

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Janaki Prasad Koirala

Food Object Recognition: An Application of Deep Learning

Master's Thesis
Espoo, June 04, 2018

Supervisor: Professor Stephan Sigg, Aalto University
Advisor: Todor Ginchev M.Sc. (Tech.)

Author:	Janaki Prasad Koirala		
Title:	Food Object Recognition: An Application of Deep Learning		
Date:	June 04, 2018	Pages:	64
Major:	Machine Learning and Data Mining	Code:	SCI3044
Supervisor:	Professor Stephan Sigg		
Advisor:	Todor Ginchev M.Sc. (Tech.)		
<p>Identifying a food from its image can save people's life. It can be used to know the presence of potential allergens in food or by estimating the nutritional content of food, it may also be used to combat the obesity epidemic. With such applications in mind, we seek to exploit the advances in machine learning and deep learning to train models that identify European food from digital photos. From the literature it was discovered that the Faster RCNN was the current state-of-art CNN based framework which could get local information of object in image and recognize it. Furthermore, we also develop an Android application for recognition of food objects.</p> <p>Faster RCNN requires a large volume of data with labels and localization information of the objects present in them. It is very challenging to find such datasets to train our network. We made up a food dataset of 69k images with 445 labels and trained our model using those images. But the dataset was skewed in terms of numbers of images per category that negatively affected the performance of the model. To improve the performance, we tried several approaches like taking only a subset of labels and equalizing the number of training samples for each label. We also used transfer learning to get around the problem of overfitting the network when our training sample size is limited. Finally, by using publicly available data set and adapting it to our needs, our model was able to identify images with 0.37 mean Average Precision. The Android application uses this model to recognize food objects from images.</p>			
Keywords:	Food objects, Computer Vision, Deep Learning, Faster RCNN		
Language:	English		

Acknowledgements

I would like to express my gratitude to Prof. Stephan Sigg for his supervision. I appreciate his encouragement, support and feedbacks. I would also like to thank my colleagues Le, Rainhard, Muneera at Ambient Intelligence group at COMNET.

This thesis would not have been possible unless my thesis instructor, Mr. Todor Ginchev, had not given me this opportunity. I would also like to thank him for good guidance throughout the thesis.

I would also like to thank Victor Nassi and COMNET IT-support staff for their technical support.

I am grateful to my friends Shishir, Manik and Prajwal for their time to go through this thesis and providing constructive comments and feedbacks.

At last but not the least, I would like to thank my friends and family for always being there for me.

Espoo, June 04, 2018

Janaki Prasad Koirala

Abbreviations and Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
CIFAR	Canadian Institute for Advanced Research
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DoG	Difference of Gaussian
GPU	Graphics Processing Unit
HL	Hidden Layer
IL	Input Layer
LASSO	Least Absolute Shrinkage and Selection Operator
NN	Neural Network
OL	Output Layer
PCA	Principal Component Analysis
RCNN	Region-proposed Convolutional Neural Network
ReLU	Rectified Linear Unit
SGD	Stochastic gradient descent
SIFT	Scale Invariant Feature Transform
SVM	Support Vector Machine

Contents

Abbreviations and Acronyms	4
1 Introduction	7
2 Background	11
2.1 Learning in neural networks	14
2.1.1 Loss function	14
2.1.2 Backpropagation	15
2.1.3 Weights Initialization	16
2.1.4 Network training	16
2.2 Convolutional Neural Network	17
2.3 CNN Architecture	22
2.4 Transfer Learning	24
3 Object Recognition	26
3.1 Non-deep-net techniques	26
3.1.1 SIFT Detector	27
3.1.2 Bag-of-visual-words	28
3.2 CNN based techniques	28
3.2.1 R-CNN	28
3.2.2 Fast RCNN	31
3.2.3 Faster RCNN	32
3.3 Food object recognition	32
4 Experiments	36
4.1 Data	36
4.2 Weights	40
4.3 Experimental Setup	40
5 Results	41
5.1 Loss Graphs	41

5.2 SmartphoneApplication	48
5.3 Discussion	49
6 Conclusions	52
A List of ‘top90’ food categories	62

Chapter 1

Introduction

Innovation is the driving factor to change the way how we live. We are currently at the age of data-boom on the time-line of computing. The advancement of technology and availability of such enormous amount of data gives us opportunities to utilise it in ways that can change our life profoundly.

Several researchers have been working in various domains to harness power provided by big data and computation. Huge amount of sensor data are produced everyday. Analysis of these data using traditional methods is not only challenging but is almost impossible. Businesses and institutions are moving away from the traditional methods of analysis to take advantage from data boom [7].

In particular, visual data from different kind of sensors has seen tremendous growth in recent years. It was forecast that eighty percent of the total web traffic would be video by 2019 [10]. Several hours worth of high resolution videos is created every second [36]. This can be seen from the fact that more and more videos are streamed from mobile devices in Periscope, Facebook, *etc.* giving rise to data growth.

In this respect, Machine Learning (ML) is becoming the primary mechanism for extracting information from data. It has been in use for different applications, from the manufacturing industry to fraud detection. The applications of these technologies appear to be virtually unlimited at the moment.

Traditionally, machine learning was limited to process only small datasets. This meant that applying the ideas from the field of Image Processing to real world problems were not feasible, mainly because we neither had the necessary image data to train the machines, nor the sufficient computation power to run the learning algorithms. However, in recent years, the computing power has grown exponentially, huge quantities of data have become available, advanced data processing algorithms have been discovered, and data generators and processors have been seamlessly integrated into the network

infrastructure and the datacenters. These developments have paved way for the deployment of software and hardware of unimaginable complexity to do everything from the very routine tasks of everyday life to the most advanced simulations.

In particular, as a result of the advancement in machine learning and deep learning, the field of computer vision has seen huge progress lately. Several disciplines have evolved to a state where algorithms may be used to automate certain real-world tasks that were previously done manually by humans. Computer vision is becoming a fundamental tool in these task automation problems, and therefore there is a huge demand of well-functioning computer vision systems at the moment.

One specific area of huge potential application is the mobile vision system that can operate in unconstrained scenarios of daily life. Computer vision is a branch of Artificial Intelligence (AI) that aims to extract information from images. The image data may arrive from many sources: frames of videos, snaps from cameras, or high resolution data from medical imaging equipment, *etc.* Computer vision applies the theories and models borrowed from several fields like machine learning, cognitive sciences, psychology, pattern recognition, *etc.*, for the creating application to solve very specific problems.

Computer vision is already applied to a wide range of applications such as face detection, anomaly detection in production plants, *etc.* In [6] an application of computer vision to detect abnormal products in the assembly line and determining the quality of the products using image processing has been presented. Similarly, [39] utilize a computer vision system to detect species of plants. These examples show that computer vision has a huge potential to be applied in different contexts where only human visual system could do the task in the past.

It has long been recognized that one of the main tasks in a computer vision systems is to locate and recognize the objects within an image, since one image may contain a number of objects in it. For a human being, the task seems trivial because Natural Selection has shaped the human visual system for millions of years for exactly this kind of tasks. Since the working of the human visual system is not very well understood, similar methods cannot be applied directly to the machines. An image is just a collection of pixels from the computer's point of view. It is hard to make machines 'understand' what changes in the pixels changes the object and what does not. It takes enormous ingenuity and computation power for a machine to handle even simple localization and recognition tasks.

One area of huge potential application of computer vision is food recognition. Food is one of the most integral parts of all humanity. Although the main purpose of food is to supply the essential nutrients to the body, eating

food also serves a wider purpose like social bonding. This can be attested by the fact that almost all the feasts, festival and holidays revolve around some unique delicacy associated with them. Moreover, in many cultures it is customary that all the members of a family have at least one meal of the day together.

Every human society has its own food and beverage and cultural practices associated with them. If a person wants to retry food from different place or culture they once tried and liked, then they can simply take a picture of the food as a memento. In future, the picture may be used in automatic food recognition software to seek the ingredients and recipe so that the person can prepare it by themselves. The ingredient search may also be useful if, *e.g.*, someone has food allergies or they cannot consume certain food for health or religious reasons. When an allergic is not certain what kind of ingredients a food typically has, the person can simply take the picture of the food and check if any allergens are present in the food.

However, the modern world has had its share of problem related with food. For example, excessive consumption of food has its down sides. The UN has reported that childhood and adolescent obesity has increased by 10 times in the last four decades [65]. Bad food habit is considered one of the main causes of this problem. Being able to devise applications that recognize food and then compute its nutritional value can have great practical application in combating obesity, a growing problem in the developed world. We believe digital devices like smartphone or smartwatch help in automatic calorie estimation and assist users to adopt good practices. The work carried out during a thesis would be a step towards that direction.

The work in this thesis will be an application of Deep Learning for food recognition. Here we have limited our scope just to the ‘European food’. In this thesis, we will create a food image dataset and use the state-of-the-art methods for object localization and recognition to create a model. This is a difficult problem because images of the same food may appear very different from each other. Source and position of illumination, position of the sensor, ingredients of food, temperature of food, *etc.*, may affect the appearance of the food. Moreover, finding a correctly labelled dataset for training our model is also a huge challenge. Furthermore, recognizing food just by looking at it may not be possible, even for human, let alone for machines, as other attributes, such as smell, texture, *etc.*, also play important roles in identifying food. Thus, there are several challenges to this problem of food recognition. One of our contribution in this thesis is to identify such problems and explore the methods from literature for solving them. Furthermore, we will use trained model to develop an Android application that could recognize food from pictures taken by users.

Structure of the thesis

We have structured our thesis in following way.

- In Chapter 1, we discussed the current trend on emergence of data, machine learning/deep learning and their applications. We set our research goal.
- In Chapter 2, we discuss traditional neural network, deep learning techniques, current state-of-art on the field and transfer learning.
- In Chapter 3, we review the traditional as well as the current state-of-art techniques for object recognition.
- In Chapter 4, we discuss about the dataset, experimental setup and frameworks used.
- In Chapter 5, we explain our experiments, also report our findings and discuss about the result.
- In Chapter 6, we conclude our thesis by reporting overall summary.

Chapter 2

Background

We begin by briefly discussing the theoretical background relevant to *Deep Learning*, which is the primary technique used in the thesis. From its origins in the field of brain biology, we trace the current state-of-the-art of deep learning applied to the problem of image recognition.

The *neural network* (NN) traces its origin to the biology of the brain, whose *neuron* cells are connected together by *synapses* giving rise to a network of neurons—the so-called neural networks. It had long been speculated that these neurons are responsible for information processing in the brain. In the 1950's an abstract mathematical model of these neurons and their synaptic network was proposed by Widrow and Hoff [67]. This was an attempt to explain the mechanism behind the information processing in the brain [5], which paved the way to the actively-researched field currently known as *Artificial Neural Networks* (ANN). Figure 2.1 is an example of a single ANN neuron.

To get an overview of how the ANN is able to process information, we begin with the description of a single neuron. A neuron (node) is defined by three parameters—the *weight* ($\mathbf{w} = (w_1, \dots, w_n) \in \mathbb{R}^n$), which is an n -dimensional vector of real numbers, the bias ($b \in \mathbb{R}$) and *activation function* $\varphi : \mathbb{R} \rightarrow \mathbb{R}$. The activation function can be both linear and non-linear. Figure 2.2 shows two classical non-linear functions *Sigmoid* and *Tanh*.

Examples of activation functions include:

- *Linear Activation Function* (2.1) is the simplest form of activation function. It is equivalent to having no activation at all.

$$\varphi(x) = x \tag{2.1}$$

- *Sigmoid function* (2.2) is also known as *logistic activation function*. This function squashes the input to values between 0 and 1. This property is useful when the desired output is probabilistic value.

