Identification and Classification in Food Images

Tom Barrett

December 20, 2017

Abstract

Final Year Project focused on identification and classification of food images. A background of previous work done on the subject is outlined, followed by experiments in creating neural networks to perform classifications and segmentations. These experiments will be applied to food images and evaluated.

Contents

Chapter 1

Introduction

1.1 Overview

This project explores the use of indentification and classification of food images for use in a calorie measurement android application. Food calorie consumption is a huge problem in the modern world. Over 25% of the population in Ireland is obese and this figure is likely to rise over the coming years. A mobile application that could help keep track of a user's calorie intake by taking pictures of their meals would be a great help. The area of Machine Vision is a very difficult topic to address as it is a very hard task for computers to undertake. We, as humans, take vision for granted as we can soon see, from the study of Machine Vision, that there are many difficult steps that have to be made for full identification and classification of an image.

When looking into calorie measurement using an image, there are three questions that have to be answered:

- Where are the Regions of Interest (ROI) in this food image?
- What food types are in these ROI's?
- What is the portion size of each food type?

Figure 1.1: Segmented Image

In this project, my main focus will be on the first question, 'Where are the Regions of Interest (ROI) in this food image?'. This is normally achieved through image segmentation. Image segmentation is the process in which you divide an image into multiple segments as per Figure 1.1.

Many researchers in various machine vision labs have attempted to solve this problem using different methodologies. There has been promising results from some papers but these are mostly under highly constrained circumstances. When mixed foods are introduced to the problem, many of the methods fail. Convolution Neural Networks (CNN) have had very promising results in the field of image classification in the recent years but to get to the classification step, image segmentation is first needed, otherwise known as image identification. Where CNN's have been shown to work best is in the area of face detection.

I have researched many different methods of image segmentation but it seems that CNN's have had the best results for multiple objects in one image and I

hope to apply these results for many foods in an image. This is because we will rarely want to classify an image that only has one food item in it. Therefore, the one shot approach approach won't work by which we build a classifier that takes in an image and gives back the most likely food in that image.

The system that I am proposing to solve the problem statement would be able to integrate with an Android mobile phone application. The idea is, that when a user is about to eat their meal, they can simply take a picture of their meal for computation. From here, the application would take the image, find the objects (ROI) in the image and take note of them. Concurrently, the application would attempt to classify each object detected. Once this is done, the size of each food type would be measured and through this an overall calorie count would be displayed for the user. This could be logged for user metrics. I will focus on finding the objects for this application as I would not be able to implement the full system due to time constraints.

1.2 Objectives

Primary Objectives

Replicate results of previous work and possibly improve the result

There are many different approaches to food identification and classification, many of which we will see in Chapter 2. I hope to select an approach that has shown promising results in the past and replicate them. As a stretch goal, I may be able to improve the results that were originally found.

Develop an application

Another objective for this project is to develop an application that can be used for dietary assessment. This application would be able to take an image and then identify and classify the foods within the image. I will not focus on size estimation of the food type in this project.

Secondary Objectives

Understanding of Convolutional Neural Networks

In the project, I will be using Convolutional Neural Networks (CNN's) for object identification in Food Images. I will be using an API for this due to time constraints but it is a key objective for me to develop a deep understanding of CNN's as they are quite pivotal in the current Machine Vision Industry and I find bio-inspired systems very interesting.

Learn about different image identification and classification techniques

Although, I will be using CNN's for my implementation, I will not be turning a blind eye to other methods of identification and classification. I have done extensive research on many different methods prior to my decision to use CNN's. I think it is very important to learn about other methods as different methods are better suited for some situations and it would be best to know about these methods due to the inevitability of their use.

1.3 Methodology

The following methodology were adapter for this project:

Define the research question

The first step to this project was to define the research question. I knew that I wanted to look into the general area of 'Food Identification and Classification' but the scope of this is too broad for an FYP. Therefore I decided that the research question would be to look at the food identification aspect ie. region of interest detection.

Literature Review

Once I had defined the question, finding related work was the next milestone. There are many attempts at Food Image Classification and these were not difficult to find, using Google Scholar, but many of these papers glossed over the segmentation aspect and relied on third parties for this step. Because of this, I had to follow quite a few references to different papers that focused solely on image segmentation.

Explore different image identification methods

I had collected various image identification from the literature review that I carried out so there were many options to evaluate. I wanted to try something with Convolutional Neural Networks so more traditional methods of identification using colour and texture was not so strenuously explored.

Select an image identification method

Research technologies and develop skills in these technologies

As I used a Convolutional Neural Network (CNN) in this project, I leveraged many resources to enrich my understanding of the process. I decide to use Tensorflow as my main resource in creating a CNN so I followed online tutorials for this technology. I also enrolled in a Deep Learning Course on Udacity to enhance my understanding and skills.

Build a prototype of the application

Compare and analyse results to other implementations

1.4 Overview of Report

This report is broken down into various main headings:

Introduction

This section is to give on overview of what this project is about, how I will approach this project and why I am doing it.

Background

I will be giving some information on the background of the subject that I am focusing on.

Experiments

In this part of the report I will carry out various experiments using tensorflow.

Empirical Studies with Food Images

This section will analyse the results I have obtained and compare them against various metrics and other implementations.

Discussion and Conclusion

In this section, I will discuss my results from the empirical studies and conclude my findings.

1.5 Motivation

I find the topic of Computer Vision a very interesting one. It excites me, to be able to 'teach' a machine how to see as we do. For this reason, I really wanted to learn about Neural Networks and this was a large motivator for this project.

Once I had a topic that I wanted to research, I needed a focus or problem statement for this research. I find that it is much more rewarding to work on something that positively impacts both myself and other people so I decided that I wanted to research something that fit this requirement.

