# Capstone Proposal - Food Classifier

Wing Hym Liu

## Domain Background

Millions of images are uploaded to the internet daily and with the explosion of accessible data one problem still remains to be solved confidently by any of the major companies. The problem is image classification and despite a growing number of companies providing this service the accuracy of the services are overall still no match for the human eye.

Deep learning is a subset of machine learning techniques that are based on feature learning and in recent years have been used heavily in the machine learning industry including image classification. This is a series of techniques that extract useful representations from datasets and is often heard with neural networks which is a model which loosely represents the neurons within the human brain. A neural network consists of a series of connected nodes which each takes a weighted input and produces an output. The accuracy of the neural network is improved by updating its weights during training through a technique known as backpropagation.

Although neural networks such as multi layered perceptrons have had success in the past with image classifications, they suffer from a major problem with considering the spatial structure of the data. Hence, pixels that are close to each in the image have the same relevance as pixels far away.

Convolutional neural networks solves these problems by using a 3D representations of the image rather than a one dimensional input that MLPs take. The representation would consists of channels (in image recognition this could be the 3 colours: red, green and blue) and each channel would be a 2D matrix. The channels are passed through different types of hidden layers before the final fully connected layer which produces the final result for the classification, an example would be softmax. Hidden layers such as pooling layers takes a convolutional layer as input and can be used to reduce the size of the dataset to avoid overfitting and the computational complexity of the CNN. Examples include AlexNet[1], ResNET[2] and VGG[3], which are used in competitions such as the Large Scale Visual Recognition Challenge (ILSVRC).

## Problem Statement

Whilst the scope of image classification is vast, I will be focusing on one particular use case and that is the classification of food. A significant proportion of images being uploaded are

[1] AlexNet winner of ILSVRC 2012, used 8 layers
[2] ResNET winner of ILSVRC 2015, used 152 layers
[3] VGG used 19 layers

food and in particular photos of people’s lunches. However, users are still having to add tags or labels manually to describe their dish. If this problem can be solved, users on sites such as Instagram would no longer have to hashtag their dish manually and business could also utilize this service to provide all sorts of applications for customers. For example, imagine an app that could tell you what recipes you can cook from a photo of your ingredients!

## Solution Statement

In this project I will create a CNN to train on a dataset of food images that will take new images as an input and classify them. The output of the application should return back a list of labels with their confidence levels. The application will train on a dataset of food images using cross validation to build the model, however, if the confidence level remains below the threshold, top-5 error rate 50%, I will consider exploring different existing CNNs and using transfer learning to improve the accuracy. Existing CNNs are trained across millions of images, using them as feature extractors could potentially greatly improve our performance given our current data sets are only in the tens of thousands.

Testing will be conducted on images not used in the validation or training of the model. 90% of the images will have food which matches to the labels I have provided at the start the training, 5% will contain food not strictly mapped to one of the labels and the final 5% will have no food in the image. In the case of classifying images with no food the model should understand that no food is present.

## Datasets and Inputs

The model is performing supervised learning to classify images into specific labels using a convolutional neural network. In order to train the model weights, we need to obtain a set of images and labels as the input. CNNs will require a suitable number of images to create enough filters to provide accurate results, given the problem at hand which is to classify foods into categories we will need thousands of images to reach our target of 50% accuracy. Datasets needs to be clean i.e. the training image needs to match up with the label and then able to be processed into the correct input size and shape for the CNN. This would include transforming the image to a square and potentially reduce the resolution to improve training times (but not too much that it’ll affect performance).

The dataset I have chosen is UECFOOD-256[4] 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. The data is distributed almost evenly across the datasets (with approximately 130-180 images per folder) with the exception of a three labels that have an abnormally larger dataset. The largest 3 labels have 729, 621 and 234 images. Our pre-processing step should include image re-sizing as the current data vary from 200 to 900 pixels for both the height and width. All images have colour layers which allow us to add extra channels to our CNN.

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

If the model becomes accurate, we could potentially add more categories and data to the training set to classify more dishes. The dataset contains exactly 31,651 images, which should hopefully be enough to train a suitable model and reach the performance target.