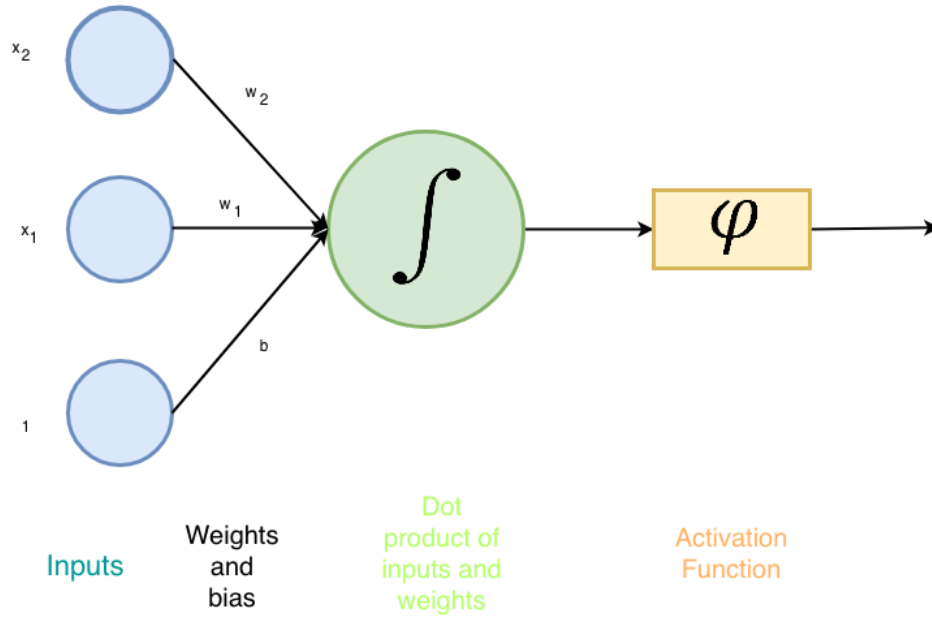


Figure 2.1: Example of neuron

$$\varphi(x) = \frac{1}{1 + e^{-x}} \quad (2.2)$$

- *Tanh* (2.3) is also known as *hyperbolic tangent function*. This function is similar to Sigmoid because it squashes the input between -1 to 1. Traditionally, this property is useful for binary classification.

$$\varphi(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad (2.3)$$

- *ReLU* (2.4) is half-rectified function. The input of real numbers less than 0 is output zero and positive input remain as it is.

$$\varphi(x) = \max(0, x) \quad (2.4)$$

Finally, the output y of any such neuron is defined as

$$y = \varphi(\mathbf{w} \cdot \mathbf{x} + b), \quad (2.5)$$

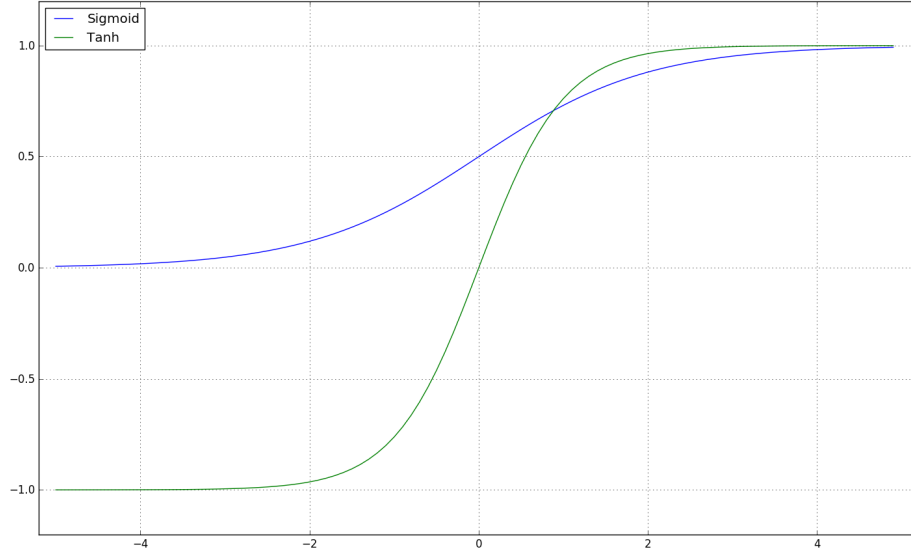


Figure 2.2: Graph of Sigmoid and Tanh functions

where $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ is the input vector¹ to the neuron. An artificial neural network (ANN) is constructed by the repeated process of using the outputs of one or more such neurons as the inputs to other neurons. In a *feed-forward ANN*, there are no loops. This means that the output of *every* neuron depends just on its input vector, weight, bias and the activation function *but never on its own output*.

A feed-forward network can be seen as having several layers. The nodes which are directly fed with the input data constitute the *Input Layer*. It is easy to see that the *Input Layer* (IL) is the input data.

The outputs of the IL is fed to the first *Hidden Layer* (HL), whose output has the same dimension as the number of neurons in this layer. Similarly, these outputs can be fed to the next HL and so on and so forth. In a *fully connected ANN*, the output of every layer is fed to every neuron of the next layer. The *Output Layer* (OL) comprises of those neurons whose output are not fed to any other neuron. Consequently, the output of the n th HL \mathbf{y}_n is given by

$$\mathbf{y}^{(n)} = \varphi^{(n)}(\mathbf{W}^{(n)\top} \mathbf{x}^{(n-1)} + \mathbf{b}^{(n)}), \quad (2.6)$$

where $\mathbf{W}^{(n)}$ is the matrix formed by concatenating the weight vectors of the

¹The column vector is used for the matrix representation of vector in this thesis.

n th HL neurons, $\mathbf{x}^{(n-1)}$ is the output of the $(n-1)$ th HL and $\mathbf{b}^{(n)}$ is a column vector formed by stacking the bias terms of the n th HL neurons in rows. The function $\varphi^{(n)} : \mathbb{R}^m \rightarrow \mathbb{R}^m$ is defined as

$$\varphi^{(n)}(\mathbf{x}) = (\varphi_1^{(n)}(x_1), \dots, \varphi_m^{(n)}(x_m)) \quad (2.7)$$

where $\mathbf{x} = (x_1, \dots, x_m)$ is an m -dimensional vector and $\varphi_i^{(n)}(\cdot)$ is the activation function of i th neuron of n th HL for $i \in \{1, \dots, m\}$.

It has been proved that with the suitable choice of weights, biases, activation functions and the number of hidden layers, a feed forward ANN can approximate a large class of continuous and non-continuous functions [25, 51]. This *universal approximation* property of multilayer perceptron is useful in the context of statistical machine learning which we will discuss in the following sections. Example of feed forward network is shown in Figure 2.3.

2.1 Learning in neural networks

The universal approximation theorem states that a feed-forward ANN with a single hidden layer containing a finite number of neuron can approximate continuous functions on compact subsets of \mathbb{R}^n , if the activation function is continuous, non-constant, bounded, and monotonically-increasing. Since ANN is capable of approximating a large class function by adjusting the weights and biases (hereafter called the *parameters*) and the activation function, a natural question follows: Are there any well-defined ways to achieve so? Fortunately, the answer is ‘yes’ and the process of determining these parameters when we know the functional values t_n for a set of inputs \mathbf{x}_n is called *supervised learning*. After ‘learning’ these parameters, an ANN is able to ‘predict’ the output for unknown inputs.

2.1.1 Loss function

For the process of learning, we first choose a suitable activation function and then define a *loss function* (also known as an *error function*), whose value is optimized (maximized or minimized) by changing the ANN parameters. It is desirable to have a loss function that is convex with continuous first derivative. This simplifies of the optimization, which can be performed using the gradient descent or related algorithms. One such loss function is the

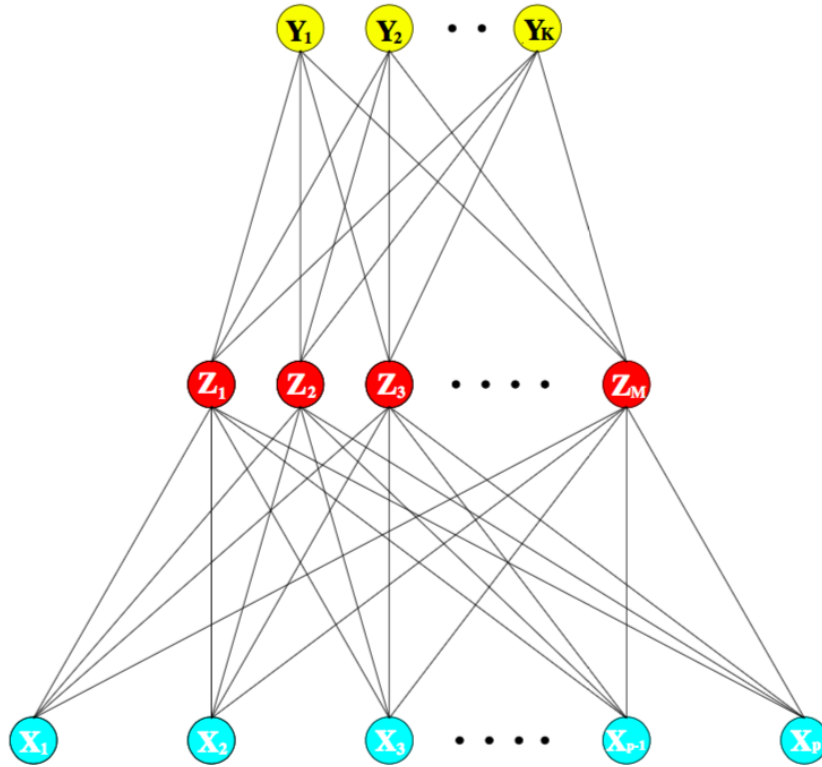


Figure 2.3: Example of single layered feedforward network. Here X s are the inputs. Z s are the hidden units and Y s are the output of the network. The image is taken from [18].

squared-sum error function defined as

$$\mathcal{E} = \frac{1}{2} \sum_{n=1}^N \|f(x_n; \mathbf{w}, \mathbf{b}) - t_n\|^2, \quad (2.8)$$

where N is the number of samples, x_n is the n th input sample, t_n is the output for the n th input, $f(\cdot; \mathbf{w}, \mathbf{b})$ and $\|\cdot\|$ represent the ANN output and the L^2 norm of \cdot , respectively. The optimal values of the parameters— \mathbf{W}^* and \mathbf{b}^* —are obtained by solving an optimization problem

2.1.2 Backpropagation

A smooth and convex loss function can be optimized using the gradient optimisation method provided that the derivative of the loss function is known.

Backpropagation is the application of the chain rule of calculating the derivative in ANN. In backpropagation, the information first flows from the input to the output and then in the reverse direction. The difference of the predicted output and the actual output (the error), is then determined. The error made by network to predict true value or label is differentiated with respect to all the parameters and weights of the network and then the weights are adjusted accordingly.

Inputs/Hidden Units/Output

Inputs and *outputs* are the first and last layers of network respectively. Hidden units are the processing units of networks. The input to the layer are transformed multiplied by weights in these middle layers to produce usable output. The weights are also adjusted here during the network training. The main processing happens in middle layers of the network, which are usually hidden. The layer gets its name from this property.

2.1.3 Weights Initialization

We have to assign some initial values to weight vector \mathbf{W} and bias term \mathbf{b} before we start our training process. This initialization of weights also plays vital role in optimization and the training time of the network. If we initialize our weight vector with good initial values, the network converges quickly. Normally we assign the random values to the parameters in the beginning. But we must be careful not to assign too high values, which might lead the network to always predicting the same labels the signal. It should not be too low either, which might lead to no signal appearing at all at the output. This is still an active area of research in convolutional neural network (CNN), which is discussed in greater detail in Section 2.2.

2.1.4 Network training

The basic principle of network training is to train using an iterative process until convergence, *i.e.*, until the values of the parameters do not change significantly between successive iterations. Some heuristic is generally defined to see if the convergence has been achieved. These heuristics are also known as the *stopping criteria*. One of the popular stopping criterion is to track the loss on the validation data. If there is no better result on the validation data set after N of iteration, we can halt the process and \mathbf{W} and \mathbf{b} become our parameter.

2.2 Convolutional Neural Network

Convolutional Neural Network (CNN) is the next stage of evolution of ANN. However, they are not very different from the traditional neural network. They too have inputs, weights and biases just like the traditional neural networks. The operation like dot-product and non-linearity are applied to the inputs and weights and their sums respectively [43, 62].

The CNN was inspired by the findings of Hubel and Wiesel based on their experiments on the visual cortex of cat [26]. They proposed the simple (S) and complex (C) cell structures of the neurons on visual cortex. The simple cells (in visual cortex) activate when the cat see simple shapes but complex cells use the combined information of S cells and activates when it sees shifted or rotated version of the original shapes. In the early 1980s, Fukushima proposed a neural network that tried to simulate visual cortex proposed by Hubel and Wiesel. [27]. Finally in 1998, Le Cunn, Bottou, Bengio and Haffner proposed Convolutional Neural Networks. [40].

The CNN is slightly different from the traditional feed-forward neural networks. One of the basic objectives in CNN is to build invariance properties into the ANN. This is achieved by creating models which are not sensitive to inputs that have undergone certain transformations. The CNNs are successfully applied to several different applications including image/video classification, voice recognition, language modelling, *etc.*

As stated earlier, the idea of CNN was proposed already on 90s [40]. However has become more popular only in the recent years. CNNs generally require huge amount of data for networks to converge and consequently, significantly larger computation (CPU) time is needed. This was one of the biggest limitations for the adoption of CNN during the 90s. The exponential growth in processing power has boosted the use of CNN in the recent years. Most significantly, the development of Graphics Processing Units (GPU) has helped tremendously in this regard. Moreover, as stated in Chapter 1 huge amount of sensor data has been generated in the recent years. These two developments, *i.e.*, advancement in the processing capability of the CPU and the availability of large amount of data, have contributed for CNN to be more adopted now-a-days.

Many frameworks [15] are available for deep learning in CNN. Caffe [30], Theano [2], TensorFlow [1], Keras [9] are few of the frameworks currently popular. In deep networks the computation is performed on multiple layers to learn the features from the data during the training process. The multiple layers makes it possible to learn multiple representation of the data. The deep learning methods have outperformed the classical machine learn-

ing methods in multiple domains, including speech recognition, visual object recognition, object detection, computational genomics, *etc* [55]. In CNN, as in the traditional ANN, backpropagation algorithm allows to change the state of the parameters to adjust to the current state of data, hence learning the features [41].