Food calorie consumption is a very big problem in the modern world. Over 25 percent of the population in Ireland are obese. A mobile application that could help keep track of a user's calorie intake by taking a picture of their meals would be a big help to combat this problem. This problem statement works very well for me because of it's application use and because of its complexity. Identifying and recognising food is much more difficult than say recognising faces as it has no uniform shape. Therefore, this problem would also be very beneficial to developing skills in the computer vision area.

I would like to develop these real world skills so that I can partake in Computer Vision projects in industry or to do further research in academia. This is because machine learning has really taken off in the last few years and is used by many in industry. While machine learning as a whole has become very popular, computer vision is probably the most prominent that has come out of it. Face detection, for one, is being researched extensively for use in personalised advertising and also secure access to devices and systems.

Chapter 2

Background

2.1 Introduction to Machine Learning

In **MLANN**, Machine Learning is defined as "the question of how to construct computer programs that automatically improve with experience". Machine Learning has blossomed in recent years with applications across multiple domains using vastly different paradigms and technologies.

There are many ways in which Machine Learning can be used in the modern world, many of which are being utilised to great affect. Some of these applications, are image recognition, natural language processing, medical diagnosis and many more. There may be fear that Machine Learning will start to take away many jobs from humans but this may not be the case. Imagine a doctor, having to diagnose a patient, a machine can offer suggestions based on very large datasets of what the diagnosis is. This is not to say that a Machine would be perscribing patients, but merely act as an assistant to the doctor.

Machine Ethics is a large problem that comes hand in hand with Machine Learning. There is a very important question of who takes the blame when things go wrong, that is why I think it is important that we only use Machine Learning to advise and not to determine but this can be very difficult in a world where, for example, an autonomous car has to decide between crashing into a vehicle beside them with an unknown number of people inside or the

two children playing in the street.

One of the most exciting avenues in Machine Learning, in my opinion, is Computer Vision. Computer Vision can be used in many areas to improve our lives. As mentioned earlier, autonomous cars are only possible when a machine can determine what objects are around it. Computer Vision can allow a machine to recognise skin diseases in an image. The applications are nearly limitless and that is without taking into account other uses.

2.2 Neural Computing

The main area of my focus for this project is in Artificial Neural Networks (ANN). This is because I have researched extensively into ConvolutionalNeural Networks which are based on ANN's.

Artificial Neural Networks

An Artificial Neural Network is a bio-inspired system that is used to model the human brain in how it learns from experience. The ANN uses this model to build a very complex web of connected units called artificial neurons. These nuerons are connected by certain weights which determines the processing capacity of the network and these weights are created by learning a dataset.(Malachy) An ANN has a set of inputs that take in a value, sometimes from network outputs and produce a single result or classification. While an ANN is bio-inspired from the human brain, there are many elements of the brain that are not present in ANN and many new elements in ANN that are not modelled from the human brain.

Before I can talk about Convolution Neural Networks which are vital the image processing, I will have to talk about the perceptron learning algorithm, the multi layer percepton, and backpropogation.

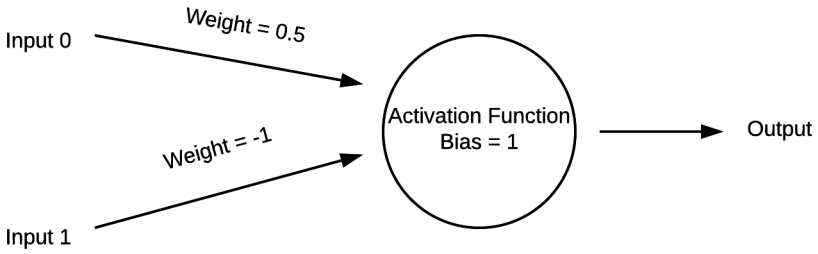


Figure 2.1: Perceptron

2.3 Perceptron Learning - Artificial Neuron

In our Artificial Neural Network a Perecptron is an Artificial Neuron. It is called an Artifical Neuron because it is a bio-inspired neuron which models a neuron in the human brain in terms of inputs and output.

In Perceptron learning, we can take two inputs which are put towards an activation function with a bias attached as seen in 2.1. These inputs are multiplied by the weights that connect the input to the activation function and depending on the result, the activation function may fire an output. These inputs are either 1 or -1.

This Perceptron Learning Rule assumes that there are two sets of instances, a positive and negative set, and that they are linearly seperable.

A perceptron is trained using supervised learning. When the perceptron classifies a results, it is told if it is correct or not. If the result is incorrect, weights are changed in value so that this error can be reduced **AI**.

The one major problem with perceptron learning and that is that it can't solve the problem if there is not a clear linear separation between the classes. There is a way in which we can attempt to solve this, through the delta rule. The delta rule utilizes gradient descent to find the best weight for the training samples **MLANN**. We will discuss gradient descent in the next section.

2.4 Multi Layered Perceptron

Multi Layer Perceptrons (MLP) are made up of multiple layers of perceprons connected together. Firstly, we have an input layer, followed by one of more hidden layers and then finally an output layer. Any Neural Network with more than three hidden layers is categorised as a deep layer.

The input layer of your network consists of the data you feed into the network in order to classify it. The input layer passes this data to a hidden layer whose purpose is to transform this data into something that the output layer can understand. The output layer normally consists of a class prediction.

Multi Layer Perceptrons are a class of feed forward Artificial Neural Networks. These means that the output of each perceptron feeds into an input in the next layer of the network.

There is one large problem with MLP's and this is why Convolutional Neural Networks (CNN) were created. If you attempting to classify images with an MLP then each pixel in that image would have to be a separate input. This created a massive amount of neurons through all the layers and this isn't feasible. CNN's solve this problem which we will discuss later.

Gradient Descent

Gradient Descent is an algorithm used to find the optimal weights to produce the smallest prediction error. It is used to overcome problems of non linearly separable classes. Gradient descent search selects a random weight value and then modifies it gradually to minimize the error. "At each step, the weight vector is altered in the direction that produces the steepest descent along the surface" **MLANN**. This step is iterated until the lowest value is met.

