­­­

Flower Recognition

Statement of work

Rui Tang | Step II

# Abstract

This Statement will show the outlines of flower recognition system, which can detect flower type in a picture and label it correctly.

# Problem Statement

Our team wants to develop an augmented reality glasses. The glasses can automatically identify items in life and mark them. As part of the function, we hope that this automatic identification system can correctly identify the type of flower.

# Data Requirement

As we mentioned above, we need a bunch of flower pictures with labels. Fortunately, we found the right data on Kaggle. There are 5 types of flowers and our team has initially processed the data.

# Algorithm

As a visual recognition system, we expect to use Convolutional neural network with TensorFlow.

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.[[1]](#footnote-1)

In this system, inputs are pixels in each picture, and the output will be the probability of each category. Loss evaluation function will be Categorical Cross entropy.



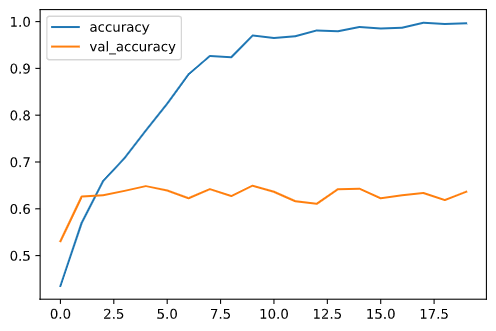
# Validation Process and Optimization

To avoid overfitting or underfitting, cross validation will be used. Firstly, we will split data into training set and test set randomly. After each epoch of training, the model will output the accuracy of both training set and test set.

After training, a curve diagram will be showed to indicate whether the model is overfitting or underfitting.

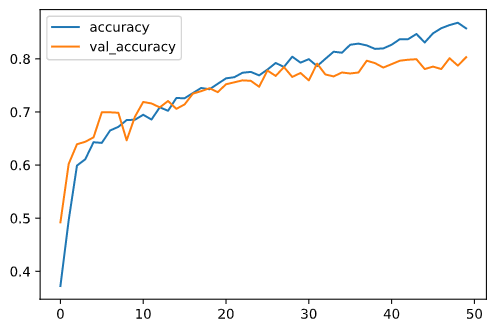
# Training without data processing

In this project, TensorFlow and Keras were used to train the model. In first version, data were divided into training dataset and testing dataset. Then with batch size 32 and epoch 20, the first model is trained. However, there was a serious overfitting phenomenon. Based on the following accuracy curve, before the training accuracy curve and validation accuracy curve cross, it is underfitting. After that, the validation accuracy stays still, but the training accuracy goes to nearly perfect.



# Training with Data Transformation

One efficient way to overcome overfitting problem is to find and input more data. However, there is another way to make more dataset. By translation, rotation, scaling, adding random noise, we can produce more data. Using these methods, we can significantly improve the overfitting problem, as shown in the figure below.



# Usage of Model

After training a decent model, we take advantage of OpenCV to process a video. In each frame, OpenCV will extract the image and resize it to fit the model data format and predict. When the model gets the result, we use OpenCV to insert a label and its precision into the frame.





# Deployment

This project offers 2 ways to deploy: integrated into another Python application or works independently.

1. For developers wanting to integrate this model into another Python application, like Flask or Django, you can create a instance of Processor, and call process\_img or process\_video function.
2. For developers or users who want to run separately, you can run it in command line. To see the detail of usage, click [here](../deployment/readme.md).

# Resources

<https://www.kaggle.com/alxmamaev/flowers-recognition>

1. <https://en.wikipedia.org/wiki/Convolutional_neural_network> [↑](#footnote-ref-1)