The neural networks were not a very popular of machine learning during the 2000s. However, after the publication of Krizhevsky's article [38], which showed a huge improvement in performance of image recognition in the ImageNet dataset, many researchers realized that CNN has a huge potential for object recognition/classification [38, 71].

In this thesis, we focus on CNN specifically for the application in computer vision. In this case, the inputs to the CNN are images which is a 3-dimensional data. The architecture of the network makes it possible to tweak the network and scale. As an example: If the image had dimensions $32 \times 32 \times 3$ (CIFAR-10) it would require 3072 weights. Similarly, for the input images of dimensions $200 \times 200 \times 3$ it would require a total of 120000 weights. Clearly, this does not scale with image dimensions. When the neurons are arranged in 3-dimensions (height, width and depth) in CNN architectures, it is possible to scale up. Here, the depth refers to dimension of the activation volume. The CNN architecture is defined in such a way that the final layer of weights are manageable for fully the connected layer.

Inputs

The input image is usually cropped into a square during preprocessing. If it is not possible to get a square image by cropping, then image is squeezed into a square. The image is then feed into the network as a 3-dimensional matrix for height, width and the 3 channels of RGB images. The input data can also be 2-dimensional, if the image is black and white. However, it will still be treated as a 3-dimensional one.

Convolution Layer

In the convolution layer, the hidden layer is convolved with part of the input image. These layers are also known as filters. The number of the filters is a hyperparameter. Each filter is convolved with whole input image and the output is stacked, increasing the size of the depth layer. The depth value is equivalent to the number of the filters. If the input image is $200 \times 200 \times 3$ the output of the layer, also known as feature map, would be $200 \times 200 \times 20$ if 20 filters and appropriate numbers of paddings are used.

Activation Functions

Different activation functions are applied in these CNN architecture. This is also an active area of research. Most commonly used activation functions used in CNN include ReLu, where negative values are made zero or $\max(0, \text{value})$, leaky-ReLu, a probabilistic approach to restrict few numbers to under zero, *etc.* Note that these are one-to-one operations which does not affect the size of the feature map.

Pooling Layer

Pooling layer is used as a tool to down-sample the feature map in height and width. The most common pooling method is known as max-pooling. If the size of feature map is $200 \times 200 \times 20$, the output feature map results in the size $100 \times 100 \times 20$ if max-pooling is performed on a 2×2 window.

Fully Connected Layer

As the name suggests, all the neurons on this layer are connected to the previous feature map as in the classical ANN. If the feature map is not one dimensional (*i.e.*, $N \times 1 \times 1$), the feature map is flattened in a one dimensional vector. The size of this layer will depend upon the output size. For example, if we need ten different class labels, the size of this layer would be 10. The probability for each of the classes is calculated on this layer.

Depending upon the requirement, the fully connected layers can be engineered to for either regression or classification.

Regularizations

If the network is trained for long enough time, it performs extremely well with the training data. However, doing so could increase the error for the test data. This is known as the problem of *overfitting*.

Ideally, we would like the network to perform well not just with the training data but also with the new inputs. This is, in fact, the core idea behind machine learning, *i.e.*, to predict good results even with the unseen data. In general, the networks should be optimised in such a way that it gives good performance with the test set or new inputs.

Regularization is used to combat the problem of overfitting. This can be achieved, *e.g.*, by adding a regularization term to the cost function to be minimized. This added term to the acts as mechanism for weight decay. This

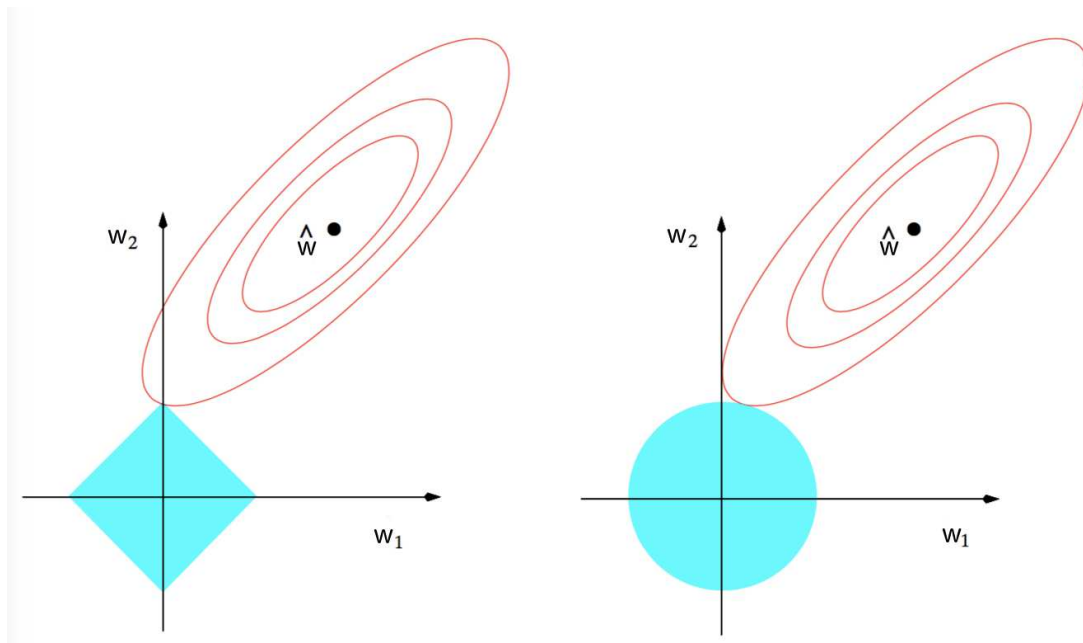


Figure 2.4: Visual Analogy of $L1$ (left) and $L2$ (right) loss. The contours are error functions and the solid blue region are the constraints. The image is taken from *The Elements of Statistical Learning* [18].

way, the regularizer is the constraint added to the learning algorithm so that the variance is reduced while the bias is not overly increased.

Note that the regularizer is a function of the network-parameters. Consequently, the parameters always remain in control while learning. In summary, the main goal of regularization is to not allow the network to overfit the training data [21]. Mathematically, the general cost function with regularizer becomes (2.9)

$$\tilde{J}(\boldsymbol{\theta}; X, \mathbf{y}) = J(\mathbf{w}; X, \mathbf{y}) + \alpha\Omega(\boldsymbol{\theta}) \quad (2.9)$$

where the last term $\alpha\Omega(\boldsymbol{\theta})$ the regularizer. The regularizer term for L2-Regularization is given by

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2} \|\mathbf{w}^T \mathbf{w}\|_2^2. \quad (2.10)$$

L2-regularizer is one of the most commonly used regularizers. It is used with squared-error loss. This combination of L2 regularizer with squared-error loss function is usually referred as *ridge regression*. Another common regularizer is *L1 – Regularizer* defined as

$$\Omega(\boldsymbol{\theta}) = \|\mathbf{w}^T \mathbf{w}\|_1 = \sum_{i=1}^N |w_i|_1 \quad (2.11)$$

which is sum of the absolute value of all the parameters. This regularizer penalises the size of model parameters. The famous LASSO integrates this L1-regularizer with least-square cost function. In general, other norms can also be added as the penalty term to the cost function to penalise the model parameters. In Figure 2.4, the graphical representation of L1 and L2 regularizer is shown. The point of intersection of solid blue region and the ellipses is the optimum point.

Dropout

The deeper a network is, the higher the number of the network parameters gets. When there are more parameters, there is chance that the model is overfitted or model tries to ‘remember’ the data than to learn from the it. Dropout is one of the regularization technique to prevent the network from overfitting. The idea behind dropping is to drop a certain number of connections in the network during the training. This prevents the network from favouring certain connections all the time. This regularization technique has improved results in many supervised learning tasks including computer vision, speech recognition, computational biology *etc* [61].

Batch Normalization

We already discussed the importance of the number of input example or training sizes. The size of the training data can be hundreds of gigabytes of data. These data can not be processed at once with the processors. The deep networks process training/input data in batches of appropriate sizes the CPU and/or GPU can handle the computations.

The distribution of the input data is different for each layer. Hence the input data changes on each layers of every iteration of the training, which adjusts parameters accordingly. One approach to get around this issue is to initialize the parameters with well crafted weights and have a very small learning rate at the same time. This causes the computing resources to be used for longer durations. This phenomenon is known as covariate shift.

Another way to deal with the problem is to the normalize the input data. In *batch normalization* the input into each layer is normalized before further processing. In some cases, the batch normalization method eliminates the need to use dropout as the regularizer [28].

2.3 CNN Architecture

We briefly discuss about a few CNN Architectures that are comprise the current state-of-art. CNN architectures is an active field of research. We have mentioned only few of them.

AlexNet

Alexnet [38] was proposed by Krizhevsky *et al.* This paper was the winner of the 2012 ImageNet Large-Scale Visual Recognition challenge (ILSVRC). It reported the top 5 test error of 15.4%. The second best result had the error rate of 26.2%. This was huge performance improvement.

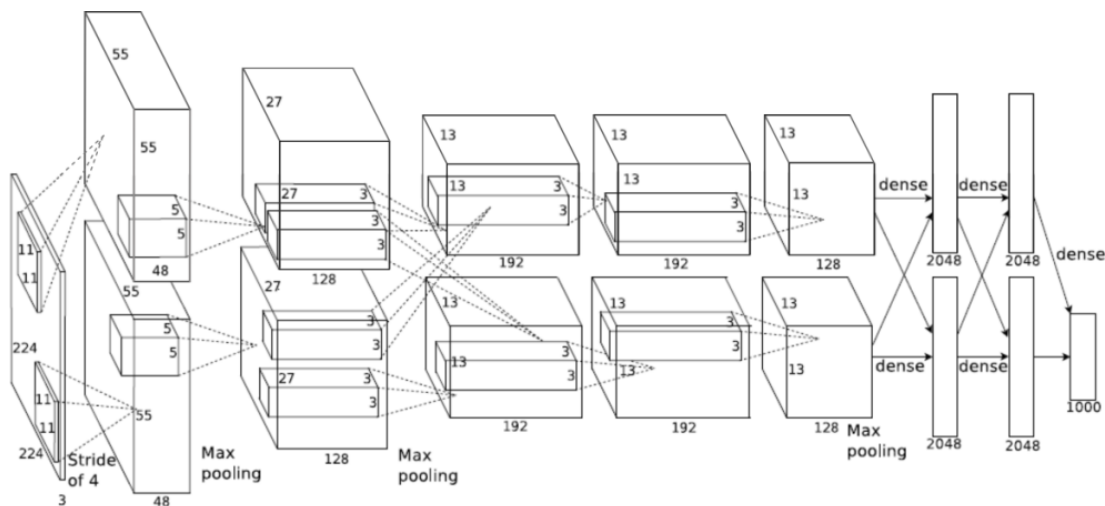


Figure 2.5: Architecture of Alexnet. The image was taken from [38]

The architecture of AlexNet is shown in Figure 2.5. The training process was computationally expensive compared to present time. We can see two streams of convolutional networks. Another feature of AlexNet was that the rectifier linear unit was used as non-linear activation function. The researchers had used augmentation to solve the image transformation problem. They had used dropout to avoid the overfitting problem. The model was trained on batch of the dataset of 15 millions images using SGD method on GTX 580 GPUs for five to six days.

The performance achieved by using this method was an outstanding 10.8% on top 5 error. This was a demonstration of CNN's capability to solve ILSVRC, which was considered a hard problem for a long time. After

the publication of this paper, more researchers in computer vision started to use CNN in their research[74]. This is also backed by the fact that this paper has been cited more 20000 times.

ZF Net

ZF net [73] was proposed by Zeiler and Fergus. This paper proposed improvements and fine tuning on certain aspects of AlexNet. It was the winner of ILSVRC in 2013 with the top 5 error rate of 11.2%. The architecture of ZF Net is shown in Figure 2.6. Although the architecture was different from AlexNet in very minor ways, this model was trained on only 1.3 million of images compared to 15 million images for Alexnet. They also changed the size of the filter from 11×11 in AlexNet into 7×7 in ZF Net. More filter were used on upper layers. ReLu was used for activation and cross-entropy loss was used the error function. This model was also trained on GTX 580 GPU but for twelve days. In the Faster RCNN model later used in Section 3.2.3 this model has been utilized.