One problem with Gradient Descent is that if we look at 2.3, we may never get to the optimal point, point B. This is because we will find point A without too many problems but when the weights change we will get too high a slope of error and therefore will never reach point B.

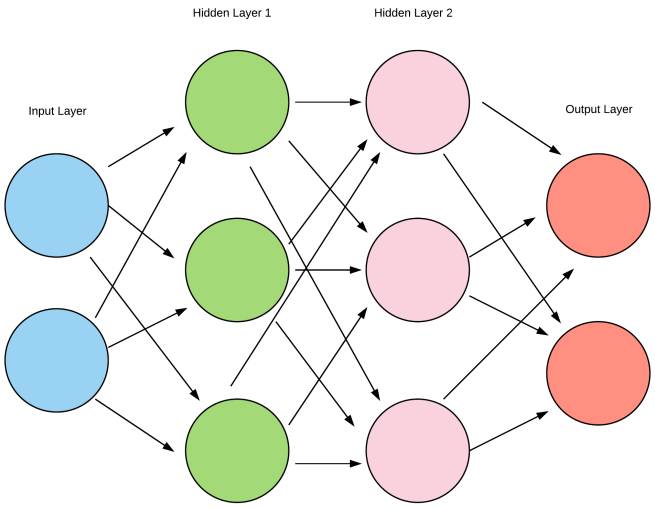


Figure 2.2: Multi Layer Perceptron

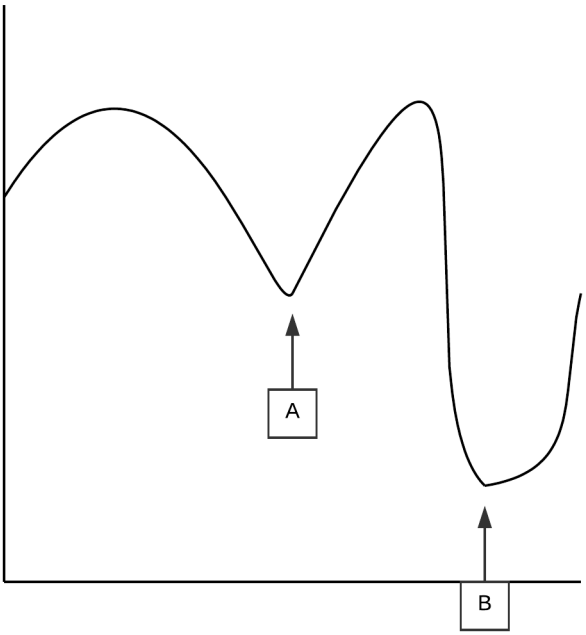


Figure 2.3: Gradient Descent

Another variation of Gradient Descent is Stochastic Gradient Descent (SGD). SGD is different because it updates "weights incrementally, following the calculation of the error of each individual example" **MLANN**.

Backpropagation

"The Backpropagation algorithm learns the weights of a multilayer network, given a network with a fixed set of units and interconnections" **MLANN**. Backpropgation attempts to minimise the mean squared error between the target output and the output of a network.

Backpropagation works by starting at the output layer of the network and going back through previous hidden layers, updating weights as it goes.

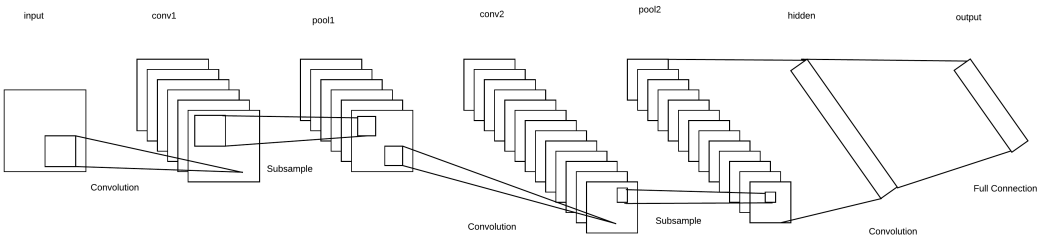


Figure 2.4: CNN Architecture

2.5 Support Vector Machine

A Support Vector Machine (SVM) is a machine learning algorithm that has been very popular before the power of a CNN was realised.

A SVM works by creating an n-dimensional space, with n as the number of inputs you have **svm**. The SVM algorithm finds the hyperplane that splits this space. This hyperplane can then be used for classification.

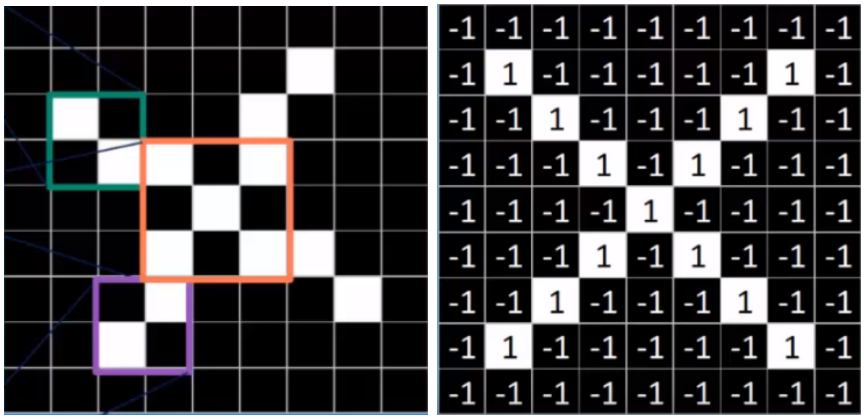
2.6 Convolutional Neural Networks Overview

Convolutional Neural Networks (CNN's) are essentially a Multi Layered Percetron with a special structure. CNN's have one major difference from a MLP, they have extra layer of convolution and pooling. The architecture of a convolution network can be seen in Figure 2.4.

