Flower Dataset Classification using CNN

Sharon Mathew Sasi

Electrical and Computer Engineering
University Of Illinois - Chicago
Illinois,USA

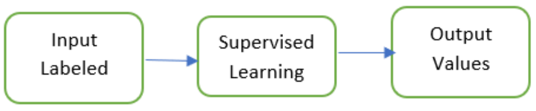
I. INTRODUCTION

This project is based on designing a convolutional neural network mode (CNN) to classify a flower dataset. A convolutional neural network (CNN) is a class of Artificial Neural Network (ANN), most applied to analyze visual imagery. CNNs are also known as Shift Invariant or Space Invariant Artificial Neural Networks. CNNs are regularized version of multilayer perceptron. A multilayer perceptron is a fully connected network, each neuron in one layer connected to the neuron in the next layer. The Convolutional Neural Network (CNN) has shown excellent performance in many computer vision and machine learning problems. Applications of CNN include image classification, image semantic segmentation, 2 object detection in images, etc.

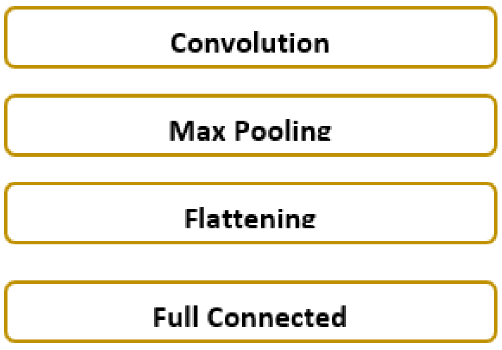
11. METHOD DESCRIPTIONS

Convolution Neural Network is a discriminative deep learning model which usually extract features for the dataset and also classify the features. CNN network is a combination of Linear filters and Non-Linear filters.

CNN can be one-dimensional or two-dimensional or three -dimensional depending on the use of the application. CNN is a supervised learning model, where the input dataset are labeled in order to train the model.



CNNs in today's use are very densely structured this makes the accuracy improvisation. For example, RESNET a dense CNN has 50m layers within it, but during early stages CNN has only few layers namely, Convolution, Max-Pooling, Flattening, Fully Connected.

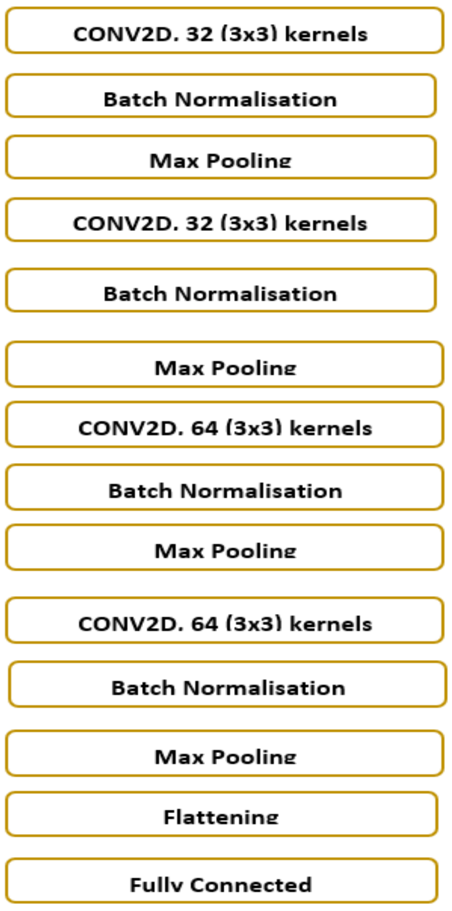


Convolution layer performs the normal convolution on the input data using kernel(filters) of different dimensions and various number. By doing so we can obtain the features of the data, as well as it reduces the dimensions of the input data, which helps in storage and improves the accuracy. Max Pooling is a convolution process where the Kernel extracts the maximum value of the area it convolves. Max Pooling simply says to the Convolutional Neural Network that we will carry forward only that information, if that is the largest information available amplitude wise.

