IMAGE CLASSIFICATION USING ARTIFICIAL NEURAL NETWORKS AND FUZZY LOGIC

Team members: Prudhivi Sai Ganesh(RA2011003011057)

Bhargav Verma(RA2011003011106)

ABSTRACT

Computer vision is concerned with the automatic extraction, analysis, and understanding of useful information from a single image or a sequence of images. We have used Convolutional Neural Networks (CNN) in automatic image classification systems. In most cases, we utilize the features from the top layer of the CNN for classification; however, those features may not contain enough useful information to predict an image correctly. In some cases, features from the lower layercarry more discriminative power than those from the top. Therefore, applying features from a specific layer only to classification seems to be a process that does not utilize learned CNN's potential discriminant power to its full extent. Because of this property we are in need of fusion of features from multiple layers. We want to create a model with multiple layers that will be able to recognize and classify the images.

INTRODUCTION

Image classification with SIFT and Neural network We roughly categorize the photos extracted from Instagram of Huangshan City, China into 5 categroies: Architecture, Cloud, Food, Pine, Hiking.Then, we manually label 100 images for each of the 5 categories, for a total of 500 images. With this set at hand, we randomly split the data with keeping 80% of the data as training data and 20% of data as testing data.

ARCHITECTURE

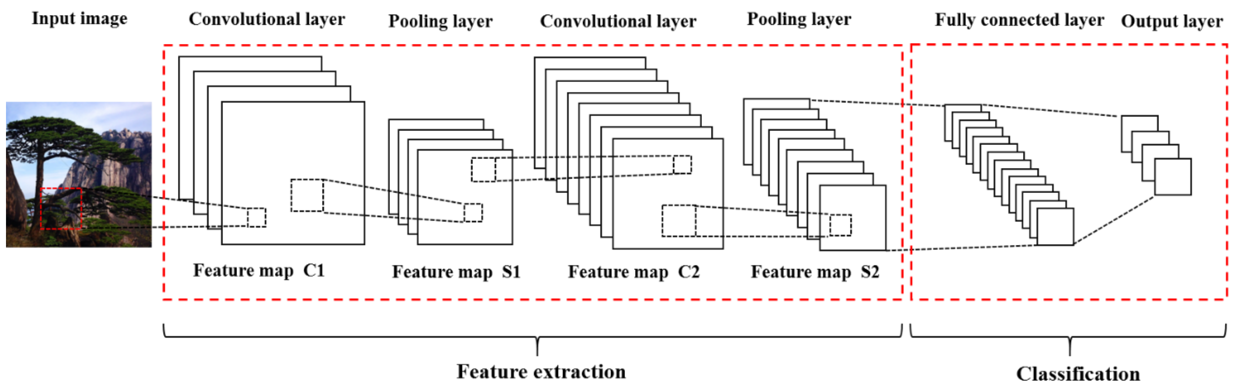
The Architecture category includes the photos of ancient architecture, like ancient residential houses, memorial archways, shrines, bridges, and street;

The Cloud category includes the photos of cloud, sunset, sunrise, and sky;

The Food category includes the photos of food, drinks, recipes;

The Pine category includes the photos whose main object is pine tree;

The Hiking category includes the photos taken while people are hiking.



SIFT + Bag-of-words + Kmeans + SVC

First, we use the scale-invariant feature transform (SIFT) algorithm to extract a set of key features for images in the dataset; The extracted key features are invariant to image translation, scaling, and rotation, as well as partial invariant to illumination changes and robustness to geometric distortions. Then, k-means clustering algorithm is used to cluster the extracted key feature vectors, which provides a set of codewords to form the dictionary of our bag-of-words (BOW) representation. Next, for the extracted SIFT features of an image, we assign each feature to the closest BOW, which allows us to express the image as a histogram of BOW.

Finally, we train the corresponding histograms of labelled images with Support Vector Machine (SVM) algorithm, to get the SVM image classifier. For the unlabeled images, we classify the BOW presentations with the SVM image classifier. The output is a vector of probability classified to each category and the category with maximum probability is the label of the unlabeled image.

