
flatten (Flatten)	(None, 500)	0
output1 (Dense)	(None, 20)	10020
dropout_14 (Dropout)	(None, 20)	0
output2 (Dense)	(None, 20)	420
dropout_15 (Dropout)	(None, 20)	0
output (Dense)	(None, 5)	105

Total params: 928,935
Trainable params: 928,935
Non-trainable params: 0

The network contains a total of 928,935 parameters all of which are trainable. This is a significantly higher number of parameters than expected, but we did achieve a decently good accuracy with it. The network has a combination of many convolutional layers of varying sizes to allow the network to recognize features of different sizes and colors, since we made them three dimensional. We also include pooling layers to help the network to minimize the parameter count by lowering the image pixels and forcing the network to combine different portions of the images into one main color of that section. We used dropout layers to try and compensate for the random images that have absolutely nothing to do with the flower data, and then flattening to bring the image into a simpler formatting with fewer parameters. As mentioned, there were random images that were not relevant to the data set, and if it were not so many we would have gone through and manually removed or relabeled the images that were not floral, but there were many images to sort through for that, and it was better to consider the option of additional dropout layers instead.
The network was trained with 80 percent of each collection of flowers and the test with the 20 percent remaining. Once there was a separation between the training and test data for each flower, the different lists of flowers were combined so all of the training sets for each flower were combined, and same with all of the test sets. Both the training and test sets had two lists: a list of images and a parallel list with the answers, both of which had to be shuffled in the same way to guarantee generalization instead of input pattern recognition. Training was ended when val accuracy plateaued and loss reached a near zero value, this occurred after 150 epochs.

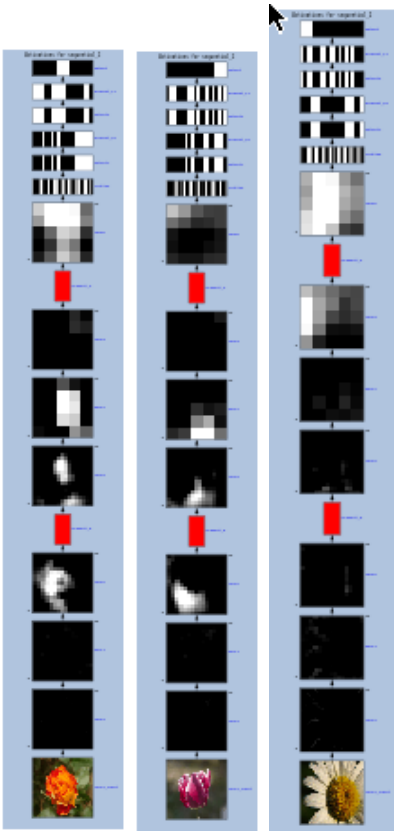
3 Results

We were able to achieve a val accuracy of 0.8324, an accuracy of 0.0.9899 with loss and val loss ending at 0.0443 and 1.2257 respectively.