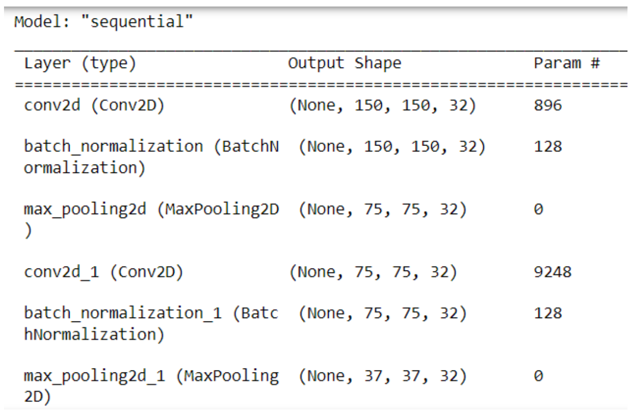
Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector. The flattened matrix is fed as input to the fully connected layer to classify the image.

Fully Connected Layer is simply, feed forward neural networks. Fully Connected Layers form the last few layers in the network. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

Dataset used here is a flower dataset which contains flowers of five different types, namely Roses, Dandelion, Tulip, Sunflower, and Daisy. The convolutional Neural Network is designed to classify the flowers when given in random as the input.
The CNN network constructed for the dataset is as below:



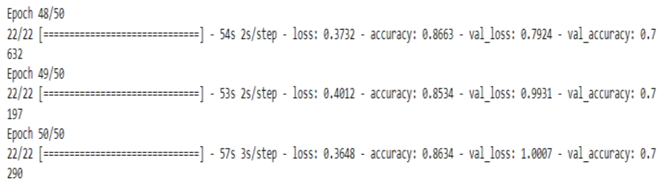
The given dataset is divided into three set, training set to train the data,to fit the model. Then the Validation set to tune the model and Test set is used to predict the input images.
The dataset used here is less in number so data augemntation is done before compiling. The data augmentation done here is horizontal flip, vertical flip, width shift range, height shift range, rotation and etc.
While compiling loss function used is categorical cross entropy, optimizer used is Adams from which accuracy is determined. After which the fitting is done and test sets are used to determine accuracy.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      896       
                                                                 
 batch_normalization (BatchN  (None, 150, 150, 32)     128       
 ormalization)                                                   
                                                                 
 max_pooling2d (MaxPooling2D  (None, 75, 75, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 75, 75, 32)        9248      
                                                                 
 batch_normalization_1 (Batc  (None, 75, 75, 32)       128       
 hNormalization)                                                 
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 37, 37, 32)       0         
 2D)                                                             
```

111. Experimental Results

The model was used to classify the flower dataset into five groups. The test flower dataset was fit into the model and the accuracy was determined. Since we used data augmentation it increased the number of the dataset used for the training. The accuracy was improvised after implementing data augmentation.
Using the model, with the training we obtained an accuracy of 0.8634 with iteration of 50(epoch).
Further using the validation and the test dataset we obtained an accuracy of 0.7298 in 50 epochs.



