COMP3211 Project Proposal
Fall 2017-18

Chiu Ho Yeung (20186719), Dhaval Shah (20341474), Chak Ho Wong (20265735), Mok Kai Long(20271203)

1. Introduction

Nowadays, with advancements in AI and computer vision, object recognition has become very popular with many different applications of the idea. Our project is related object classification as we plan to make a Rice Classifier. Our project makes use of a library called Keras to help us with our project. Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow [1]. We use Keras because it has many functionalities as it can support convolutional neural network (CNN), recurrent networks (RN) and a combination of the both. We will mainly be using CNN for our project. And is very user-friendly which makes it easier to write the code.

2. Convolutional Neural Network

Three decisions we have to make for designing the architecture.
1. What do we do for arranging the layers?
2. Which layers we choose to use?
3. How many neurons we want to use in each layer?

For two or more convolutional layers, we need the images from all classes to feed the layers. After the conventional layers, the output was flattened and we have to add all fully connected layer in the end. The second fully connected layer has distinct output which indicate the probability of an image of different objects.

Figure 1: Structure of Neural Network
3. Rice Classifier

3.1 Problem description
Rice is a staple food of many cultures and there are many different ways you can cook it. It can be cooked really simply or it can be a full fancy meal. Our project takes in pictures of food or any other object from the user and finds out whether or not the entered picture is about rice. Our project is inspired from a user on GitHub which can be found on https://github.com/hayzamjs/Hotdog-Classification [2]. However we believe that the method used to collect data and process the images was not clear enough, and found the training part of the CNN to be more complicated than necessary.

Therefore, our work focuses on improving data collection and processing methods and re-implementing the algorithm to train our network.

3.2 Structure of program

1) Data collection algorithm (getData.py)
As we’ll need thousands of images to be able to classify whether or not the image is about rice, we use these functions to save the photos from the given links.

```python
def store_raw_images(paths, links):
    global pic_num
    for link, path in zip(links, paths):
        if not os.path.exists(path):
            os.makedirs(path)
            image_url = str(urllib.request.urlopen(link).read())
            pool = ThreadPool(8)
            pool.starmap(loadImage, zip(itertools.repeat(path), image_url.split('
'), itertools.count(pic_num)))
            pool.close()
            pool.join()

def loadImage(path, link, counter):
    global pic_num
    if pic_num < counter:
        pic_num = counter+1;
    try:
        urllib.request.urlretrieve(link, path+'/'+'str(counter)+'*'.jpg')
        img = cv2.imread(path+'/'+'str(counter)+'*'.jpg')
    except:
        if img is not None:
            cv2.imwrite(path+'/'+'str(counter)+'*'.jpg', img)
            print(counter)
```

This is the main function of our getData, here we use links from ImageNet [3], which has thousands of images of all different types of objects. We search for what we want and use the download link provided by ImageNet.

```python
def main():
    links = [
        'http://www.image-net.org/api/text/imagenet.synset.geturls?wnid=876877289',
    ]
    paths = ['White rice', 'polished rice', 'Rice', 'Brown rice', 'Chickpea and rice', 'Risotto, Italian rice', 'Fried rice, Chinese fried rice', 'Spanish rice', 'Macaroni', 'People', 'Noodles', 'Nose']
    store_raw_images(paths, links)
```
2) Processing our data (trainingCNN.py)
We converted all the images obtained from getData.py into gray scale to ensure that the colour doesn't affect the outcome of our model. The algorithm for gray scale conversion was found online at MathWorks [5].

```python
def toGray(images):
    images = 0.2989*images[:,:,0] + 0.5870*images[:,:,1] + 0.1140*images[:,:,2]
    return images

def preprocessData(images):
    grayImages = toGray(images)
    return normalizeImages(grayImages)
```

3) Implementing CNN (trainingCNN.py)
Our CNN has three convolutional layers and two fully connected layers. It is based of a model from user comma.ai on GitHub [4].

```python
def kerasModel(inputShape):
    model = Sequential()
    model.add(Convolution2D(8, 5, 5, border_mode='valid', input_shape=inputShape))
    model.add(Convolution2D(16, 3, 3))
    model.add(Convolution2D(32, 3, 3))
    model.add(Flatten())
    model.add(Dense(240))
    model.add(Activation('relu'))
    model.add(Dense(120))
    model.add(Dense(2))
    return model
```

4. Innovation

Many fundamental AI methods were taught in class however, our project uses more complex models which are widely used in daily life. We have also redesigned the structure of the CNN of the project that inspired us to build a more user friendly algorithm.

5. Conclusion

With our improved way to collect images, the classifier can get the result much faster than before. On average the accuracy is similar compared to the former model. We are confident to conclude that we improved the classifier and there are a lot of potential in this topic.
6. Further improvements

Our model can also be used to classify different types of food however you’d need to collect images of all types of food which would be very tedious work. Furthermore, we could improve this project by outputting nutritional values of the food. This could be done by detecting multiple food items in the photo, and calculating the nutritional value of each item. Crowdsourcing would help make this easier, for example, a user inputs a photo taken by him/her and state what the food is and what it's nutritional value is. After sometime, human supervision could be included to make sure that the photos inputted by the user is actually what they say it is.

Reference