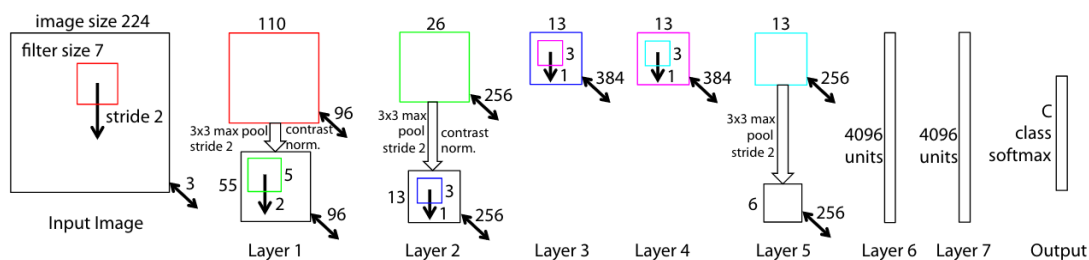


Figure 2.6: Architecture of ZF. The image is taken from [73]

VGG Net

VGGNet [59] was proposed in 2014 by Simonyan *et al.* They replaced the 11×11 Alexnet and the 7×7 ZF Net filter by a 3×3 filter in their model architecture. Simpler but stricter design decision were made for this network. The filter were always 3×3 with a padding of 1, along with a 2×2 max pooling layer with the stride of 2. Consequently, the depth of the network grows while spatial dimension shrinks. This performed well not just for the recognition task but also for the localization task. This had 7.3% of top 5 error rate on ILSVRC challenge.

GoogLeNet/Inception

GoogLeNet/Inception model was the winner of ILSVRC 2014 with top 5 error rate of 6.7% [63]. This was developed by researchers at Google. The CNNs that were used in earlier models such as AlexNet, VGG, *etc.* had simple stack modules of convolution and max-pooling. However, GoogLeNet contains a stack of 22 convolutions and a new module called the Inception Module. This model is composed of parallel computations of convolutions of different shapes and max-pooling. Output from all these modules is concatenated at the end of each Inception module.

ResNet

ResNet was deep network architecture with 152 layers [22]. This was developed by Microsoft Asia. It was the winner of ILSVRC 2015 with the top 5 error rate of only 3.6%, which is considered better than human level accuracy [54].

The researchers used residual modules for this architecture. Normally in CNN the input is transformed into a different dimension when it reaches at the output layer. On the other hand, the Residual Layers in ResNet calculate the changes in the input. This is then added to the input to produce the output. The researchers believe that this sort residual mapping is easier to optimize.

This architecture is one of the best state-of-art architectures because of its substantial performance improvement over its predecessors.

2.4 Transfer Learning

The main idea of transfer learning is to use the previously learnt ideas in new problems. The modern CNN needs quite a lot of data for good performance [38]. It is not possible to satisfy the demanding nature the CNN with small or moderately sized datasets. Although the CIFAR data has 50K images, it is just considered as a toy dataset in computer vision problems.

However, the collection of such a huge number of data (images) is difficult. So it is not always possible to have a huge amount of problem-specific data and train the models from the scratch. In such cases, transfer learning is an easier and practical approach to consider.

One of the methods of deep learning is to try to figure out the unknown distribution of features of the input, mostly at a higher level. The idea of transfer learning is to turn these high level representations into more abstract

features so that the features are invariant to more diverse features in the new input[4]. There are different approaches of transfer learning in machine learning, data mining, computer vision.

In deep learning networks, when we lack a huge dataset, the architecture and the weights of a well trained network can be used as the basis or the starting weights in new but similar problems. In this approach, we first remove the last layers of the old networks and replace it our new problem-specific design. Then we train the small dataset on it. This means that we treat the pretrained network as an unsupervised network. One concrete example of this approach would be to use some trained networks on the ImageNet dataset and use it to train for recognizing the food objects.

As stated earlier, it is very expensive computationally to train a CNN. In a image recognition problem, a CNN requires large numbers images for good results [56]. Researchers do not always have enough images for domain-specific problems. It requires weeks to months to train convolutional neural network models. It is a common practice to use pretrained weights on large datasets as initial weights for a new CNN. This processes is known as transfer learning [43].

Let us take an example. The final fully-connected layers outputs 1000-class scores in ImageNet. The fully connected layer can be removed from this and instead the convolutional layers can be used as the feature extractor from the images. It requires the knowledge of the pre-trained weights and architecture of the convolutional network. Additional convolutional network can be added according to the requirement followed by the fully-connected layer. Then entire network can be trained on new dataset. This process is known as fine-tuning since the network adjusts the weights by back-propagation on the new training data [43]. *Model zoo* [29] is a repository of weights of pre-trained models for many data sets for the Caffe framework, from which the weights can be taken to initialize a model for own networks.

Researchers have categorized different settings or methodologies in transfer learning. *Inductive transfer learning* is a setting where the source task is different from the target task of the same or a different domain. In *transductive learning* the source and target tasks are the same, but the problem domains are different. In the *unsupervised transfer learning* the source and the target tasks are different but related is some ways [50].

Chapter 3

Object Recognition

Object recognition comes very naturally to human eyes. Due to their highly evolved mammalian visual cortices [66], they can look around the environments and recognize objects. However, the task of object recognition appears to be exceptionally hard for machines. It has been acknowledged as one of the hardest problems in computer vision [48]. Early attempts at solving the problem were constrained by the limitation of the computing power.

The images are represented on a computer as matrices of numbers. Images of the same object taken from same the same device in a short span of time are highly likely to be different pixel-wise because of small changes in the environment, the error in camera sensor, the light conditions, *etc.* Thus, even under very controlled conditions, a simple pixel comparison cannot reliably tell whether two pictures represent the same object or not. Hence, a very simple problem for a child can turn out to be very hard for computers.

In the 1970s, Hubel and Wiesel showed that edges are important features in visual object recognition through their experiments on cats and monkeys [27]. Researchers used this knowledge to make rules to find objects in images. Certain rules (features) were searched in an image to recognize object in it.

3.1 Non-deep-net techniques

We first review several detection techniques which do not use ANN. These are all classic techniques that were popular and performed well even when the computing resources were scarce.

3.1.1 SIFT Detector

Scale Invariant Feature Transform (SIFT) detector is a recent (developed in 2004) but very popular (more than 10000 citations) object detection and retrieval tool. The detection is achieved in four primary steps. At first, key interesting points over the image are searched thoroughly. This is achieved by calculating the difference-of-Gaussian technique and building image pyramid. The low contrast points and edge responses are rejected in this localization step. The interesting key points found are fitted to the model for determining its scale and location. The stable key-points are then prioritized. Following this, at least one orientation is attached to all the interesting key points. Finally, the image data is transformed according to this orientation, scale and localization for each feature/interesting points using the magnitude and direction of the gradient. The image gradient are calculated around the key interesting points locally. This information is transformed using histogram of magnitude and direction into smaller key point descriptor which helps in case of shape distortion and change in illumination [44, 45].

These methods can find about 2000 features in an image of 500×500 pixels. At least three features should match to correctly recognize (small) object in an image. Hence, the number of features identified in each image greatly affects the performance of the algorithm. These features are extracted from the reference images first, and then stored in a database. The features from future images are extracted, which are the compared against the existing features in the database. The similarity of the features are compared with euclidean distance or nearest neighbour algorithm between the candidate features [44, 45].

Many variants of SIFT detector can be found in literature. Two of them are discussed as follows:

PCA-SIFT Detector

Principal component analysis (PCA) is one of the most popular dimensionality reduction techniques. The application of PCA on SIFT detector, instead of the original histogram for gradient patches, reduced the dimensionality of the features by a large margin. The input vector of an image is created by concatenating the horizontal and the vertical gradient maps. Since this method reduces the dimensionality, it leads to less computation time and faster performance [35].

SURF Detector

The SURF detector also uses the key-point descriptors like the SIFT detector. But unlike the SIFT detector, the SURF detector builds a stack with same resolution of image. The layers are filtered using the box filter approximation of the second order partial derivatives. The key-point is matched using the euclidean and/or the nearest neighbour methods like in the SIFT detector [3].

3.1.2 Bag-of-visual-words

In the Bag-of-visual word method, local features from the images are gathered using difference-of-Gaussian method, which are then PCA transformed to get only about 30 coefficients, thereby reducing the dimension. The color information is also taken into consideration in this technique. The frequent patterns in the available training data are extracted in an unsupervised way using a mixture of Gaussian model. For new images, histogram of the feature is produced using the local information of the image and the previously trained mixture of Gaussian model. The histogram is then used to find the category of the image in the second phase [13, 69].

Most of the techniques described above were based on content categorization. These techniques mostly used the information of local patches, the image intensity, the gradient wavelet and the histogram equalization. We can find other techniques also in the literatures, *e.g.*, ORB [53].

Since the advancement in the processing power, specially with GPUs, memory and I/O devices, researchers are using a data-driven approach in image recognition [55]. Since the year 2012 the data-driven approach based on neural networks has outperformed the classical rule-based methods in computer vision problems [38].

3.2 CNN based techniques

Now we discuss about the localization and object detection methods backed by CNNs. As stated earlier, these represent recent development in the object recognition problem.

3.2.1 R-CNN

The use of convolutional neural networks for object recognition became popular after huge improvements on different computer vision problems were

reported in 2012 [71, 74]. These improvements are often reported for new algorithms and methodology on the standard datasets. The image datasets like Imagenet [11], Pascal VOC [16], MNIST [42] have become the benchmark datasets. In addition to the advancement in computing power, the performance improvement has sky-rocketed in the recent times after researchers started to take a competitive approach and devised techniques and architectures to outperform each others. Currently it is one of the most actively researched areas in machine vision.

Object detection with convolutional is not as simple as the classification task. It requires the localization of the object in the image and then classification of that local object. In order to tackle this problem, we use regular CNN with additional features.

First we need the coordinates of the objects inside the image. That is, we need four numbers to find the localization information, namely, the coordinates x and y and the height, and width of the bounding box. We can use regression to find these values. Moreover, two loss functions are needed to localize and identify the objects. Girshick and his team have been working actively to develop RCNN Region-proposed convolutional neural network (R-CNN) [20].

The R-CNN family of object detection and localization method, like other object recognition methods, assumes that the possible objects in the images are known beforehand. R-CNN works in three modules and starts from region proposals. Region proposals are the possible regions within the images which contain the possible objects. R-CNN uses selective search as a tool for the region proposals.

Selective search uses traditional image processing techniques for the region proposals. The main goal of this method is to look into the regions in the image where potential object is located. It uses a hierarchical bottom-up grouping method on the image to obtain a region by calculating the similarities between neighboring regions and merging together similar regions. It also uses diversified techniques to yield regions. One technique is to use complimentary color spaces with a range of invariant properties. Images with different lightning condition and different scenes are taken into account with this technique. It uses different measures of similarity and different initial regions to get better results. Selective search is very likely to find the objects if there is any. It also has a high recall [64].

Selective search returns potential 2k Region-of-Interest (RoI). Most of these regions are just noises. RCNN takes 227×227 pixel mean-subtracted image as input. Region proposals can come in different shapes and sizes. All the regions are packed in the required image size. Each region-of-interest is propagated through five convolutional layers and two fully-connected layers.

RCNN uses the Caffe [30] implementation of CNN by Krishevsky et al. [38]. Next, 4096-dimensional features for each region proposals are extracted.

In the test time, around 2k regions proposals are generated and forward-propagated in CNN. Each extracted feature is scored for each class using a SVM. A greedy non-maximum suppression is performed for each class independently to reject regions if it is an intersection-over-union overlap. All the CNN parameters are shared across all the categories. The calculated feature vectors are low-dimensional compared to bag-of-visual-words encoding or spatial pyramids. SVM is used to predict the category labels. Apart from the category, RCNN also predicts the bounding box. Moreover, initial proposals might not be perfect. So RCNN predicts the correction to initial proposals as well.

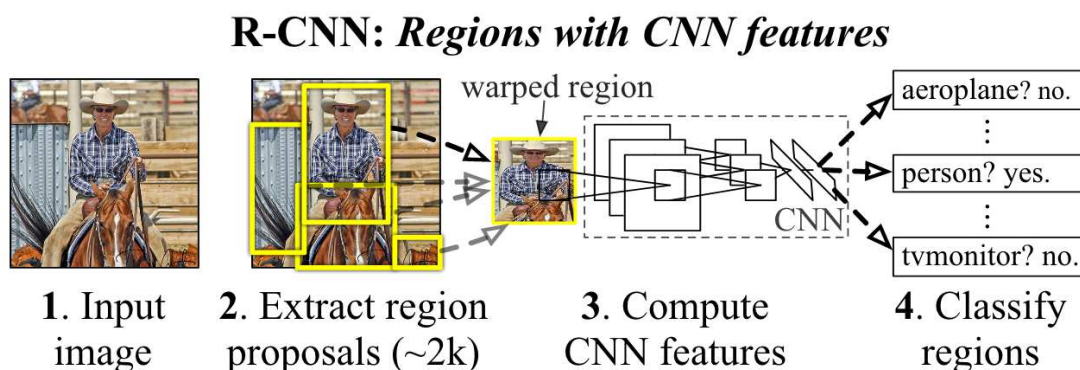


Figure 3.1: Object detection system overview of RCNN. Workflow of the RCNN taken from the R-CNN paper [20]. First input image (1) is used to generate region proposals (2) which are then used to compute the CNN features. Finally these features are used to classify regions.