Figure 2.5a show an image that we want to compare against Figure 2.5b. For humans, it is quite easy to determine that these images are very similar but for a computer this task is surprisingly difficult.

So what a CNN does, to combat this problem, is to take a small feature from Figure 2.5a and compare it to a subsection of Figure 2.5b. The CNN multiplies the feature and a section of Figure 2.5b, adds up the results and divides by 9. This then gives a decimal value of how likely it is that the feature is in the part of the image, as seen in Figure 2.6b. This is called filtering. The

Figure 2.5: Image filtering



(a) Image to Classify (b) Image to Compare

Convolutional layer is composed of carrying out this filtering for every single possible location in Figure 2.5b.

Next is the Pooling Layer, what this layer does, is it takes the convoluted layer output, you can use Figure 2.6b as reference, and from a user defined size ie. 2x2, gets either the highest decimal value (Max pooling) or the average value (Mean pooling) and records that as the new value for the section. This is then applied to the entire image. As we can see in Figure 2.7 we know have a much smaller image stack in which to classify, thus making the computation easier.

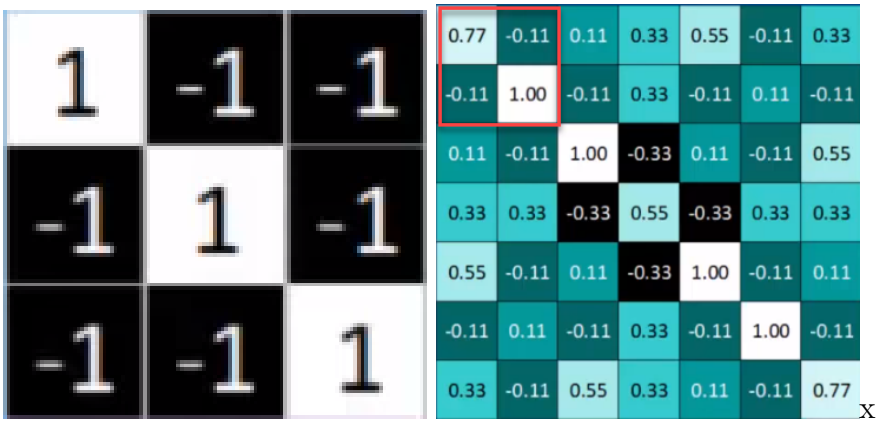
In between the Convolution and Pooling layer, there is sometimes a Normalization layer. This Normalization layer creates Rectified Linear Units (RLU's). In other words, if we take Figure 2.6b, it changes all minus values to zero.

There are some problems with CNN's however. One of the main problems is that you need a very large dataset in order to produce am accurate model and the training can be very time consuming without a GPU.

Fully Convolutional Networks

A Fully Convolutional Network is one that does not have a fully connected layer and in a fully connected layers place is another convolution layer.

Figure 2.6: Image Convolution



(a) Image Feature to Search (b) Convoluted Image

Figure 2.7: Pooled Image

2.7 Convolutional Neural Networks for use in Classification

Many reserachers have used convolutional neural networks for image classification with various network architectures and many have used a food image dataset. I will be looking at some of these papers below.

Deep Learning Based Food Recognition

One paper focused on a deep learning approach to food image recognition based their neural network architecture on Inception-ResNet and Inception V3. They also used the Food-101 dataset **deepLearning**. For this system, Googles Tensorflow was used for image preprocessing. Preprocessing was needed as the environmental background is different in many food images. Because of these "Grey World method and Histogram equalization" **deepLearning** were used.

AWS GPU instances were used for training and the results on completion were quite impressive with a Top-1 Accuracy of 72.55% and a Top-5 Accuracy of 91.31% **deepLearning**

Food Image Recognition Using Deep Convolutional Neural Network With Pre-Training and Fine-Tuning

Another research team in Japan researched this topic. They were aware of how difficult the problem was and therefore employed many techniques to solve the problem such as "pre-training with the large-scale ImageNet data, fine-tuning and activateion features extracted from the pre-trained DCNN" **yanaiFood**.

In conclusion, they found that the "fine-tuned DCNN which was pre-trained with 2000 categories" **yanaiFood** from ImageNet was the best method. A DCNN is a Deep Convolution Neural Network. The achieved results of 78.77% for Top-1 Accuracy in the UECFOOD100 dataset.

Food Detection and Recognition Using Convolutional Neural Network

kagayaFood also employed the use of convolutional neural networks for image detection. They used a CNN for the "tasks of food detection and recognition through parameter optimization" **kagayaFood**.

They found that a CNN is much better suited to the task than a Support Vector Machine (SVM). They achieved an overall classification accuracy of 93.8% against their baseline accuracy of 89.7% **kagayaFood**. This accuracy was calculated using a dataset that they created specifically for this task. When they had completed the task they analysed the trained convolutional kernels and came to an interesting conclusion. They found that "color features are essential to food image recognition" **kagayaFood**.

DeepFood: Deep Learning-based Food Image Recognition for Computer-aided Dietary Assessment

The last paper that I will look at, oriented around using a convolutional neural network for food image recognition, focused on developing a dietary assessment application for use on a smartphone. They used the UEC-256 and Food-101 datset for their experiments and achieved impressive results.

They used a convolutional neural network but "with a few major optimizations, such as optimized model and an optimized convolution technique" **deepFood**. They used the Inception module for their CNN. After the inception module was complete, they made the GoogleNet by combining modules. In total, the network had 22 layers.

They achieved the results shown in 2.1.

Table 2.1: DeepFood Results

	Top-1	Top-5
UEC-256	54.7%	81.5%
UEC-100	76.3%	94.6%
Food-101	77.4%	93.7%
UEC-256 With Bounding Box	63.8%	87.2%
UEC-100 With Bounding Box	77.2%	94.8%

2.8 Convolutional Neural Networks - Extensions

Region Based Convolutional Neural Networks

Ross Girshik and other contributers had some very positive results in the area of object detection using region based convolutional neural networks. There were four iterations of papers based on this work by Ross and groups in UC Berkley, Mircosoft and Facebook. A PHD student at the time of Ross's first paper also completed his dissertation on the subject. I will analyse this papers, their results (2.2) and the changes made through each iteration.