```
Epoch 48/50
22/22 [============================] - 54s 2s/step - loss: 0.3732 - accuracy: 0.8663 - val_loss: 0.7924 - val_accuracy: 0.7
632
Epoch 49/50
22/22 [============================] - 53s 2s/step - loss: 0.4012 - accuracy: 0.8534 - val_loss: 0.9931 - val_accuracy: 0.7
197
Epoch 50/50
22/22 [============================] - 57s 3s/step - loss: 0.3648 - accuracy: 0.8634 - val_loss: 1.0007 - val_accuracy: 0.7
290
```

Since data augmentation is used the accuracy is obtained is greater than without data augmentation.
Further this implementation than be done using Transfer learning. The feature detectors of other networks (RESNET, VGG etc) or other networks can be used and only the fully connected layers are optimized, by minimizing the loss function.
CNN is a deep learning network; it is Modeled to mimic the human brain ability to recognize the object. CNN is the main building block.

IV.Conclusion

In this report, Convolutional Neural Network is used to implement image recognition. This convolutional model can be used to implement any other data set for image recognition. Since we perform data augmentation before compiling the dataset improves the accuracy of recognition. Convolution Model helps improving the accuracy with less human supervisions.

V. Reference

[1] Image Classification using Convolutional Neural Network(CNN) and RNN by Patel Dhruv, Subham Naskar.

[2] Convolutional Neural Network(CNN) for Image Detection and Recognition by Rahul Chauhan Kama Kumar.

[3] Understanding of a convolutional Neural Network(CNN) by Saad Albawi, Saaad Al-Zawi.

[4] Advancement in Image Classification using CNN by Farhana Sultan, Abu Sufian, Paramartha Dutta.

[5] Impact of Fully Connected layer on performance of CNN for image classification by S.H. Shabbeer Basha, Sneha Mukherjee.

[6] Deep Convolution Neural Network for Image Deconvolution by Li Xu, Jimmy SJ Ren, Jiaya Jia

VI.Appendix

```
import os
import cv2
import tqdm as tqdm
print(os.listdir('flowers (2)/flowers'))
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns


import tensorflow as tf
import random as rn

import cv2
import numpy as np
from tqdm import tqdm
X=[]
Z=[]
IMG_SIZE=150
FLOWER_DAISY_DIR='flowers
(2)/flowers/daisy'
FLOWER_SUNFLOWER_DIR='flowers
(2)/flowers/sunflower'
FLOWER_TULIP_DIR='flowers
(2)/flowers/tulip'
FLOWER_DANDI_DIR='flowers
(2)/flowers/dandelion'
FLOWER_ROSE_DIR='flowers
(2)/flowers/rose’

def make_train_data(flower_type,DIR):
    for img in tqdm(os.listdir(DIR)):
```

```
def make_train_data(flower_type,DIR):
  for img in tqdm(os.listdir(DIR)):
        label=assign_label(img,flower_type)
        path = os.path.join(DIR,img)
        img = cv2.imread(path,cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        X.append(np.array(img))
        Z.append(str(label))
make_train_data('Daisy',FLOWER_DAISY_DIR)
print(len(X))
make_train_data('Sunflower',FLOWER_SUNFLOWER_DIR)
print(len(X))
make_train_data('Rose',FLOWER_ROSE_DIR)
print(len(X))
make_train_data('Dandelion',FLOWER_DANDI_DIR)
print(len(X))
make_train_data('Tulip',FLOWER_TULIP_DIR)
print(len(X))
fig,ax=plt.subplots(5,2)
fig.set_size_inches(15,15)
for i in range(5):
    for j in range (2):
        l=rn.randint(0,len(Z))
        ax[i,j].imshow(X[l])
        ax[i,j].set_title('Flower: '+Z[l])

plt.tight_layout()
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
Y=le.fit_transform(Z)
Y=to_categorical(Y,5)
X=np.array(X)
X=X/255
x_train_val,X_test,y_train_val,y_test=train_test_split(X,Y,test_size=0.25,random_state
=42)
np.random.seed(42)
rn.seed(42)

from keras.models import Sequential
from keras.layers import Dropout, Flatten,Activation
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization

from keras.layers import Dense
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape = (150,150,3), padding='same', activation =
'relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3), padding='same', activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3), padding='same', activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3), padding='same', activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1024, activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(5, activation = 'softmax'))
datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.1, # Randomly zoom image 
        width_shift_range=0.2,  # randomly shift images horizontally (fraction of 
total width)
        height_shift_range=0.2,  # randomly shift images vertically (fraction of total 
height)
        horizontal_flip=True,  # randomly flip images
        vertical_flip=False)  # randomly flip images


datagen.fit(x_train)

model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
History = model.fit_generator(datagen.flow(x_train,(y_train), batch_size=batch_size),
                              epochs = epochs, validation_data = (X_test,y_test),
                              verbose = 1, steps_per_epoch=x_train.shape[0] // 
batch_size)
```