RCNN was the state-of-the-art method when the reports were first published in 2014. However, it still had many drawbacks. For instance, it used disk space to store intermediate results causing it to perform frequent time-consuming I/O operation on disk. Since the initial proposal has a large number of region proposals and each region needs to be propagated in CNN, it is also costly computationally. The whole recognition process was dependent on the initial region proposals, which were not learnt but estimated using traditional image processing methods. The total RCNN is a multi-stage pipeline shown in Figure 3.1.

SSPnets tried to improve the drawbacks on RCNN. The *SSPnets* takes the whole image as an input instead of just the region of interest. Following

this, objects are proposed from the feature vector that is extracted from the feature map. Fixed-sized feature, extracted from the feature map by max pooling, are concatenated like in spatial pyramid pooling. This improved the performance of RCNN by 10 to 100 times. However, SSPnets itself is a multi-stage pipeline. One another drawback was that the convolutional layers were not updated.

3.2.2 Fast RCNN

The authors of RCNN came up with a new idea to improve the whole training process into a single-stage task. This improved version of RCNN was reported as Fast-RCNN. The training process is a single stage-task where all the convolutional layers gets updated during the training process. The main bottleneck of disk I/O was completely removed from the pipeline, contributing to much a faster training and testing times [19] compared the previous RCNN model. Experiments revealed that [19] this model was 10 times faster to train compared to RCNN. Furthermore, it was also discovered that computing region proposal took less than a second in best case to two seconds in the worst case. However, the drawback was that was the region proposals still had to be determined using some traditional methods. The pipeline of Fast RCNN is presented in Figure 3.2.

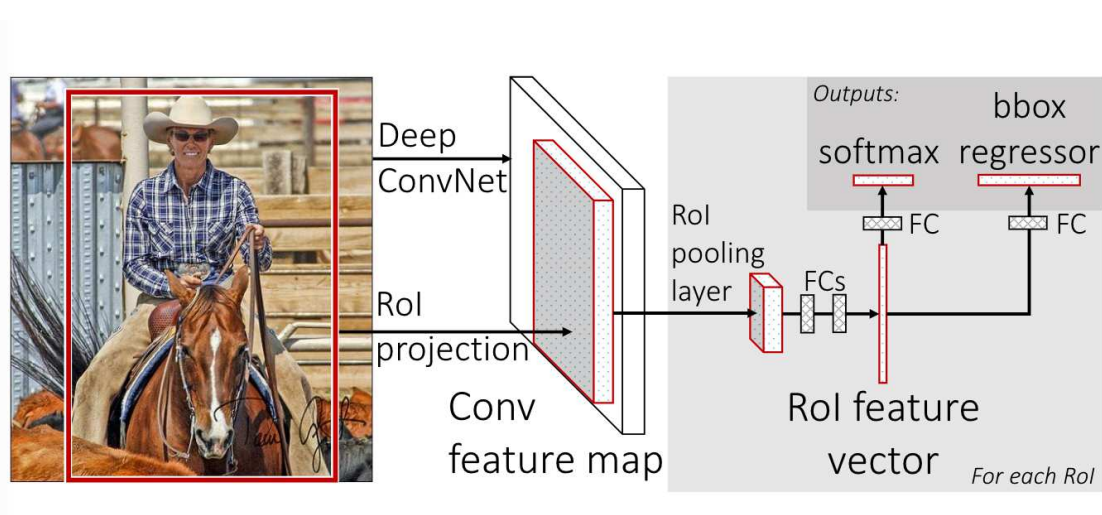


Figure 3.2: The RoIs and the input image are the inputs for fast-RCNN convolutional network. The RoIs are pooled into features maps which are further mapped into fully connected layers. These are used to predict class (softmax) as well as bounding-boxes (regressor). Image is taken from Fast RCNN paper [19]

3.2.3 Faster RCNN

Faster RCNN [52] is a more advanced version of Fast RCNN. The framework supports stochastic gradient descent, AdaDelta [72], AdaGrad [14], Adam [37], Nesterov [49] and RMSProp [23] as the optimizers. The algorithm takes the full image as the input. Following this, the input is propagated through all the convolution layers of the network. The main improvement of the technique is that the region proposal network (RPN) is placed on top of the convolution network. Faster RCNN could just ‘look’ at the last convolutional feature map and produce the region proposals. It then classifies the input as ‘object’ or ‘not object’, using binary classification loss. Finally, if recognized as an ‘object’, the classification decision and the bounding box for the object is produced. Consequently, Faster RCNN is computationally much cheaper than its predecessors. The pipeline of Faster RCNN is shown in Figure 3.3.

3.3 Food object recognition

The main objective of this thesis was to localize and recognize food objects in image. Several researchers have been working on this for a long time.

Chen *et al.* [8] prepared the Pittsburgh Fast-Food Dataset (PFFD). They selected 101 food items from different chain restaurants. Each food item was bought on three different dates. They created two baselines for the recognition task. One was based on histograms of colors and another on bag of SIFT features. Both features were later used with SVM classifier.

Shroff *et al.* [58] proposed, DiaWear, a mobile or wearable camera-based semi-automatic food recognizer. They also used a reference object on the side of food item for size-reference. They worked with only 4 categories of food, namely, Hamburger, Fries, Chicken nuggets and Apple pie. They pre-processed the image to remove the background noises and then propagated the preprocessed image through an ANN. Separately, they had experimentally achieved a threshold value for output of the network. If the output of the network was above the threshold, they accepted the result as a valid food item. They calculated contextual weights from each food and then used the Law of Total Probability for calculating the probabilities of each class. A lookup table was used to display the calorie range of highly probable food. Finally, they reported an improvement of 6.97%, 12.82%, 12.19% and 2.17% in Hamburger, Fries, Chicken Nuggets and Apple Pie, respectively on contextual setting compared to non-contextual setting in PFFD baselines.

Yang *et al.* [70] attempted to utilize the spacial relations of ingredients of food. They worked with sixty-one categories of food. Their approach to

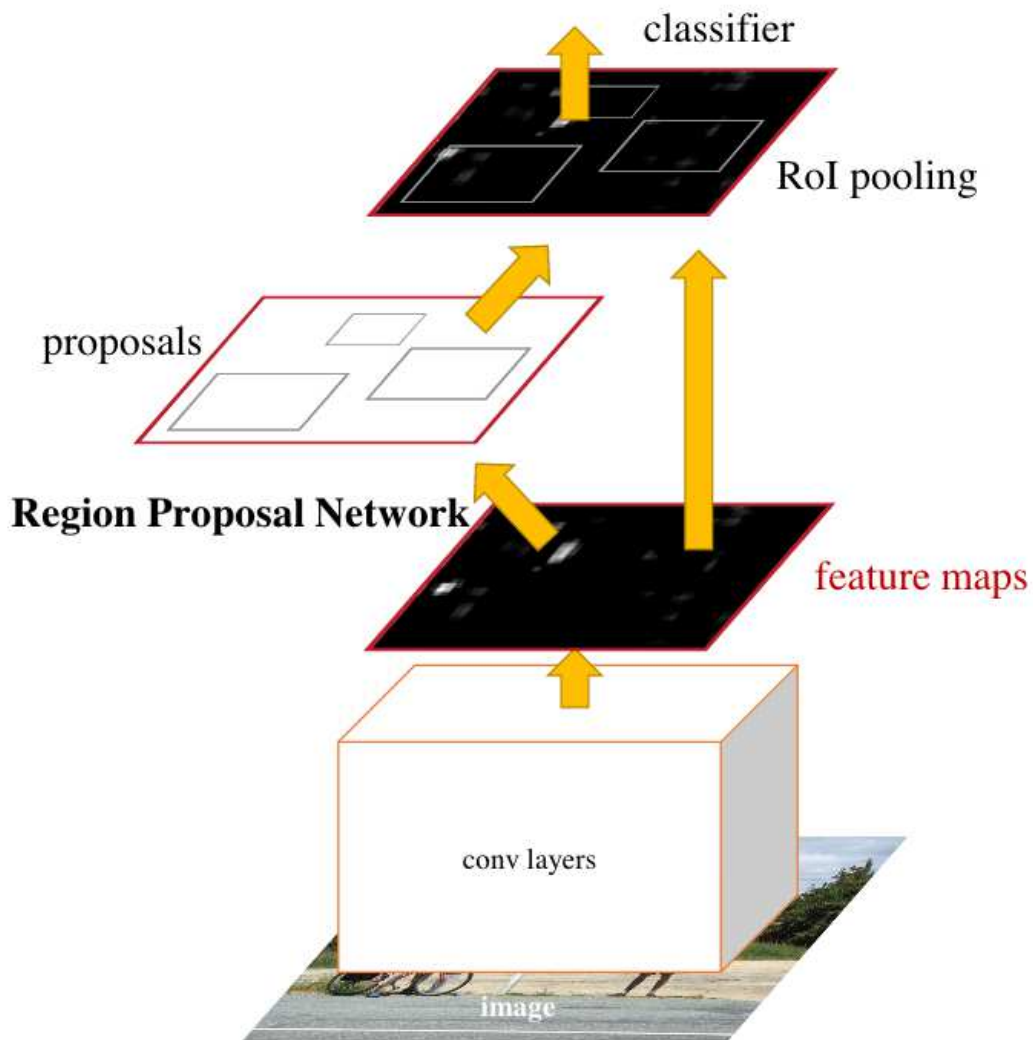


Figure 3.3: There is different region proposal layer in faster RCNN. The feature map is used to generate region proposals. The RoI pooling is created from the feature map and region proposal network. The final part flow is similar to Fast-RCNN. Figure is taken from Faster RCNN paper [52]

recognition was to assign the pixels to one of these categories probabilistically using the STF [57]. They extracted the pairwise statistic of local features (distance, orientation, mid-point category, between-pair categories) to create a multi-dimensional histogram. These histograms were used to classify the images with a multi-class SVM model. The researchers used the Pittsburgh

Fast-food Image Dataset [8]. They used bag of SIFT and color histogram as their baselines. They reported an accuracy of upto 28%.

Keiji Yanai and his research group at the University of Electro-Communications, Tokyo, have also been working on food-image recognition and automatic calorie-estimation. They have a series of academic papers on food-image recognition which are discussed as follows.

Joutou *et al.* [31] studied recognition of food-images with multiple kernel learning (MKL) [60]. They used multiple kernels to combine image features like bag-of-features, colour histogram, Gabor texture, SIFT-features, *etc.* They trained MKL-SVM models on the extracted features from the training images. They extracted features from the test images and used the trained MKL-SVM models to classify the food image. The researchers achieved 61.34% classification accuracy on 50-categories of food, which they claim was good enough results to be used in mobile applications.

Hoashi *et al.* [24] integrated seventeen kinds of different image features and used the same techniques as Joutu *et al.* [31] to classify 85 categories of food items. They report 62.52% of accuracy which is an improvement, while the number of categories is much higher.

Matsuda *et al.* [46] attempted to take into account multiple food items in a single picture. Their approach is a manual effort of multiple stages. First they use different detectors for feature extraction: *Deformable Part Model (DPM)* [17] is a sliding window based image pyramid which uses linear SVM for detecting object. *Circle detector* tries to detect circular objects in the image, mainly plates, bowls, *etc.* *Region Segmentation* is performed using JSEG algorithm [12] which takes the number of regions needed as a parameters and returns the regions. They combine all the candidate regions returned by the aforementioned methods. They apply the work done by Joutou *et al.* [31] on the combined image. They have reported the accuracy of 55.8% and 68.9% on multiple-item food images and single-item food images respectively.

Kawano *et al.* [32] developed a real-time mobile food recognition system using smart phone. Users are asked to take picture of food and draw a bounding box over the food item they want recognized. The image on the bounding box is further segmented using GrubCut. The image-feature-color moment, color histogram, color-autocorrelogram, HoG, PHoG, Bag-of-SURF, Gabor texture feature are extracted in the next phase and the features are then used to train a linear SVM and a fast χ^2 kernel. The same procedure is applied first to extract features from the test images. Then these features are used to classify one of the fifty categories of images using trained linear SVM and fast χ^2 kernel. The authors report 81.55% of classification rate on top 5 food items if the ground truth bounding-box is supplied by the user.

The same authors of the article published another paper [34]. They used same approach as in [32]. They claim to utilize multiple cores available in the device to speed up the process. They have reported an accuracy of 79.2% for top 5 category which is less than reported values on previous paper.

FoodCam [32] was developed by Kawano *et al.* [33] that used convolutional neural network to recognize food items. They used hand-crafted traditional image features with deep convolutional neural network. They took pre-trained weights from ILSVRC and used the UEC-FOOD100 dataset. They report from their experiments that deep CNN methods did not outperform the traditional methods. They used pre-trained ILSVRC weights to extract 4096-dimensional CNN vector. They also extracted features like RootHoG Patches and color patches, and used them to code Spatial Pyramid using the Fisher Vector. They report 72.26% on top-1 accuracy while 92.00% top-5 accuracy on UEC-FOOD100 dataset.