RCNN

In the first paper written by Ross Girshik, while researching at UC Berkeley, focused on two main insights. These were that "one can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects" and that "when training data is scarce, supervised pre-training for n auxilary task, followed by domain-specific fine-tuning, yields a significant performance boost" **rcnn**.

The system that they developed followed these steps:

- Take image as input
- Extract approximately 2000 region proposals from the image

- Compute fixed length vectors of features for the regions using a convolutional neural network
- Use a Support Vector Machine (SVM) to classify these regions
- Bounding box regression for final region proposals

This system utilised selective search to gather these region proposals but they mention that a sliding-window detector is also an option. Ross Girshik and his team used the open source Caffe CNN library for this system. The system is quite efficient and scalable. It is scalable because of the fixed length vector of features which will remain constant regardless of inputs and additional outputs.

The team evaluated their results on a few metrics and test sets as seen in 2.2.

Fast RCNN

Ross Girshik's next iteration of work on region based convolution neural networks took place in Microsoft Research. This paper was titled "Fast R-CNN" as it's aim was to decrease training and testing time "while also increasing detection accuracy" **fastRcnn**.

This paper analyses why RCNN **rcnn** was slow and therefore how it could be improved. RCNN was classified to be slow because of three main factors:

- There are multiple stages to training as both a CNN and a SVM need to be trained.
- In training of the SVM, each region proposal must be written to disk and is therefore expensive.
- Object detection takes 47s per image **fastRcnn**.

Due to these problems with RCNN, a new algorithm, titled Fast RCNN was proposed. The architecture is as follows. An image is taken as input along with a proposals for regions. The image is pushed through convolutional and

pooling layers (using max pooling). A fixed-length vector of features is then extracted from each region proposal. These vectors are inputted to fully connected layers for bounding box location prediction **fastRcnn**.

At detection time, a pass through of the net is all that is needed so this runtime is significantly less than RCNN.

Faster RCNN

Due to the success of RCNN and Fast RCNN, Faster RCNN was introduced to combat the problem of region proposal computation **fasterRcnn**.

The architecture for this system comprises of two modules. These consist of a convolutional neural network for region proposals (RPN) which the feeds into a Fast RCNN detector. These combine to produce a single neural network for object detection.

Instead of training these networks seperately, the team had to look at how to share layers between the two networks. There were three option available:

- Alternating training whereby RPN is trained, and then used to train Fast RCNN. The Fast RCNN network is then used to initialise RPN and the process is iterated **fasterRcnn**. This paper follows this approach.
- Approximate joint training.
- Non- approximate joint training.

Mask RCNN

The most recent paper on this topic was also written by Ross Girshik while working with Facebook AI Research **maskRcnn**. Mask RCNN "extends Faster RCNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box regression" **maskRcnn**.

Mask RCNN has two modules, similar to Faster RCNN, where the first module is the Region Proposal Network. In the second module, in parallel to classifi-

Table 2.2: Results from Region Based CNN Research

	VOC07	VOC10	VOC11	VOC12	COCO15	COCO16
RCNN	58.5%	53.7%	47.9%	N/A	N/A	N/A
Fast RCNN	70.0%	68.8%	N/A	68.4%	N/A	N/A
Faster RCNN	78.8%	N/A	N/A	75.9%	42.7%	N/A
Mask RCNN	N/A	N/A	N/A	N/A	N/A	63.1%

cation, a binary mask is outputted for each region. Bounding box regression and classification are done in parallel.

2.9 Image Segmentation

Fully Convolutional Neural Networks for Semantic Segmentation

There has been a very interesting paper from UC Berkeley focused on using Full Convolutional Networks for semantic segmentation **fcn**. Fully Convolutional Networks (FCN) do not have any fully connected layers. They are replaced with more filtering layers. Nvidia Digits have a semantic segmentation implementation based off the work of this paper.

They took this approach because "feedforward computation and backpropogation are much more efficient when computer layer-by-layer over an entire image instead of independently patch-by-patch" **fcn**. This was also because they were focused on object detection. Normal classifiers do not work very well when they are to classify more than one subject in an image and image segmentation was a way to solve this.

There are a set of steps you can follow to turn a CNN into a FCN for semantic segmentation as follows ie. change to a convolutional layer from a fully connected one:

- The size of the filters must be set to the size of the input layers.

Table 2.3: FCN Resulys **fcn**

	FCN
VOC11 mean IU	62.7
VOC12 mean IU	62.2
PASCAL VOC10 pixel acc.	67.0
PASCAL VOC10 mean acc.	50.7
PASCAL VOC10 mean IU	37.8
PASCAL VOC10 f.w. IU	52.5

- For every neuron in the fully connected layer, have a filter.

Segmentation

Graph Based Segmentation

Graph cut segmentation has been used extensively in image segmentation. OpenCV has an implementation of a graph cut algorithm called grabcut which has been used to segment food on occasion **graphCut**.

According to **graphCut**, "Graph cut based method is well-known to be efficient, robust, and capable of finding the best contour of objects in n image, suggesting it to be a good method for separating food portions in a food image for calorie measurement". Along with the graph cut segmentation algorithm, this research team also used color and texture segmentation. Gabor filters were used to measure texture features **graphCut**. When color and texture segmentation was applied, the method came into difficulty with mixed foods but by applying graph cut segmentation, clearer object boundaries were shown.

In conclusion, the accuracy of the classification increased when using graph based segmentation rather than color and texture as seen in 2.4.