Convolutional Neural Network

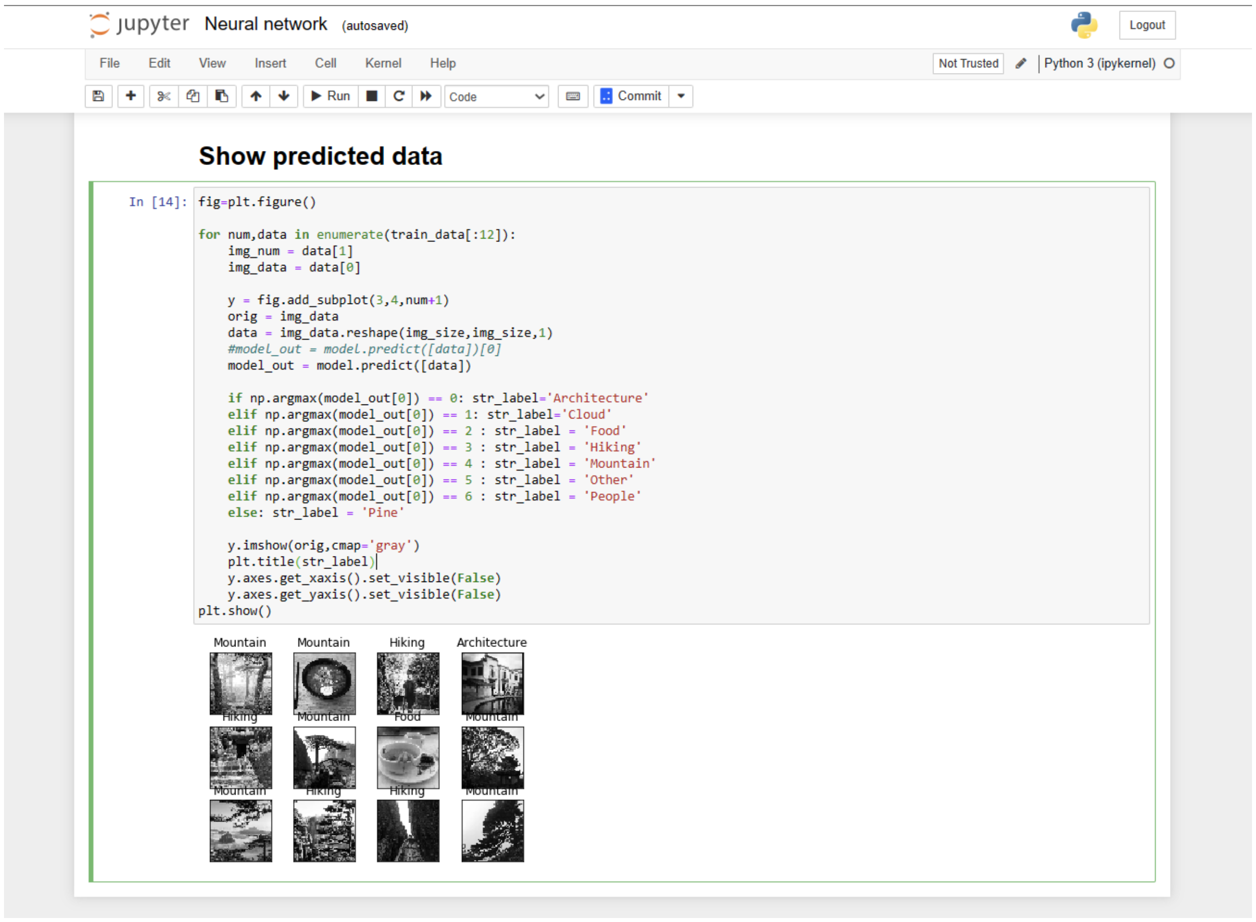
Neural networks and a set of derivative algorithms have long been implemented for image classification. A neural network is composed of multiple layers, and each layer has a various number of neurons with trainable weights and biases. All the neurons are fully connected to the neurons in previous and post layers. The first layer is input layer, whose input is the input data. The last layer is the output layer, whose output is the predicted result. Other layers are called hidden layer processing and passing the information from the previous layer to post layer.

CNNs are a category of Neural Networks that have demonstrated effectiveness in various fields such as image recognition and video analysis. The CNN architectures consist of three basic components, which are convolutional layer, pooling layer, and fully connected layer . Each layer is composed of a certain number of feature maps, which indicates a level of feature representation. The convolutional layer and pooling layer together act like a hidden layer in traditional neural network. The convolutional layer consists of multiple convolution units, which acts like the weight-sharing version of the traditional neural network and is designed to learn filtering-based feature representation of the inputs. It can reduce the model complexity thus making the network easier to train. The pooling layer is often inserted between two successive convolutional layers and is designed for non-linear down-sampling to achieve invariance to some degree. The fully connected layer can be viewed as the classifier.