Yanai *et al.* [68] published their results on deep CNN-based experiments. They took pre-trained weights from ILSVRC and modified to produce a 6144-dimensional vector from layer 7. They used these features to compare the results with other traditional methods they had been doing in the past. They report 78.77% and 67.57% for the UEC-FOOD100/256 datasets, respectively.

Chapter 4

Experiments

The RCNN family of algorithms produce good results for object localization and recognition [20]. We chose this framework to train models to recognize food objects in image along with their location in image. The Detectron framework contains the most up-to-date implementation of faster RCNN. We chose faster RCNN for our experiments since Detectron was not available at the time when the experiments were conducted.

4.1 Data

Since our aim was to use the state-of-the-art Faster RCNN framework for food recognition, we needed a dataset that was suitable to use in it. The Faster RCNN framework requires a dataset with images with bounding box information for localization of object in the image, and a label for the object inside the bounding box. Hence, images with just the label(s) of its contents and no bounding box information was not useful for our purposes. Furthermore, we wanted to work exclusively on ‘European’ food, fruit and drinks. These were the major requirements for the training dataset, which constrained our choices and limited the sources we could obtain the data from.

We started with experimenting on a readily available Asian food dataset compiled by Yanai *et al.* [68], which was in the format supported by Faster RCNN. This enabled the us to understand the working of the Faster-RCNN and Caffe, and at the later stages, helped in our work with European food. However, our ultimate goal was to work with the European dataset and develop an application which recognizes European food and beverages. This obviously needed a dataset that has images of food objects.

The standard image datasets like Pascal VOC, ImageNet, *etc.* have images

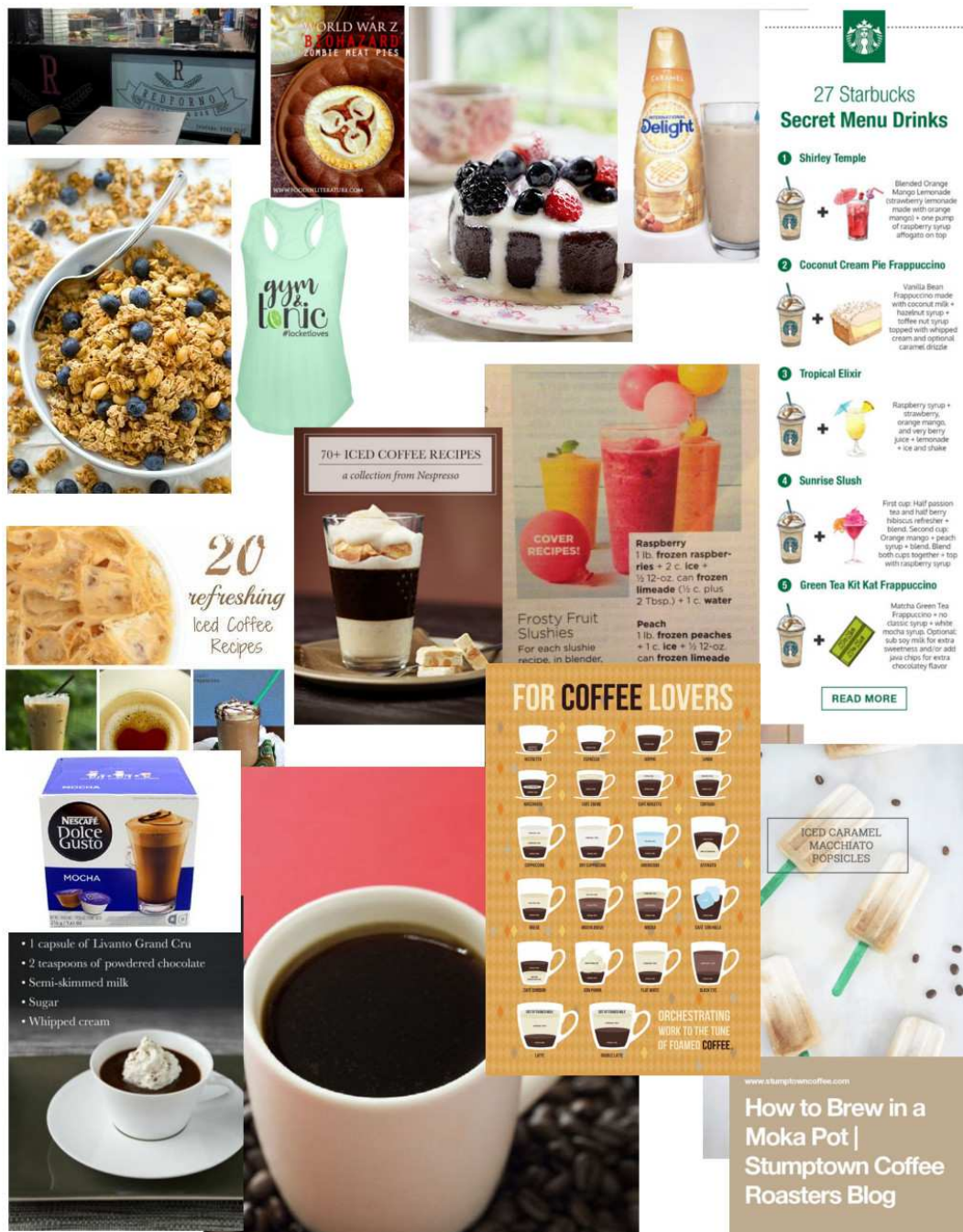


Figure 4.1: Example of noise images we got from Google Image API when we tried to download European food . We were trying get images for “Affogato-Mocha” in this particular example.

with bounding box and class labels, and therefore were suitable for using in the Faster RCNN framework. However, since they contained images of several general categories including, but not limited to the food, fruits and beverages, they were could not be directly used in the framework in their original formats and thus, we searched for a more suitable dataset.

Mezgec *et al.* [47] had worked with European food images in a problem similar to ours that needed a dataset not too different from ours. We discovered that the images they used were downloaded from Google image search. They generously provided us a bash script and instructions on how to download those data.

To use their approach, we first scrapped a few food recipe websites and prepared 612 European food labels. Using these labels, we downloaded images using the Google API. However, after downloading the images, we discovered that the dataset was full of noise. Moreover, there was no consistency in the number of images downloaded per label (food item). For example, we were able to download a total of 26657 images for 612 food labels. This number, which included even the noise images, was too low for us to carry out any useful experiment. At the same time, the number was too high to manually remove the noise and augment some other data set with these images.

We can see as an example the noise images in the food label “Affogato-Mocha” in Figure 4.1. When the search was done for this label, we found 34 noise images out of a total 82 images that were returned. There was a similar trend for the other labels as well. Moreover, these images lacked the information about the location of the food in the images, *i.e.*, there were no bounding boxes. For the framework we were planning to use, Faster RCNN, the bounding box information of the objects in the images is an absolute necessity. Therefore, this approach was discarded.

Finally, since the bounding box information was necessary along with the food labels, we started to deeply explore the ImageNet dataset. It was the largest image dataset which had bounding box information of objects in images as well. The different architecture we discussed in Section 2.3 also had used this, among others, image dataset to for their work. After failing to find a ‘clean’ dataset, we decided to explore this image dataset. We used only the “food” and the “fruit” categories in it to serve our purpose of food object recognition since only those two labels were found relevant to our work.

As stated earlier, ImageNet dataset is one of the largest available image data. It has 1000 categories, where each category is a generic type. We needed to use specific food/fruit items. Food label (or category) in ImageNet are based on synsets. Synsets are used to organize groups, categories and subcategories of images in ImageNet. We looked at synset for “food” and

“fruits” categories and we searched for the images only under these two categories. We found many images with just the food labels (category) and without bounding box information of food object inside the image although they truly belonged to the labelled category. Again, we rejected those images without the bounding box information.

Finally, we ended up with 445 food labels which were subcategories of ‘food’ and ‘fruits’ in ImageNet which had the bounding box information. There were 68 913 such images. These images were returned by Google Image API. The duplicate images were downloaded when we rerun the script. We took these images for our experiment, although this number was much less than we had anticipated for our experiments.

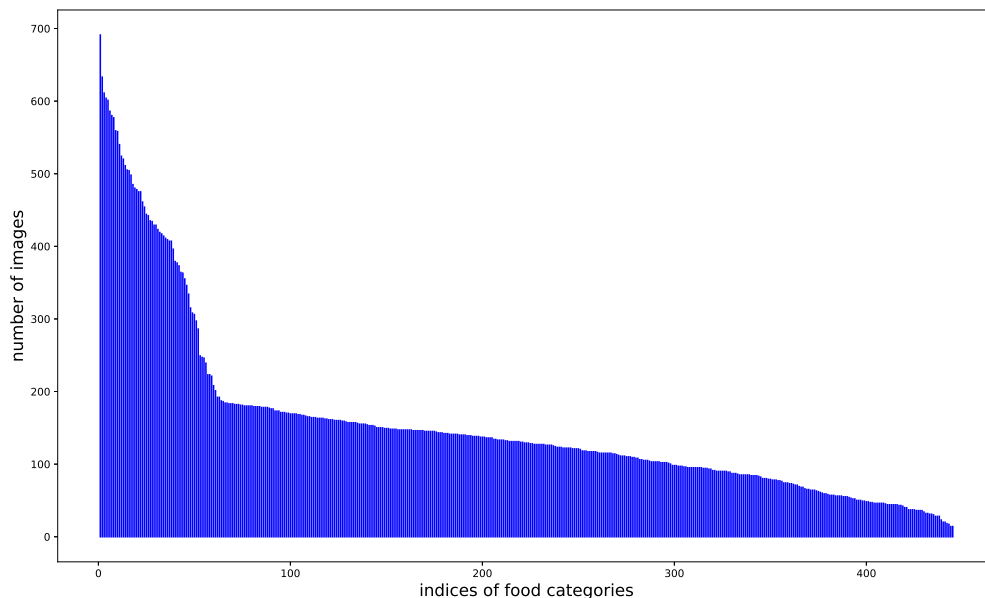


Figure 4.2: Distribution of the food and fruit categories taken from ImageNet dataset. The horizontal axis is the index of categories.

We were able to compile significantly small amount of data of food images considering the total number of categories. Moreover, the number of images fluctuated widely for different categories: it was as low as 14 images and as high as 691 for a single food label. We already discussed in Section 2.2 that deep neural networks require a large number of images in datasets. We decided to train several different Faster R-CNN models and compare the results. Finally, we would choose the best model for our application based

on the performance of the model in unseen data and later use it for our food recognition application.

4.2 Weights

The problem with having too few training data is that the models might easily overfit. In recent years, Transfer Learning has become the *de facto* method for learning features from small datasets that avoids the overfitting problem. Therefore, we decided to use the publicly available trained models for faster RCNN to learn features of our dataset. We used pre-trained weights for Caffe, based on ZF architecture. These weights were suitable for our experiments mainly because of our computational constraints.

4.3 Experimental Setup

We had one CPU box with NVIDIA GPU with 4 GB of memory were available to us for computation. It had Ubuntu Server 16.04 LTS installed on it. We installed standard python Faster RCNN with all the required dependencies on it. We used ZF architecture on Caffe because the requirement for available architectures required the GPUs with at least 8-12 GB of memory. The 10k iterations of training took about 2 hours on this setup. Furthermore same device was used as a server.

Chapter 5

Results

We started our experiment with only five labels to check everything was running properly. Finally, we trained the model in the full 445 data categories following the test phase. However, we observed that the framework completely failed to predict labels from some of the categories. We then realized that we can not use all our data because of very small number of images in some classes. We wanted to find the optimal number of categories for we would achieve the best recognition results. For this purpose, we ranked the labels by the number of training images they had. Twelve sub-datasets were created with different numbers of ranked categories. For example, sub-dataset “top10” contains the dataset for 10 labels with the highest number of training images. These sub-datasets and number of categories each sub-dataset are listed in Table 5.1.

Following this, we ran the experiments, *i.e.*, trained the network for all the sub-datasets separately. We repeated the training with different optimizers. We observed that the optimization process was smooth, *i.e.*, the loss dropped and saturated quickly for all the optimizers. Still, we could not get a good result on test samples. We also observed that the numbers of categories were increased in dataset, the model could not predict even a single result for many categories when the number of categories per sub-dataset was high enough but images per category was low. We call this behaviour *zero prediction* throughout this thesis.

5.1 Loss Graphs

Figures 5.1–5.5 are the loss graphs of the training process for “top90” sub-dataset with various optimizers that Faster RCNN supports. Training processes in Machine Learning are generally explained based on loss graphs,

SN	Sub dataset name	Number of categories
1	top10	10
2	top50	50
3	top90	90
4	top130	130
5	top170	170
6	top210	210
7	top250	250
8	top290	290
9	top330	330
10	top390	390
11	top410	410
12	top445	445

Table 5.1: The division of dataset into sub-datasets. We have created sub-dataset by first sorting the food labels based on numbers of images per category in decreasing order. The sub dataset “top10” contains first 10 categories, the sub-dataset “top 50” contains first 50 categories, and so on.

where the prediction error is plotted against the training iteration. These graphs are expected to start from high values and gradually settle around some minimum value as the training progresses. When the loss function changes very little with iterations, it is an indication the training process may be ended. This is also known as the stopping criteria. Based on loss graphs, we do not see much difference between the optimizers in Faster RCNN.