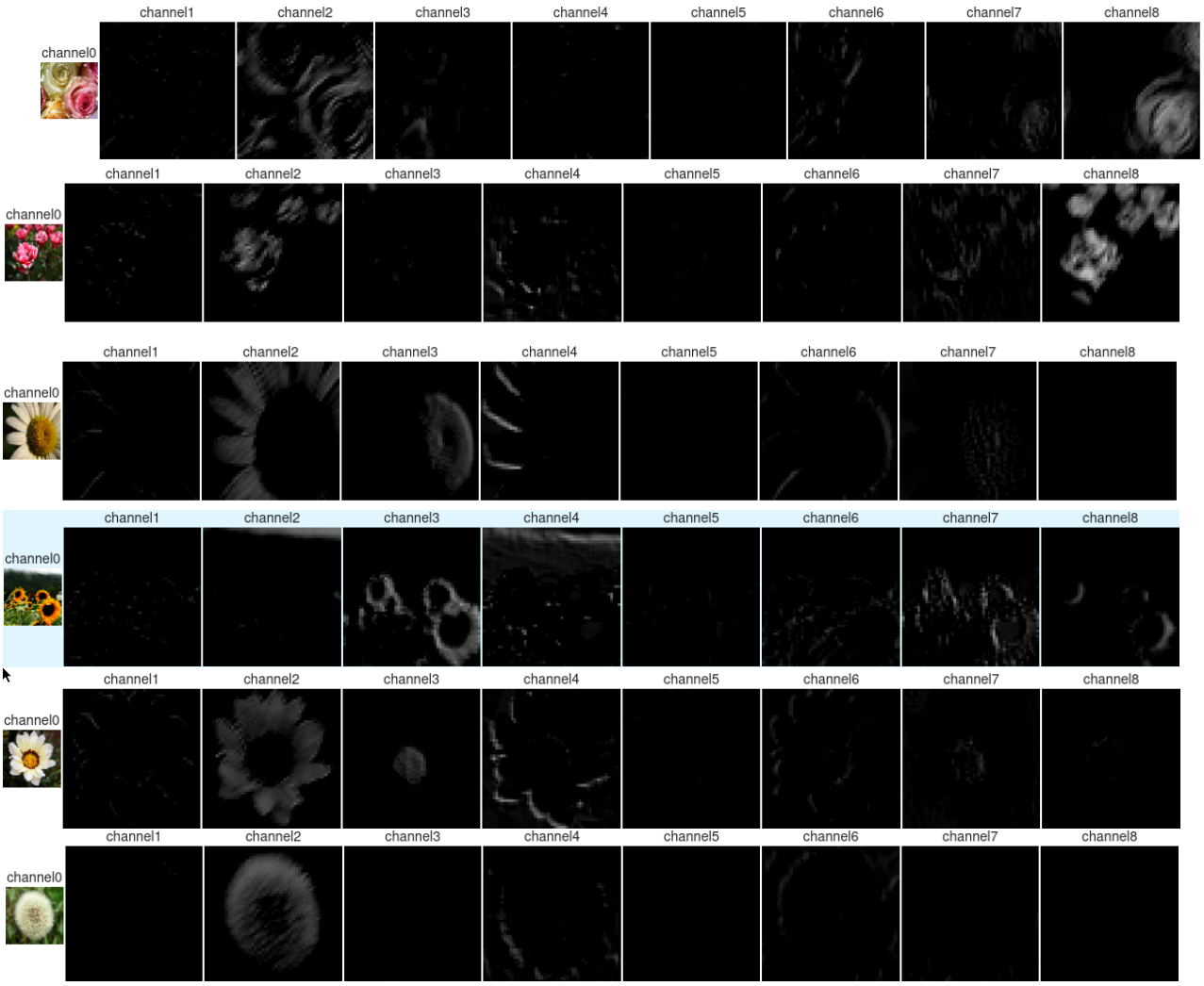



```
Epoch 150/150 loss: 0.044312786310911118 - accuracy: 0.9898609519004822 - val_loss: 1.2257075309753418 - val_accuracy: 0.8323699235916138
```