Table 2.4: Results

	Single Food Portion	Non-mixed Food	Mixed Food
Color and Texture	92.21	N/A	N/A
Graph Based	95 (3% increase)	5% increase	15% increase

Local Variation Framework

Another paper was published in which the research team attempted to create a food calorie extimation system **foodImageAnalysis**. This system would compromise of three steps, image segmentation, image classification and weight estimation. For the segmentation module, a local variation approach to segmentation was performed. Local variation is by which intensity differences between neighbouring pixels is measured. This is a type of graph based segmentation.

The team also carried out some segmentation refinement when the segmentation algorithm had been performed. This consisted of removed small segments (defined as less than 50 pixels) and trying to prevent over and under segmentation. After classification was performed on each segment, segments with low confidence values were removed **foodImageAnalysis**.

Conclusion

Both of the above papers of **graphCut** and **foodImageAnalysis** used a graph based segmentation. The first paper used a more generic implementation while the second used a local variation framework. Both methods provided successful results in the image segmentation process.

2.10 Technologies

Tensorflow

Tensorflow is a deep learning api for various machine learning paradigms. I will be using tensorflow to create neural networks. Tensorflow has two utilisations, through a Graphics Processing Unit (GPU) and also through a Central Processing Unit (CPU). GPU computation is recommended for CNN training.

Central Processing Unit Computation

It is quite easy to get tensorflow up and running if you are only using a CPU to train. I have successfully installed tensorflow cpu on both Windows and Ubuntu. For Windows you can download and install using the tensorflow website and on ubuntu you can using apt-get. Once installed, tensorflow can be imported into any python shell or script for use. Tensorflow can also be used in C. There will be various python implementations of neural networks in Chapter 3.

Graphics Processing Unit Computation

For use with a GPU, the set up for tensorflow is a bit more complicated. Firtsly you use check that the GPU in your machine is compatible for CUDA 8.0 using the NVIDIA website. If your GPU is compatible, you must install CUDA after signing up as an NVIDIA developer. CUDA 8.0 is compatible with tensorflow. You alos need to install cudnn6. The NVIDIA website contains tutorials to install these. Once these are installed, download and install tensorflow-gpu. This can imported into python similar to CPU computation.

2.11 Evaluating the Output

There are various different metrics that can be used for evaluating the output of a classifier or segmentation algorithm, many of which we have seen in previous

sections. I was analyse a few of these evaluations of results so that I can use them as a reference to apply to my own experiments. I will also look into some problems associated with evaluating models.

Research into Diagnosing Errors in Object Detectors

There has been some research into the question of how to evaluate object detectors, one of which I will discuss in detail **diagnosingErrors**. This paper in question "analyzes the influences of object characteristics on detection performance and the frequency and impact of different types of false positives" **diagnosingErrors**. They found that there were many effects that had influence on detectors as follows:

- occlusion
- size
- aspect ratio
- visibility of parts
- viewpoint
- localization error
- confusion with semantically similar objects
- confusion with other labeled objects
- confusion with background

The research team goes on to analyse false positives in object detectors. Localization errors were a large factor. This is were bounding boxes overlap to other objects in the image. Confusion with similar objects had a large influence on false positives also by which, for example, a dog detector had a high score for a cat **diagnosingErrors**. Confusion with dissimilar objects and confusion with background are the categories of the rest of the false positives they measured.

In conclusion the team would that "Most false positives are due to misaligned detection windows or confusion with similar objects" **diagnosingErrors**. They had some recommendations towards improves detectors as follows:

- Smaller objects are less likely to be detected
- Localization could be improved
- Reduce confusion with similar categories
- Robustness to object variation
- More detailed analysis

Detection Average Precision

The average detection accuracy of a system. See 1.

Mean Average Precision

The mean accuracy of a system across all results. See 2.

Distribution of top-ranked false positives

This metric is used to tell where most errors occur when false positives are evident. These errors are broken down into four categories of localization, similar objects, background confusion and others. See 3.

Segmentation Mean Accuracy

This is the mean accuracy of segmentation. See 4.

Per-category segmentation accuracy

This metric measures the accuracy of segmentation at a category level. See 5.

Per-class segmentation accuracy

The accuracy of segmentation at a class level is analysed. See 6.

Top-1 and Top-5 Accuracy

When a classifier is given an image it normally responds with a list of predictions along with a decimal representation of the likelihood that it is of that class. The Top-1 accuracy is the top valued prediction and Top-5 accuracy is the top 5 predictions.

Chapter 3

Experiments

3.1 Experiment 1

Network Architecture

Dataset

API's

Script

Udemy CNN

December 17, 2017

1 MNIST CNN

In [1]:
```
import tensorflow as tf
import input_data
```

In [2]:
```
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```

Helper Functions

In [3]:
```
#INIT WEIGHTS
def init_weights(shape):
    init_random_dist = tf.truncated_normal(shape, stddev = 0.1)
    return tf.Variable(init_random_dist)
```

In [4]:
```
#INIT BIAS
def init_bias(shape):
    init_bias_vals = tf.constant(0.1, shape = shape)
    return tf.Variable(init_bias_vals)
```

In [5]:
```
#CONV2D
def conv2d(x, W):
    #x -> [batch, H, W, Channels]
    #W -> [filterH, filterW, ChannelsIn, ChannelsOut]
    return tf.nn.conv2d(x, W, strides = [1,1,1,1], padding = 'SAME')
```

In [6]:
```
#POOLING
def max_pool_2by2(x):
    #x -> [batch, H, W, Channels]
    return tf.nn.max_pool(x, ksize = [1,2,2,1], strides = [1,2,2,1], padding = 'SAME')
```

In [7]:
```
#CONVOLUTIONAL LAYER
def convolutional_layer(input_x, shape):
    W = init_weights(shape)
    b = init_bias([shape[3]])
    return tf.nn.relu(conv2d(input_x, W) + b)
```

```
In [8]: #NORMAL (FULLY CONNECTED)
        def normal_full_layer(input_layer, size):
            input_size = int(input_layer.get_shape()[1])
            W = init_weights([input_size, size])
            b = init_bias([size])
            return tf.matmul(input_layer, W) + b
```

Placeholders