CODE

```
import tflearn

from tflearn.layers.conv import conv_2d, max_pool_2d

from tflearn.layers.core import input_data, dropout, fully_connected

from tflearn.layers.estimator import regression

from tflearn.data_preprocessing import ImagePreprocessing

from tflearn.data_augmentation import ImageAugmentation


#numpy, plt

import numpy as np

import matplotlib.pyplot as plt

import matplotlib
import os
import cv2
from numpy import *

# SKLEARN
from sklearn.utils import shuffle
from sklearn.cross_validation import train_test_split
train = r'traindata'    #path of folder of images
train_names = os.listdir(train)
img_size = 50   # image size
lr = 1e-3   # learning rate
MODEL_NAME = 'huangshan-{}-{}.model'.format(lr,'CNN')
def imlist(path):
    """
    The function imlist returns all the names of the files in
    the directory path supplied as argument to the function.
    """
    return [os.path.join(path, f) for f in os.listdir(path)]
def create_train_data():
    class_id = 0
    training_data = []
    for train_name in train_names:
        dir = os.path.join(train, train_name)
        class_path = imlist(dir)
        for path in class_path:
            img = cv2.resize(cv2.imread(path,cv2.IMREAD_GRAYSCALE), (img_size,
img_size))
            img = img / 255.0
            training_data.append([np.array(img).astype('float32'),
np.array(label_img(class_id))])
        class_id+=1
    training_data = shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data
train_data = create_train_data()
#CNN MODEL
img_prep = ImagePreprocessing()
img_prep.add_featurewise_zero_center()
img_prep.add_featurewise_stdnorm()

# Real-time data augmentation
img_aug = ImageAugmentation()
img_aug.add_random_flip_leftright()
img_aug.add_random_rotation(max_angle=25.)

# Building convolutional convnet
convnet = input_data(shape=[None, img_size, img_size, 1],
data_preprocessing=img_prep,
                     data_augmentation=img_aug, name='input')
```

```
convnet = conv_2d(convnet, 32, 3, activation='relu')

convnet = max_pool_2d(convnet, 2)


convnet = conv_2d(convnet, 64, 3, activation='relu')

convnet = max_pool_2d(convnet, 2)


convnet = conv_2d(convnet, 32, 3, activation='relu')

convnet = max_pool_2d(convnet, 2)


convnet = conv_2d(convnet, 64, 3, activation='relu')

convnet = max_pool_2d(convnet, 2)


convnet = fully_connected(convnet, 1024, activation='relu')

convnet = dropout(convnet, 0.8)


convnet = fully_connected(convnet, 8, activation='softmax')

convnet = regression(convnet, optimizer='adam', learning_rate=lr,
loss='categorical_crossentropy', name='targets')


model = tflearn.DNN(convnet, tensorboard_dir='log')
```



Show predicted data

```
In [14]: fig=plt.figure()

         for num,data in enumerate(train_data[:12]):
             img_num = data[1]
             img_data = data[0]

             y = fig.add_subplot(3,4,num+1)
             orig = img_data
             data = img_data.reshape(img_size,img_size,1)
             #model_out = model.predict([data])[0]
             model_out = model.predict([data])

             if np.argmax(model_out[0]) == 0: str_label='Architecture'
             elif np.argmax(model_out[0]) == 1: str_label='Cloud'
             elif np.argmax(model_out[0]) == 2 : str_label = 'Food'
             elif np.argmax(model_out[0]) == 3 : str_label = 'Hiking'
             elif np.argmax(model_out[0]) == 4 : str_label = 'Mountain'
             elif np.argmax(model_out[0]) == 5 : str_label = 'Other'
             elif np.argmax(model_out[0]) == 6 : str_label = 'People'
             else: str_label = 'Pine'

             y.imshow(orig,cmap='gray')
             plt.title(str_label)
             y.axes.get_xaxis().set_visible(False)
             y.axes.get_yaxis().set_visible(False)
         plt.show()
```

Conclusion:

we used Convolutional Neural Networks (CNN) for image classification using images . This data sets used both and training and testing purpose using CNN. It provides the accuracy rate 98%. Images used in the training purpose are small and Grayscale images. The computational time for processing these images is very high as compare to other normal JPEG images. Stacking the model with more layers and training the network with more image data using clusters of GPUs will provide more accurate results of classification of images. The future enhancement will focus on classifying the colored images of large size and its very useful for image segmentation process.

REFERENCES:

Carlos Silva, Daniel Welfer, Francisco Paulo Gioda, Claudia Dornelles," Cattle Brand Recognition using Convolutional Neural Network and Support Vector Machines ", IEEE Latin America Transactions, vol. 15, no. 2, pp. 310-316, 2017.

A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in Proc. Neural Inf. Process. Syst., Lake Tahoe, NV, USA, 2012, pp. 1106–1114.

D. Lunga, S. Prasad, M. M. Crawford, and O. Ersoy, "Manifold-learningbased feature extraction for classification of hyperspectral data: review of advances in manifold learning," IEEE Signal Process. Mag., vol. 31, no. 1, pp. 55–66, Jan. 2014.