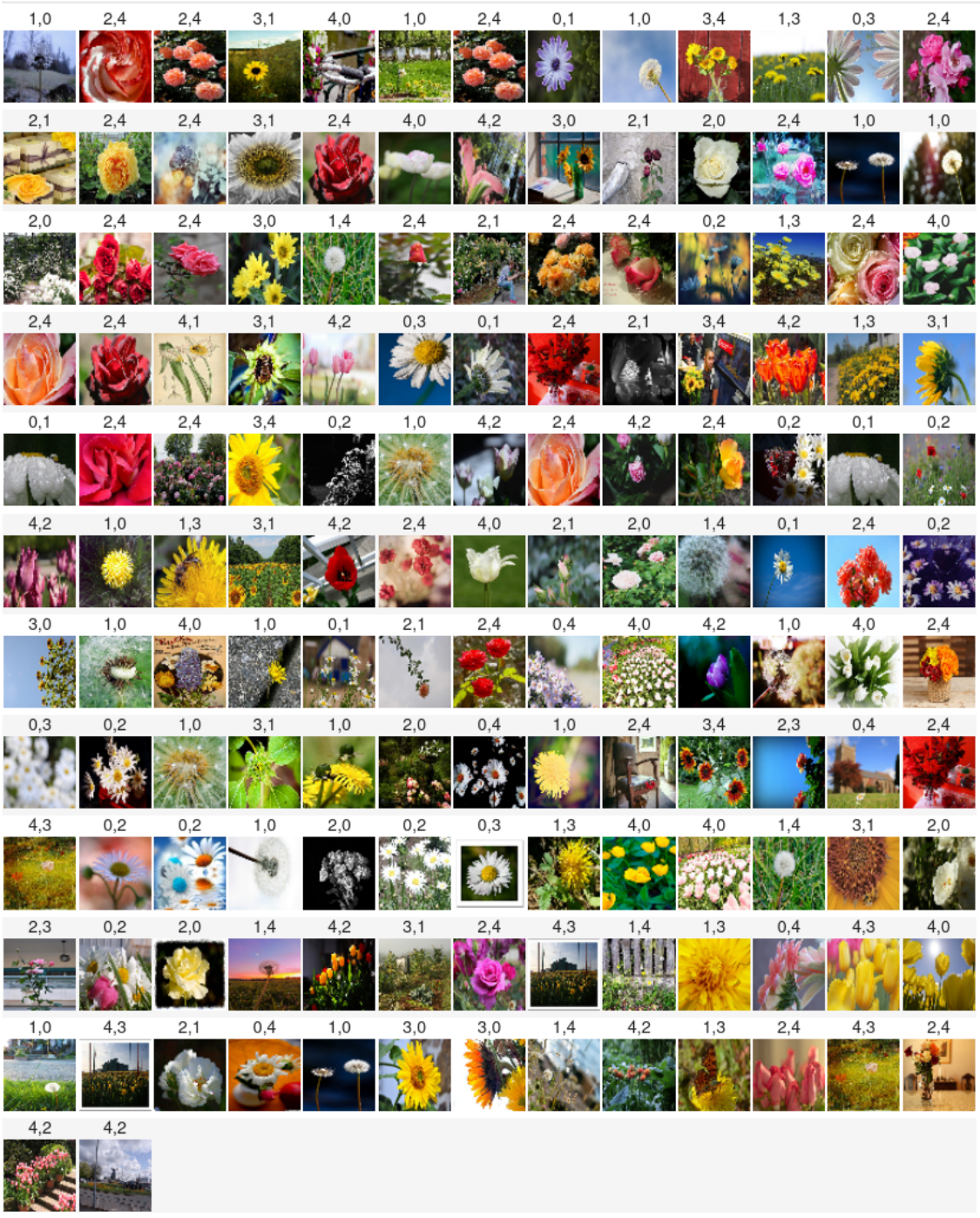
The following images where taken during training. They show the where the network focused most of its "attention" as it trained on the example image and how the weights percolated up throughout the network.



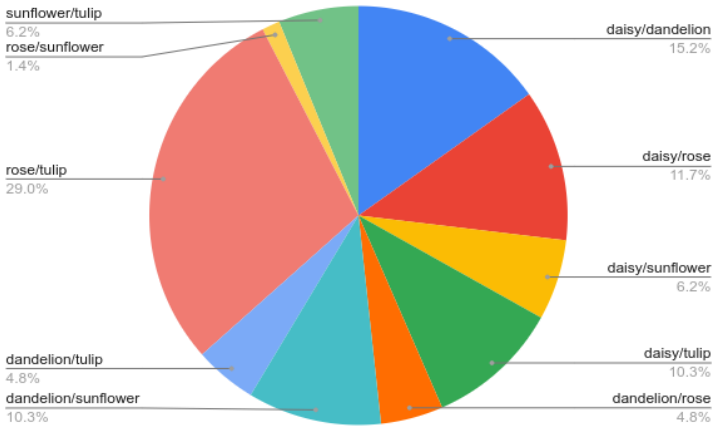
The network was successful in creating feature detectors, as visualized below, and the network appeared to do a pretty good job at locating where the flower was (channel 2 most notably) along with the flowers outline (channel 4 most notably). Additionally the network was often able to distinguish individual petals and a flower center(channel 3 most notably) when there was one.



Below are the images that the network labeled incorrectly along with what the flower actually was on the left and what the network guessed on the right:



4 Conclusions



While the network was pretty successful, we did not reach the 90 percent accuracy we had wanted, but we are able to see how the different layers detected certain shapes and light levels. We achieved 83 percent accuracy, and while this was below our goal of 90 percent accuracy, we still consider this to be a successful network in that it is still a high percentage, and we acknowledge that 90 percent was a relatively high goal. Some of the error we are encountering we believe to be attributed to some off topic images within the data set, such as the random people who had no flower elements in the images but were included in certain flower folders.

We predicted the network would have some trouble distinguishing between certain types of flowers that had similar features such as shape and color, tulips and roses, and sunflowers and dandelions for example. The pie graph contains the breakdown of which flowers the network most often confused. We can see that the network confused roses and tulips the most frequently, followed by daisies and dandelion both of which we predicted to be frequently confused. However, the third most frequently confused was daisies and roses which we were not expecting and are not entirely sure why there is such a high rate of confusion between those two.