Figure 5.1 is the bounding box loss graph. In the graph, we can see the decreasing trend of the loss for about 1500 iterations. After that, the graph saturates. Figure 5.2 is loss graph of class-labels. The trend of the graph is decreasing toward iterations 1500, after that the graph is flat like bounding-box loss graph in Figure 5.1. Figure 5.3 is RPN bounding-box loss and Figure 5.4 is RPN class loss. Both of these losses are quite low and flat through out the training process. From this observation, we understand that the RPN network performs quite well. *i.e.*, it can produce good bounding for objects and recognize if there is any object inside that bounding box. However we can see a few spikes on both of these graphs. These spikes are not for the same optimizer. We can see overall loss in Figure 5.5. The trend of Figure 5.1 and Figure 5.2 is clearly visible on this graph. In the graph, we see that optimizers quickly drop the loss value and remain saturated. We recorded the snapshots of the weights after every 10k iterations. We continued the optimization process until 100k iterations. We did not find

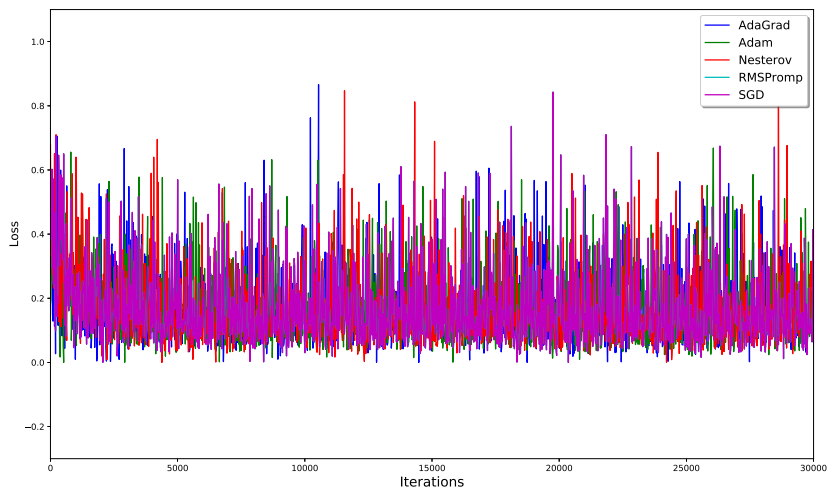


Figure 5.1: Graph of bounding box loss. We can see a downhill trend for a few iteration and the graph quickly settles. The loss values are between 0 and 1.

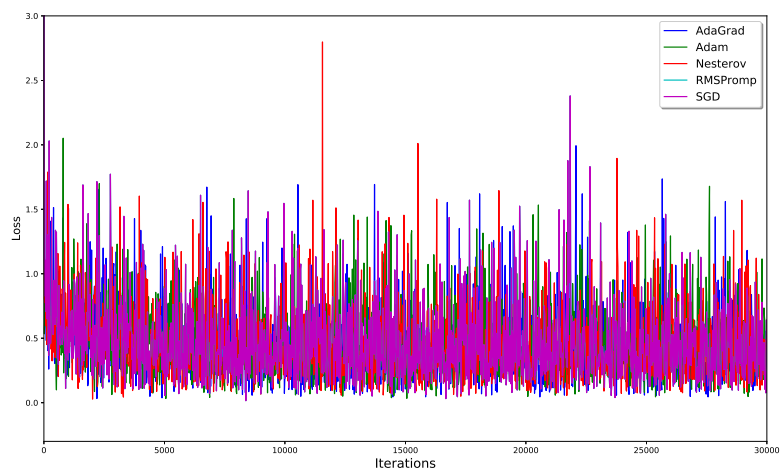


Figure 5.2: Graph of class loss. The graph decreases in first few iterations settles quickly. In comparison to 5.1, the class loss vary quite much.

any performance improvements after 20k iterations. But the performance

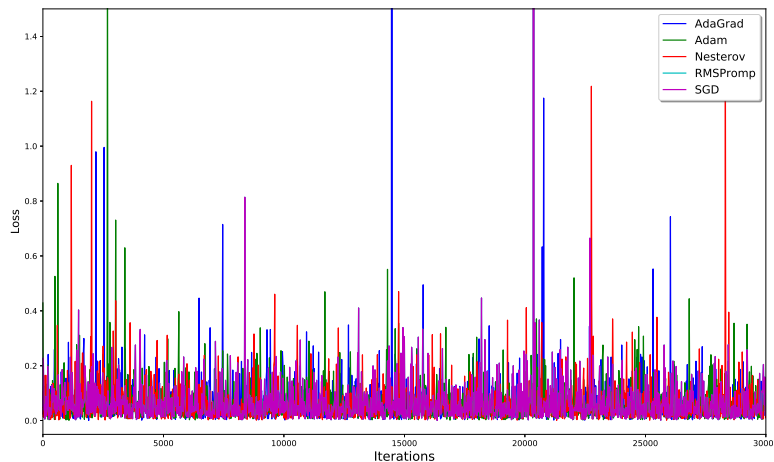


Figure 5.3: Graph of bounding box loss. It looks flat through out the training process with the exception of few spikes.

did not drop either.

There was similar trend on the loss graph of the training processes for all sub-datasets. In Figure 5.6, we have shown loss from ‘sgd’ optimizer only for four subdatasets “top10”, “top50”, “top90” and “top130”. We observed similar trends of optimization in all the sub-datasets. The number of categories per dataset does not seem to matter much; however we can see the average loss increasing as dataset has higher numbers of categories. Similarly the predicting ability of the model drops as the number of categories per dataset grows.

Summary of the results are presented in Figures 5.7 and 5.8. We already mentioned that we did not have uniform numbers of images for each category. From Figure 5.7 we can see that as the number of average training samples (or equivalently minimum number of training sample per category) decreases the number of categories with zero prediction in test set increases. On “top445” dataset, number of zero predictions is almost 400. We can also see that upto “top90” dataset, we do not have any zero predictions.

We have plotted the mean Average Precision for each of 12 sub datasets. It drops as the dataset contains more categories except for “top170”. The mean average precision for “top90” sub-dataset is 0.29.

Since “top90” is our sub-dataset with the largest number of categories which does not give zero-predictions, we chose it as our dataset for final

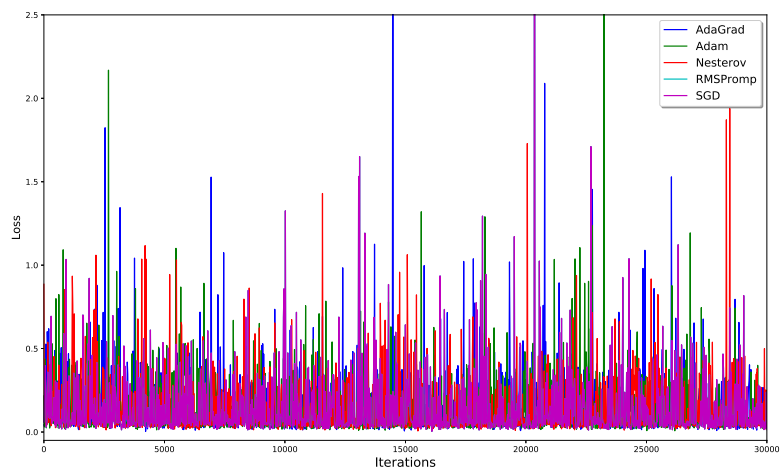


Figure 5.4: Graph of RPN classes loss. This is quite similar to RPN bounding-box loss, some spikes for all optimizers. Minimum value is similar between optimizers.

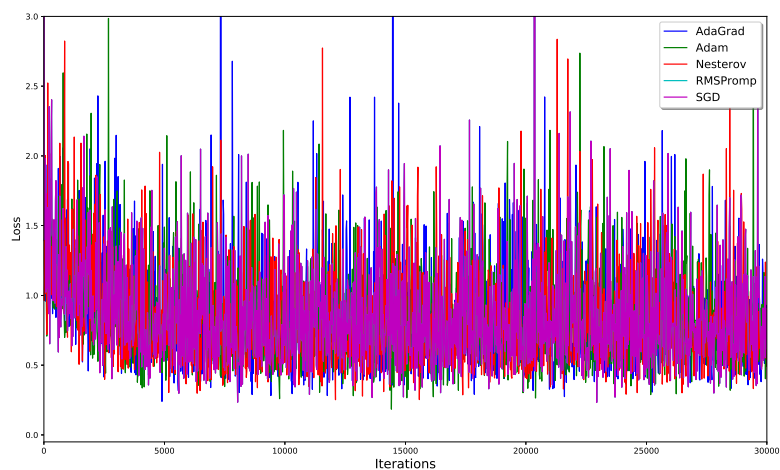


Figure 5.5: Total (combined) graph of all four losses. The optimization trend is dominated by class loss and bounding-box loss.

series of experiments and final model. The “top90” food categories (labels) are listed in Appendix A.

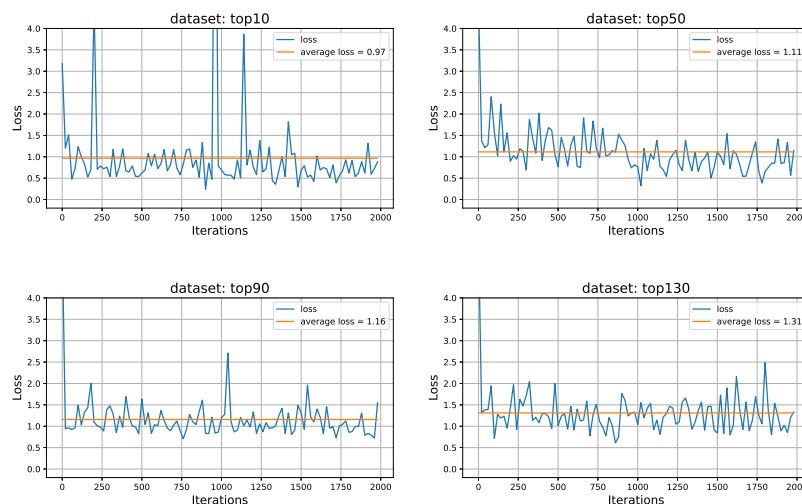


Figure 5.6: Loss graph of the training process for “top10”, “top50”, “top90” and “top130” (clockwise from top-left) sub-datasets. The optimization loss graph has similar trend on all sub-datasets.

We continued our experiment with only 90 categories of food, beverages and fruits, with largest number of image-count being 691 and smallest being 176. Now, we also wanted see the effect of the small numbers of images on the result.

Given the small amount of data, the results reported above was good. We decided to experiment one more time with “top90” sub-dataset. From Figure 4.2 we knew our data distribution was skewed having long tail towards the later food-fruit categories. We were not satisfied with this situation. The lowest number of categories in “top90” dataset was 176. Hence, we took only 176 images from each category on “top90” dataset to make the dataset consistent while sacrificing some images from each category. We trained our Faster RCNN model with this dataset.

In the last series of our experiments, we run training process with consistent “top90” with various optimizers for 100k iterations. Like in previous experiments, the more iterations of training and different optimizers did not produce the better results. However the result did improve slightly from mean average precision from 0.29 to 0.31.

We trained our final networks with all the optimizer that Faster RCNN

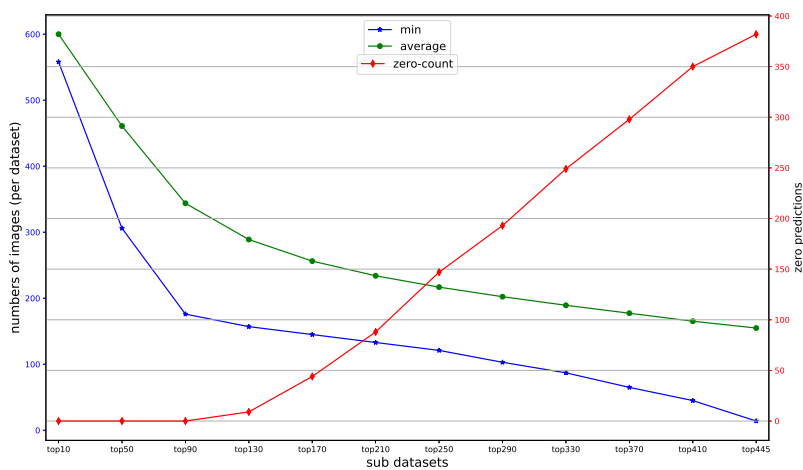


Figure 5.7: Summary of results from 12 sub-datasets with different number of categories on each sub-dataset. When the number of average image count per dataset (green line), and similarly image count on the category with smallest number images per sub-dataset (blue line) goes down, the zero-prediction count (red line) increases.

