Lab Assignment 5

Notebook:	Machine Learning Lab		
Created:	18/11/2018 14:46	**Updated:**	19/11/2018 06:45
Author:	Prateek Chanda		

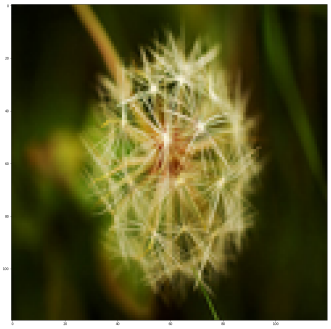
Classification using Convolution Neural Network

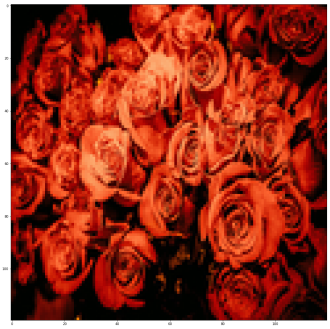
In this assignment, we had to classify flower images from https://www.kaggle.com/alxmamaev/flowers-recognition and try to predict flower classes through the help of Convolution Neural Network.

Dataset

The dataset consists of five classes of flower images, namely chamomile, tulip, rose, sunflower, dandelion. The images were all resized to 120x120 pixels before further processings.
The dataset was then shuffled to create separate training, test and development sets with a ratio of 90:10:10 respectively. This was done with the help of *sklearn.split* function into training, test and development sets of corresponding proportions.

Once we have the training set, we analyse the data at hand.



Training with the Convolution Neural Network

Convolution networks are used to recognize images by transforming the original image through layers to a class scores. CNN was inspired by the visual cortex. Every time we see something, a series of layers of neurons gets activated, and each layer will detect a set of features such as lines, edges. The high level of layers will detect more complex features in order to recognize what we saw.

We first add 2 convolution neural network layers using the relu as activation function and we finally add a top fully connected layer on top of it for classification.
In between the layers, we use Max Pooling.
This was implemented using *keras.*

Architecture of the CovNet

```
model = models.Sequential()
# We first add 2 convolution neural network layers using the relu as activation
function and we finally add
# a top fully connected layer on top of it for classification.
# In between the layers, we use Max Pooling.
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=
(img_size,img_size,3)))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(2,2))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(5, activation='softmax'))
```

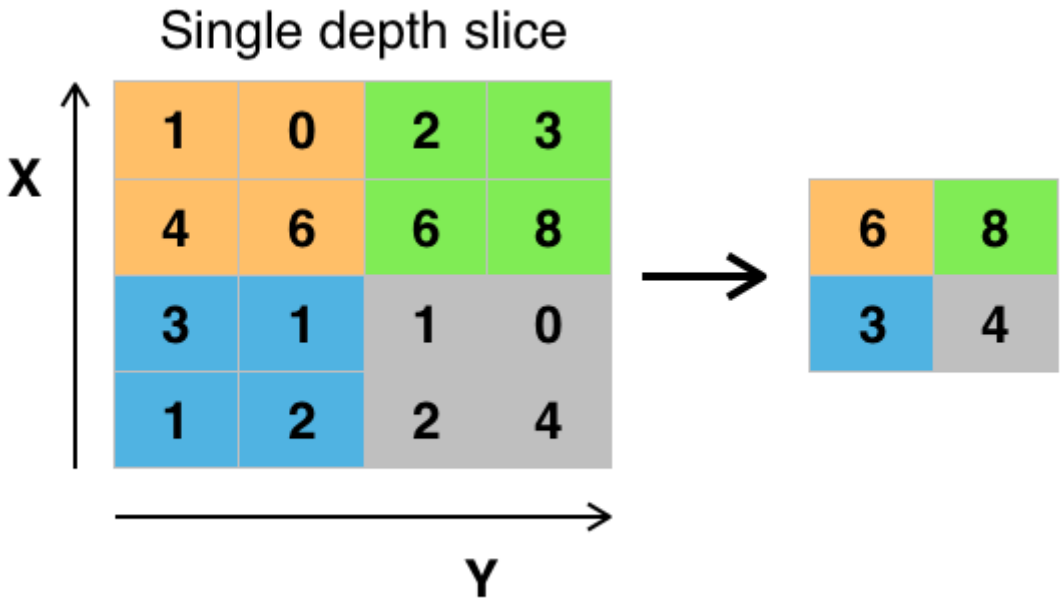
ReLU layer :

ReLU Layer applies an element-wise activation function max(0,x), which turns negative values to zeros (thresholding at zero). This layer does not change the size of the volume and there are no hyper-parameters.

