Project Overview

Currently there are over 170,000,000 food photos[1] on social media sites but searching and discovering data is still a manual process via tags. The aim of the project is to investigate image classification using convolutional neural networks as a means to automatically tag food within images.

Problem Statement

Modern smartphones have lead to an explosion of images on the internet. In recent years the number of photos uploaded to social media sites have grown exponentially to the rate where millions are now uploaded daily onto the internet. The increase in data however brings new challenges to the users and companies.

Organising and search images have become more challenging as photos are lost in a mess of unstructured data dumps on the net, images unlike documents cannot be searched without meta data. Social media and photo sharing companies have allowed users to manually tag or label photos but this process is usually tedious and laborious, a more automated solution was needed.

Metrics

//TODO

Data Exploration

If a dataset is present, features and calculated statistics relevant to the problem have been reported and discussed, along with a sampling of the data. In lieu of a dataset, a thorough description of the input space or input data has been made. Abnormalities or characteristics about the data or input that need to be addressed have been identified.

//Similar to proposal

Exploratory Visualization

A visualization has been provided that summarizes or extracts a relevant characteristic or feature about the dataset or input data with thorough discussion. Visual cues are clearly defined.

// Perhaps a table of the number of images per label

[1] https://www.business.com/articles/food-photo-frenzy-inside-the-instagram-craze-and-travel-trend/

Algorithms and Techniques

In the last decade the rise of available data and increased computing power lead to companies investing in machine learning. These techniques allowed companies to perform analysis on large amounts data that would otherwise be impossible to do manually, one of which was called deep learning.

Deep learning is a series of techniques which extracted features from datasets to solve problems such as classification, optimisation or decision making. The term deep learning refers to the layout of the model which includes an input layer, multiple hidden layers and a final activation layer. The more hidden layers that a model has, the deeper it is going to extract features. Each layer consists of a series of nodes which resembles the neurons within a brain and hence the name neural network.

Convolutional neural networks

Convolutional neural networks (CNN) were specifically designed to extract features from images. Unlike multi layered perceptrons in other neural networks, CNN takes a multi dimensional input thus allowing it to extract features with spatial awareness. The input can also be comprised of multiple channels which can be used to represent colours e.g. red, green blue.

After the input layer, the model consists of the hidden layer(s) and the activation layer. The hidden layers comprises of convolutional layers which extract features from the previous layer and pooling layers which compresses the data from the previous layer. Both uses a sliding frame for the calculations and there are options to pad the data if the frame goes beyond the bounds of the input.

The data flows through the model, calculating the weights of the nodes as it reaches to the activation layer which produces the final result. Once the result is produced, the margin of error is calculated and fed backwards in a technique known as back propagation. The weights of the nodes are updated in the process and the margin of error reduced, several iterations are performed until the weights do not change any more.

When training our model it is important follow the cross validation process. Cross validation enables us to split the dataset into three parts: training data, validation data and testing data. During the training process training data is used as input to train the model whilst the validation dataset is used to calculate the accuracy of the model in each iteration. Testing data is kept separate from the training process and used to calculate the final accuracy of the model.

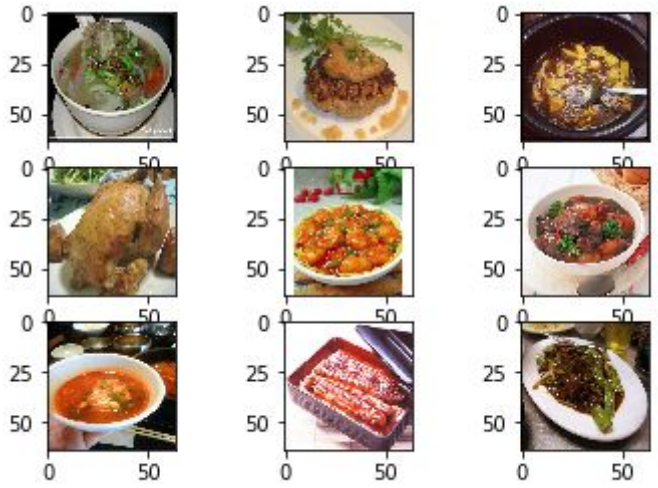
The model will be compiled with RMSprop as the optimizer which adjusts the rate decay with the change in the gradient as it approaches the optimum. Categorial entropy loss will be used as the loss function which is more performant for back propagation over means squared error.

If the accuracy of the results are below the target specified earlier, transfer learning will be used for feature extraction from a well known CNN. This will require the weights of an existing CNN to be loaded before applying my model as the final layers. The weights of the exiting CNN will be frozen as the my model is training.