```
In [9]: x = tf.placeholder(tf.float32, shape = [None, 784])

In [10]: y_true = tf.placeholder(tf.float32, shape = [None, 10])
```

Create Layers

```
In [11]: x_image = tf.reshape(x, [-1,28,28,1])

In [12]: #32 features for every 5 x 5 patch with 1(grayscale)
         convo_1 = convolutional_layer(x_image, shape = [5,5,1,32])
         convo_1_pooling = max_pool_2by2(convo_1)

In [13]: convo_2 = convolutional_layer(convo_1_pooling, shape = [5,5,32,64])
         convo_2_pooling = max_pool_2by2(convo_2)

In [14]: convo_2_flat = tf.reshape(convo_2_pooling, [-1,7*7*64])
         full_layer_one = tf.nn.relu(normal_full_layer(convo_2_flat, 1024))

In [15]: #DROPOUT
         hold_prob = tf.placeholder(tf.float32)
         full_one_dropout = tf.nn.dropout(full_layer_one, keep_prob= hold_prob)

In [16]: y_pred = normal_full_layer(full_one_dropout, 10)

In [17]: #LOSS FUNCTION
         cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels = y_true,

In [18]: #OPTIMIZER
         optimizer = tf.train.AdamOptimizer(learning_rate = 0.001)
         train = optimizer.minimize(cross_entropy)

In [19]: init = tf.global_variables_initializer()

In [20]: steps = 1000

         with tf.Session() as sess:
             sess.run(init)

             for i in range(steps):
                 batch_x, batch_y = mnist.train.next_batch(50)
                 sess.run(train, feed_dict = {x:batch_x, y_true:batch_y, hold_prob:0.5})
```

```
                if i%100 == 0:
                    print("ON STEP: {}".format(i))
                    print("ACCURACY: ")
                    match = tf.equal(tf.argmax(y_pred, 1), tf.argmax(y_true, 1))
                    acc = tf.reduce_mean(tf.cast(match, tf.float32))
                    print(sess.run(acc, feed_dict = {x:mnist.test.images, y_true:mnist.test.lab
                    print('\n')
```