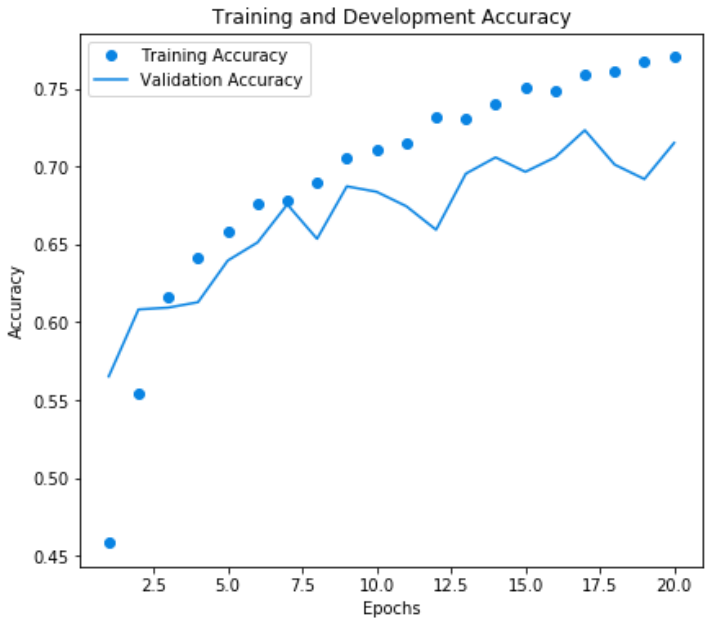
POOL layer:

Pool Layer performs a function to reduce the spatial dimensions of the input, and the computational complexity of the model. There are different functions such as Max pooling, average pooling. However, Max pooling is the most used type of pooling which only takes the most important part (the value of the brightest pixel) of the input volume.

Example of a Max pooling with 2x2 filter and stride = 2. So, for each of the windows, max pooling takes the max value of the 4 pixels.

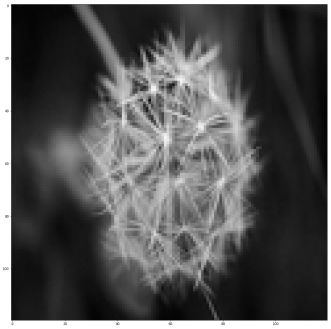


Test and Development Test Accuracy

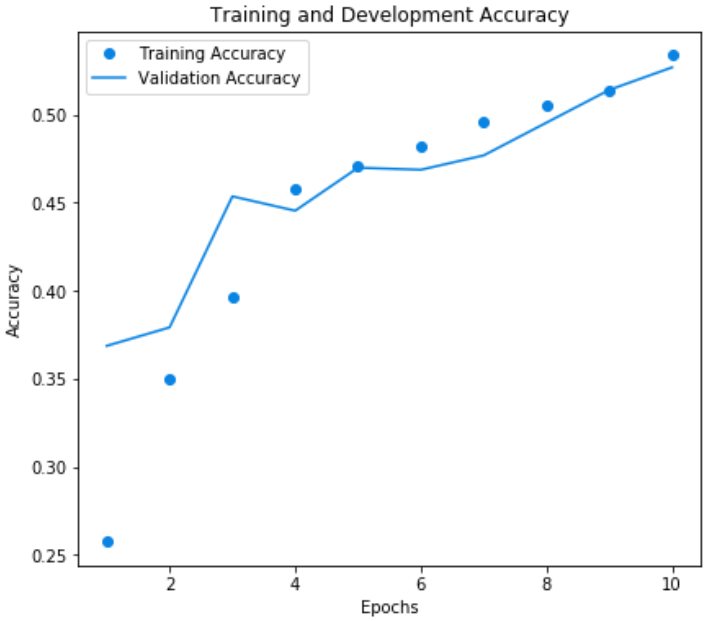


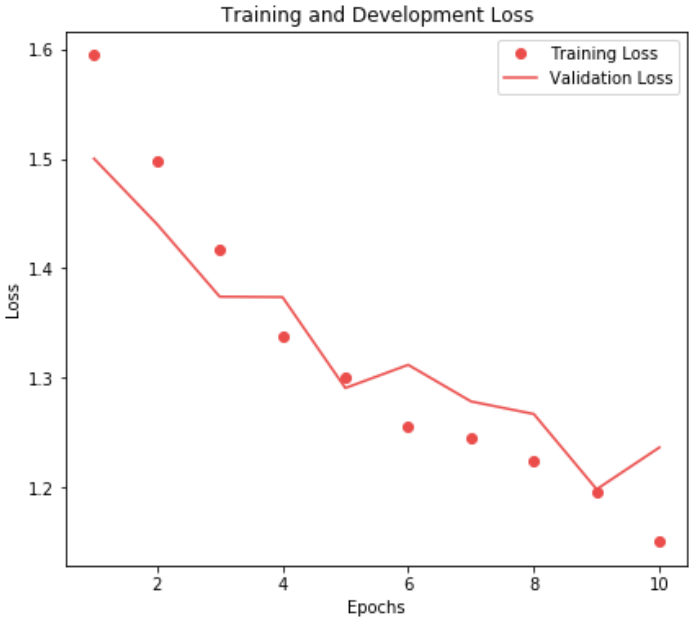
Training with grayscale images

Next, we convert the images into gray scale and build upon the convolution neural network , consisting of 3 layers and accompanied with Max Pooling Layers.

Again we check for the Training and Development Accuracy and Loss





We also predicted 10 image classes for both colored and gray-scaled images. For colored images, we had total of 2 mis-classification out of 10 and for gray scaled images, we had 4 mis-classifications.