Data Augmentation

Data is the key component to the success of any deep learning classifier and data augmentation is a technique that allows us to build models with little data. The image augmentation process involves taking the existing training data and making copies of it after applying a series of transformations on it. This helps generalize the existing training set and the extra data helps prevent overfitting in the model. Here are examples of the augmentation process on our training data:

No Augmentation



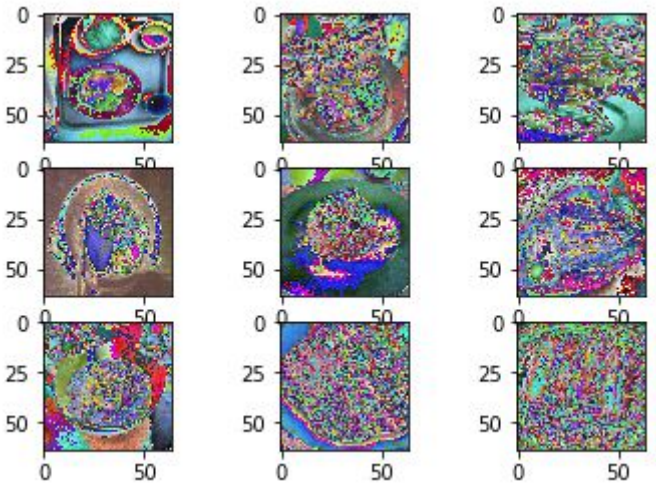
Random Rotations

Random Flips

Random Shifts

Feature Standardization

This is applied to ensure they have the properties of a standard normal distribution of mean 0 and standard deviation of 1. This is important to ensure the data is not affine transformed invariant i.e. there's no linear function that can transform one image onto another.



ZCA Whitening

The ZCA whitening process is applied to reduce redundancies in the data and find features that strong differentiators for the classification process. Hence data that are highly correlated with each other will be merged together (in PCA this will be an eigenvector). In this project we will choose to use ZCA over PCA as it retains the spatial qualities of the original image during the whitening[2].

Data Preprocessing

The dataset I have chosen is UECFOOD-256[3] which was used for research in The University of Electro-Communication in Tokyo and is free to use for non-commercial activities. The dataset has 256 labels for different kinds of dishes and each folder contains the images corresponding to the numerical value of the label. For example, folder 1 corresponds to rice.

The mapping between numbers and labels were stored in a text file on one line which had to be manually edited to be comma delimited. Once properly formatted, the labels can be read by pandas.

Cross validation was performed on the images to split into training (70%), validation (20%) and testing (10%) sets. This was performed using the sklearn libraries and running the cross validation function twice. The images were then transferred to the sub folders train, valid, test within food-images.

---

[2] Learning Multiple Layers of Features from Tiny Images

[3] http://foodcam.mobi/dataset256.html

Implementation

Once the data has been pre-processed we can begin the experiments

Benchmark

The benchmark will be the simplest CNN on the smallest pixel size we will experiment with consisting of one convolutional layer.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 64, 64, 16)        208
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 32, 32, 16)        0
_________________________________________________________________
global_average_pooling2d_6 ( (None, 16)                0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               4352
=================================================================
Total params: 4,560.0
Trainable params: 4,560.0
Non-trainable params: 0.0
_________________________________________________________________
Pixel size 64, channels 3, number of conv layers 1, number of pool layers 1
```

```
Checkpoint at basic.64.conv.1.pool.1.hdf5
Train on 22402 samples, validate on 5730 samples
Epoch 1/5
22380/22402 [============================>.] - ETA: 0s - loss: 5.5109 - acc: 0.0226Epoch 00000: val_loss improved from inf to
5.48188, saving model to saved_models/basic.64.conv.1.pool.1.hdf5
22402/22402 [==============================] - 37s - loss: 5.5109 - acc: 0.0226 - val_loss: 5.4819 - val_acc: 0.0229
Epoch 2/5
22380/22402 [============================>.] - ETA: 0s - loss: 5.4736 - acc: 0.0234Epoch 00001: val_loss improved from 5.48188
to 5.45531, saving model to saved_models/basic.64.conv.1.pool.1.hdf5
22402/22402 [==============================] - 38s - loss: 5.4737 - acc: 0.0233 - val_loss: 5.4553 - val_acc: 0.0229
Epoch 3/5
22380/22402 [============================>.] - ETA: 0s - loss: 5.4449 - acc: 0.0239Epoch 00002: val_loss improved from 5.45531
to 5.42683, saving model to saved_models/basic.64.conv.1.pool.1.hdf5
22402/22402 [==============================] - 37s - loss: 5.4447 - acc: 0.0239 - val_loss: 5.4268 - val_acc: 0.0232
Epoch 4/5
22380/22402 [============================>.] - ETA: 0s - loss: 5.4131 - acc: 0.0240Epoch 00003: val_loss improved from 5.42683
to 5.39175, saving model to saved_models/basic.64.conv.1.pool.1.hdf5
22402/22402 [==============================] - 37s - loss: 5.4132 - acc: 0.0240 - val_loss: 5.3918 - val_acc: 0.0234
Epoch 5/5
22380/22402 [============================>.] - ETA: 0s - loss: 5.3768 - acc: 0.0236Epoch 00004: val_loss improved from 5.39175
to 5.35531, saving model to saved_models/basic.64.conv.1.pool.1.hdf5
22402/22402 [==============================] - 37s - loss: 5.3768 - acc: 0.0236 - val_loss: 5.3553 - val_acc: 0.0241
Test accuracy: 2.5437%
CPU times: user 7min 14s, sys: 1min 59s, total: 9min 13s
Wall time: 3min 12s
```