## Benchmark Model

The world's best CNN models produce incredible results in classification and the award winning ones outperform humans when classifying images. During the ImageNet Large Scale Visual Recognition Competition, GoogLeNet had a 6.7% top 5 error rate which is less than the average 10% error rate that humans have in classification. However, these CNNs are trained across millions of images and even with the latest specifications of GPUs, this will still take a week to get the model trained. With consideration to how existing CNNs have performed, our aspiration in this experiment, with 31, 651 images, is to reach a 40-50% top 5 error rate for our model. However, if results are unsatisfactory after iterating on the model, I will apply transfer learning into the solution and use one of the existing models to act as the feature extractor which will improve the overall performance.

Our initial base benchmark will be the top 5 errors from a basic CNN model of one layer hidden layer with 16 nodes running on the same training data. This will be the initial model that we will iterate over and hopefully as we add more layers, produce successful results.

## Evaluation Metrics

Top 1 and top 5 errors will be calculated as each version of the model is run against the test set. If transfer learning is used then the top 1 and top 5 results will also be recorded.

## Project Design

The project will involve pre-processing the data, training with cross validation on to the CNN and measuring its performance against the test data. The project will done in Python and version 2.7 is chosen to it's large support for machine learning libraries. The model will be constructed using Keras with a TensorFlow backend. Keras is chosen since it is simpler to setup a deep learning model than using TensorFlow directly and we avoid Theano due to its lack of support.

### Hardware Requirements

The code will run on a local desktop that own which is a 4th Generation i7 with an NVIDIA GTX 970 graphics card. If the hardware cannot compute the training model within an hour, I will consider running the code on an AWS AMI[5] with more processors and GPUs.

[5] https://aws.amazon.com/amazon-ai/amis/

## Pre-Processing

The aim of the pre-processing phase is ensure the data is in the correct state as an input for the training model.It is not mandatory for images to be of the same size for convolutional layers but having it the same shape reduces complexity of padding different shapes or cropping the images. In this project, the images will be resized to 224 x 224 which is commonly used in many CNNs.

After resizing the images we will begin the augmentation process. First feature scaling 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.

After feature scaling, a 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[6].

Once the data has been normalized, we can create more images with the existing data through rotations, transpositions and flipping the image randomly. This will not only provide more training data for the image but will reduce the chance of overfitting. All the augmentation steps outline so far can be done via the keras image library.

The current labels are currently written in a text file and this will be extracted to correlated with the folder names of the images. Once this is done we are setup for cross validation with the model. Here I will choose a 70:20:10 split, therefore 70% of the randomly selected images will be used for training, 20% for validation and 10% for testing. I wanted the training set to have a larger proportion of the dataset to reduce the chances of overfitting.

## CNN Model

The CNN model will consists of convolutional layers, pooling layers and a connecting layer at the end to produce the classification result. The number of each layer along with the total number of nodes will be experimented on in order to produce the best results in the testing phase. Dropouts will be used to ensure the weights are trained evenly throughout the CNN. The first convolutional layer will require the shape of the input and number of channels. Since I wish to include the colour information within this model I will specify three channels to represent the colours: red, blue and green.

Convolutional layers will use the ReLU activation function to avoid the vanishing gradient problem because it’s gradient is always 1 when positive. It has a performance boost[7] when handling negative values (negative values are represented as 0) which makes calculations

[6] Learning Multiple Layers of Features from Tiny Images has extensive research on the whitening process
[7] AlexNet uses ReLU which reaches the 25% training error rate ten times faster than tanh neurons

much simpler. The padding for the sliding window will be ‘same’ which automatically pads the window if it reaches the edge of the image rather than discarding the information, thus reduce data loss.

The number of layers to have will be interesting, AlexNet uses 8 layers and produced strong results during the ILSVRC 2012 competition and was ahead of any other CNN model. However the winner of 2015, ResNET, uses a total 152 layers, but stacking extra layers does not necessarily produce better results by default. Normalization is more important to avoid the vanish gradient problem and adding more layers can increase the training error which ResNET avoids with a deep residual learning framework. Due to the hardware and time constraints of this project, I suspect the number of layers will not exceed 30 as it’ll be too computationally expensive.

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.