```
ON STEP: 0
ACCURACY:
0.101


ON STEP: 100
ACCURACY:
0.94


ON STEP: 200
ACCURACY:
0.9603


ON STEP: 300
ACCURACY:
0.971


ON STEP: 400
ACCURACY:
0.9717


ON STEP: 500
ACCURACY:
0.9759


ON STEP: 600
ACCURACY:
0.9772


ON STEP: 700
ACCURACY:
0.9811
```

```
ON STEP: 800
ACCURACY:
0.9793


ON STEP: 900
ACCURACY:
0.973
```

Results

Analysis

Chapter 4

Empirical Studies with Food Images

Chapter 5

Discussion and Conclusion

Appendix A

Appendix

VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 (Girshick et al.)	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA (Uijlings et al. 2013)	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets (Wang et al. 2013a)	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM (Fidler et al. 2013)	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN BB	**71.8**	**65.8**	**53.0**	**36.8**	**35.9**	**59.7**	**60.0**	**69.9**	**27.9**	**50.6**	**41.4**	**70.0**	**62.0**	**69.0**	**58.1**	**29.5**	**59.4**	**39.3**	**61.2**	**52.4**	**53.7**

Figure A.1: Detection Average Precision **donahue**

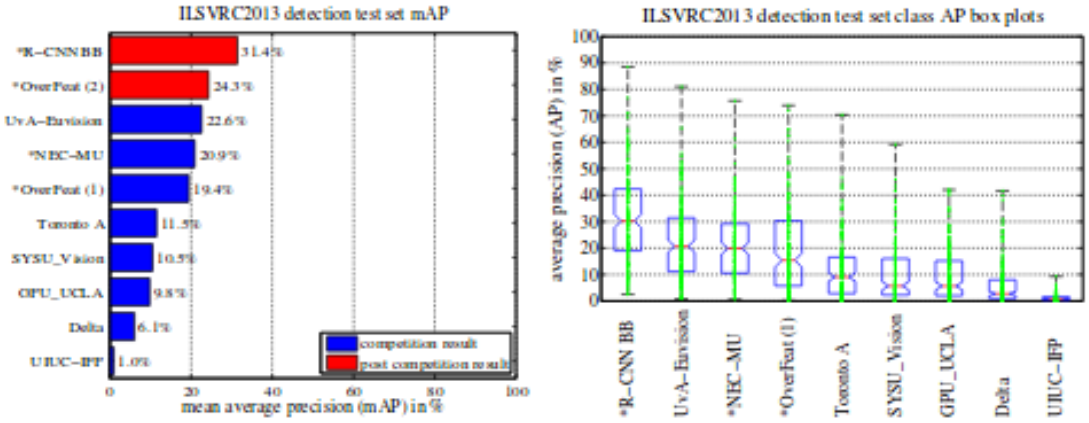


Figure A.2: Mean Average Precision **donahue**

Table A.1: My caption

Task/Deliverable	Deadline
Interim Report	21/12/17
Select method and paper to replicate	10/01/18
Iteration 1	???
Iteration 2	???
Iteration 3	???
Draft Final Report	08/03/18
FYP Product	10/04/18
FYP Report	17/04/18

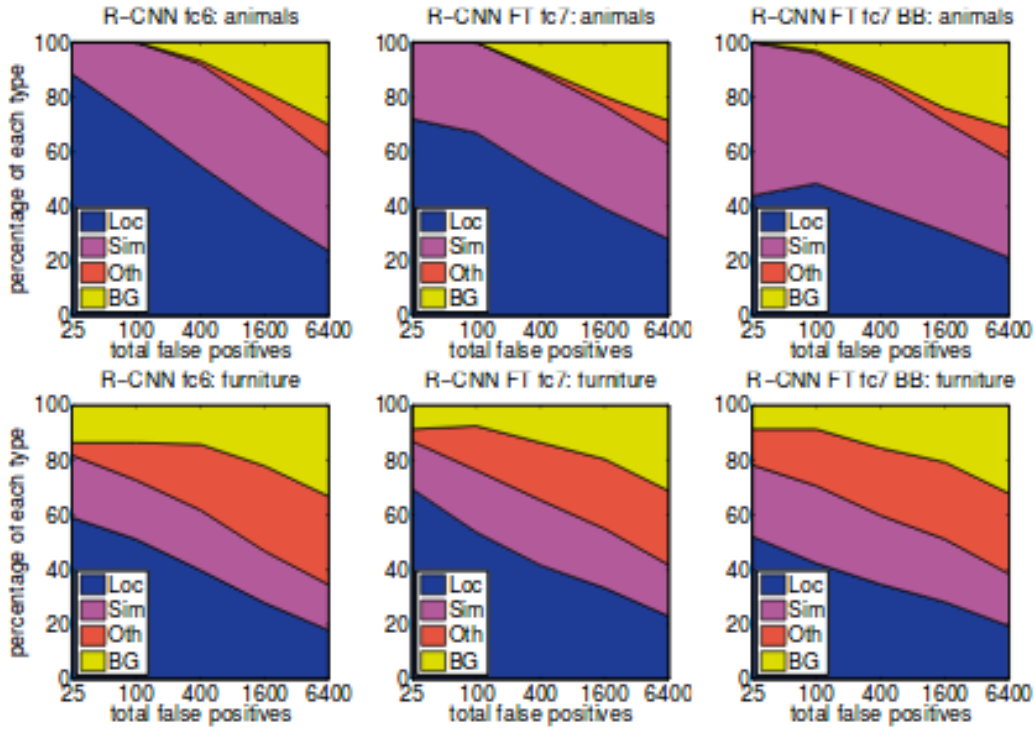


Figure A.3: Distribution of top-ranked false positives **donahue**

	full R-CNN		*fg* R-CNN		*full+fg* R-CNN	
O_2P (Carreira et al., 2012)	fc_6	fc_7	fc_6	fc_7	fc_6	fc_7
46.4	43.0	42.5	43.7	42.1	**47.9**	45.8

Figure A.4: Segmentation Mean Accuracy **donahue**

VOC 2011 val	bg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
O_2P (Carreira et al., 2012)	**84.0**	**69.0**	21.7	47.7	42.2	42.4	**64.7**	**65.8**	57.4	**12.9**	37.4	20.5	43.7	35.7	52.7	51.0	**35.8**	**51.0**	28.4	59.8	49.7	46.4
full R-CNN fc_6	81.3	59.2	23.9	42.9	40.7	38.8	59.2	56.5	53.2	11.4	34.6	16.7	48.1	37.0	51.4	46.0	31.5	44.0	24.3	53.7	51.1	43.0
full R-CNN fc_7	81.0	52.8	**25.1**	43.8	40.5	42.7	55.4	57.7	51.3	8.7	32.5	11.5	48.1	37.0	50.5	46.4	30.2	42.1	21.2	57.7	**56.0**	42.5
fg R-CNN fc_6	81.4	54.1	21.1	40.6	38.7	**53.6**	59.9	57.2	52.5	9.1	36.5	**23.6**	46.4	38.1	53.2	51.3	32.2	38.7	**29.0**	53.0	47.5	43.7
fg R-CNN fc_7	80.9	50.1	20.0	40.2	34.1	40.9	59.7	59.8	52.7	7.3	32.1	14.3	48.8	42.9	54.0	48.6	28.9	42.6	24.9	52.2	48.8	42.1
full+fg R-CNN fc_6	81.1	60.4	23.2	48.4	**47.3**	52.6	61.6	60.6	**59.1**	10.8	**45.8**	20.9	**57.7**	43.3	**57.4**	**52.9**	34.7	48.7	28.1	60.0	48.6	**47.9**
full+fg R-CNN fc_7	82.3	56.7	20.6	**49.9**	44.2	43.6	59.3	61.3	57.8	7.7	38.4	15.1	53.4	**43.7**	50.8	52.0	34.1	47.8	24.7	**60.1**	55.2	45.7

Figure A.5: Per-category segmentation accuracy **donahue**

class	AP	class	AP	class	AP	class	AP	class	AP
accordion	50.8	centipede	30.4	hair spray	13.8	pencil box	11.4	snowplow	69.2
airplane	50.0	chain saw	14.1	hamburger	34.2	pencil sharpener	9.0	soap dispenser	16.8
ant	31.8	chair	19.5	hammer	9.9	perfume	32.8	soccer ball	43.7
antelope	53.8	chime	24.6	hamster	46.0	person	41.7	sofa	16.3
apple	30.9	cocktail shaker	46.2	harmonica	12.6	piano	20.5	spatula	6.8
armadillo	54.0	coffee maker	21.5	harp	50.4	pineapple	22.6	squirrel	31.3
artichoke	45.0	computer keyboard	39.6	hat with a wide brim	40.5	ping-pong ball	21.0	starfish	45.1
axe	11.8	computer mouse	21.2	head cabbage	17.4	pitcher	19.2	stethoscope	18.3
baby bed	42.0	corkscrew	24.2	helmet	33.4	pizza	43.7	stove	8.1
backpack	2.8	cream	29.9	hippopotamus	38.0	plastic bag	6.4	strainer	9.9
bagel	37.5	croquet ball	30.0	horizontal bar	7.0	plate rack	15.2	strawberry	26.8
balance beam	32.6	crutch	23.7	horse	41.7	pomegranate	32.0	stretcher	13.2
banana	21.9	cucumber	22.8	hotdog	28.7	popsicle	21.2	sunglasses	18.8

Figure A.6: Per-class segmentation accuracy **donahue**