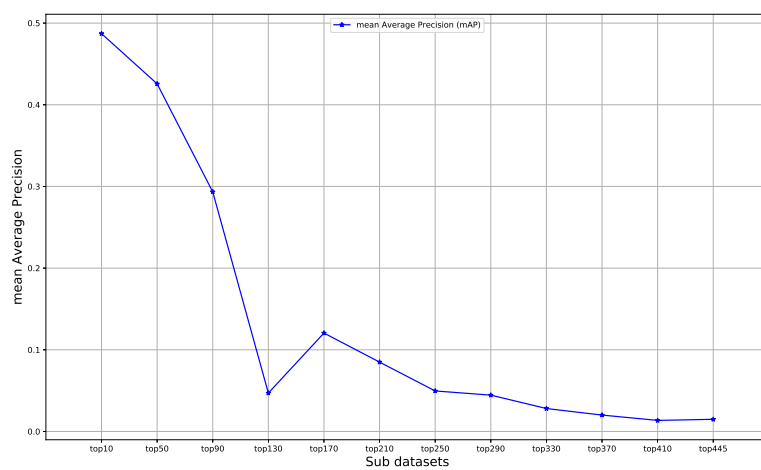


Figure 5.8: Mean AP on test set of dataset with different categories per dataset

iterations	Threshold	mAP
10k	0.1	0.3693
	0.2	0.3661
	0.3	0.3582
	0.4	0.3423
	0.5	0.3139
20k	0.1	0.3701
	0.2	0.3668
	0.3	0.3581
	0.4	0.3416
	0.5	0.3137
30k	0.1	0.3701
	0.2	0.3668
	0.3	0.3581
	0.4	0.3416
	0.5	0.3137

Table 5.2: The mean AP on different iterations and different threshold values on model trained on uniform “top90” dataset using *sgd* optimizer.

supported. One of the parameter while testing the results is *threshold* value for recognition. The recognition layer of the network gives probabilities for each of the categories in the dataset of the test subject being that category. We can then give a threshold value between of [0-1) so that the predicted values looks for at least that value. While testing, we set the threshold 0.1, 0.2, 0.3, 0.4 and 0.5. for three snapshots of weights on 10k, 20k and 30k. The result is tabulated in Table 5.2.

5.2 SmartphoneApplication

The smartphone (Android) application was developed as part for this thesis as mentioned in Chapter 1. The main functionality of developed android application was to provide a user interface which user can use to recognize food objects in a image.

The user takes a photo of the food he/she wants to recognize from smartphone camera and the photo is sent to the server for recognition. The external request to and from the server is handled by NGINX frontend on the server-side. The request from the android application is first received by NGINX and forwarded to Elixir data server, where the image is stored. Elixir also sends the request to recognize the food to Django recognition server. The

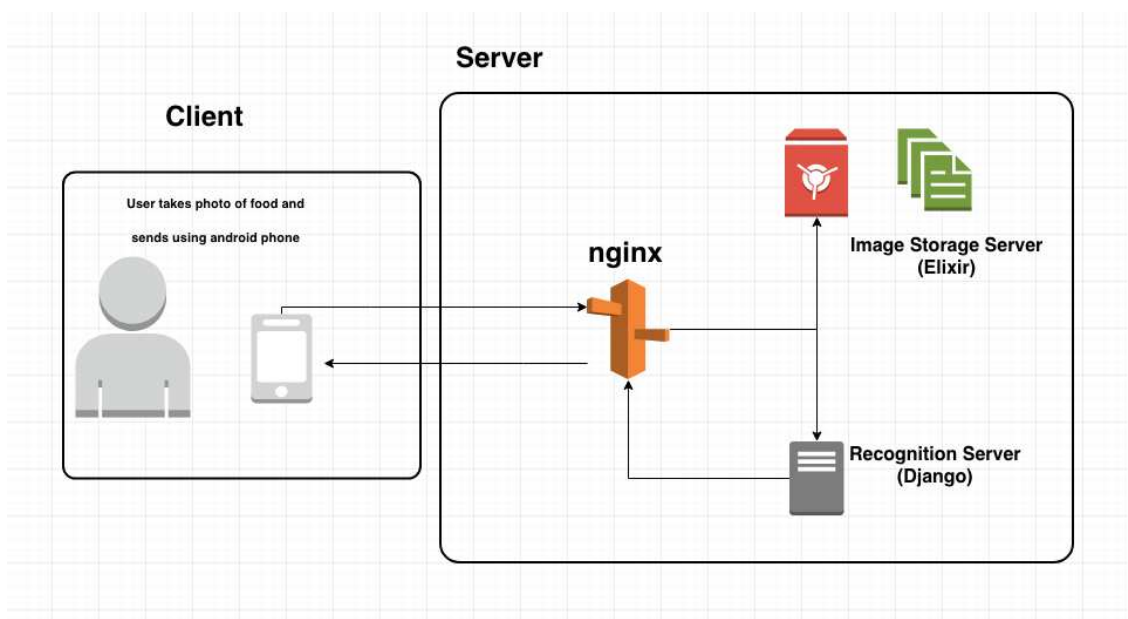


Figure 5.9: Simple Client-Server interaction chart. The chart was made using *draw.io*.

recognition response is then send back to the user as a JSON object via NGINX frontend. The Faster RCNN framework is integrated with Django server. The server uses a model trained for the “top90” dataset (with same number of images for all 90 categories) to recognize the image produced from . On the client-side, the recognized object is cropped and put on view for user based on the JSON response, *i.e.*, food label and bounding box information. Additional information¹ including calories, recipes *etc.*, are also shown in the result screen. A few screenshots of the app is shown from Figure 5.10 and Figure 5.11.

5.3 Discussion

We took food, beverages and fruits image from ImageNet Challenge dataset and used its subcategories as categories for our dataset. We experimented with many subsets of data. But the final set of experiments were run on a dataset with only 176 images per categories. This number was selected to avoid the sampling bias, as that was the number of images in the label with fewest samples. We used pretrained weights from ImageNet dataset and our

¹These information was manually compiled.

main dataset was subset of ImageNet as our dataset. Due to this, we also observed that our network quickly converged after around 10k iterations. We learnt from the previous experiments that we did not need long network training iterations to get the good results. Since the pretrained weights we used were trained on the whole ImageNet dataset, this fast convergence on a subset of the same dataset did not come as a surprise.

To present the result, an android-based mobile application has been developed, which can take picture and get his food recognized.

We did not collect our data, hence, we knew that it would not be an application of practical use unless we have authentic data. However it was not possible to collect the data in our case either. One of the limitations was the thesis was a work of a single person.

Since we were limited by a small dataset, we had to explore and understand the effect of small numbers of training samples on Faster RCNN framework, or even find optimized model that perform well with small number of data set.

We observed the training losses for five different optimizers for each of these 12 datasets. We witnessed the trend of decreasing loss upto 1500 iterations specially on class loss and bounding-box loss. On the contrary, the RPN network loss was small and flat throughout the training process. Overall loss was dominated by class and bounding-box loss.

We observed the effect of training samples in datasets on quality of recognition/prediction. We also investigated the effect of shorter vs longer training times and its effect on the result. We used optimizers like stochastic gradient descent, adam, nesterov, RMSprop, *etc*. In our experiment we found out that Faster RCNN is immune to the selection of the optimizer. The prediction quality of Faster RCNN is negatively correlated the numbers of categories (labels) in the dataset. We learnt that it was necessary to have balanced number of training samples in the dataset for the models to perform better on unseen data.

We do not get zero prediction for any category for datasets “top10”, “top50” and “top90”. We observed that we had at least 176 images per category in “top90” dataset. So we randomly chose only 176 images per category even if they had higher number of images and trained the Faster RCNN model on this newly created dataset. We got better mAP values compared to previous model.

We used this model for our application.

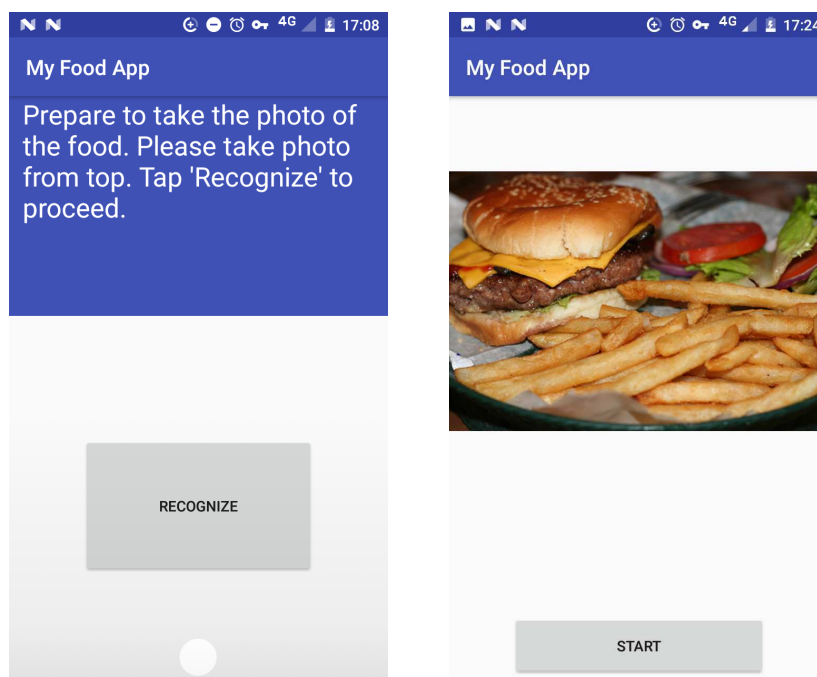


Figure 5.10: The figure on the left is the welcome screen of the app. Instruction for the user is shown in the first page of application. The right screenshot is second screen of the application. User either takes a photo from camera or selects an existing food image from phone and presses *start* button.

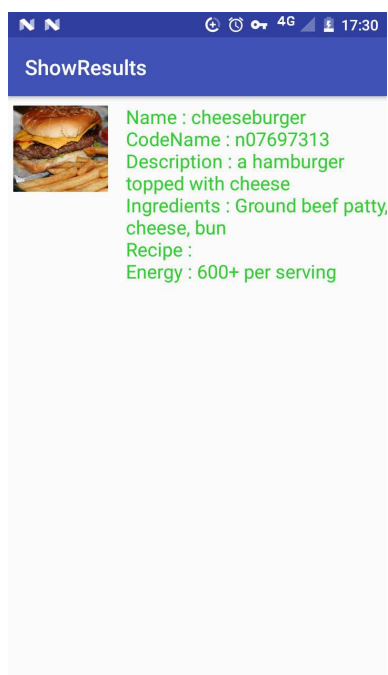


Figure 5.11: Finally the user is shown the recognition results.

Chapter 6

Conclusions

The main objective of this thesis was to develop a deep neural network model that could localize food objects in image and identify those food objects. For this purpose, a model was developed using Faster RCNN and was deployed in our server. Our initial plan was to utilize European food dataset. Unfortunately, it was not accessible for our research purpose. Instead, we trained our model with a subset of ImageNet dataset containing about 69k images of food, beverages and fruits. This subset dataset contained images of 445 food, beverages and fruits. The dataset itself did not have same numbers images for per category and was skewed. We still continued our experiments on this dataset.

We started the thesis by looking into the ongoing research in Machine Learning, Deep Learning and/or Computer Vision, specially related to object recognition. We found in the literature that the techniques based on CNN were promising due its performance and accuracy in computer vision related tasks. We started by investigating the relationship between the brain and the neural networks. First we looked into the building blocks of neural networks and deep networks. Then we looked into the classical techniques of neural networks; *e.g.* activation function, back propagation, loss functions and different optimizers. Finally we discussed deep networks and convolutional networks. We also briefly discussed current start of art CNN architecture.

We used the Faster RCNN framework for our experiments. As stated, we wanted to find the localization information on images first and then recognize the object. Faster RCNN was developed to do exactly this operation. Since we were limited by to small dataset, we had to explore and understand the effect of small numbers of data on this framework. We first made a dataset consisting of 445 categories of food/fruits with different number of labels. They were made 12 sub-datasets out of it and we experimented on those sub-datasets.

We experimented with the different optimizers and then observed training loss for each one of them. We also investigated the effect of shorter vs longer training times and its effect on the result. In our experiment, we found out that the Faster RCNN is immune to choice of optimizer supported by the framework. However the recognition result is negatively correlated to the numbers of categories.

Moreover we found that with “top90” dataset the we do not get zero prediction for any category. We also observed that we had at least 176 images per category in top 90 dataset. So we selected exactly 176 images per category even if they had higher number of images and trained the faster RCNN model on this dataset. We got better mAP values compared to previous model. We also observed that even with relatively small training sample size, the Transfer learning performs quite well in circumstances such as ours.

In addition to food detection and localization algorithm, for the client side, we developed an Android application which acts as interface for user who can take picture of a food tray and send it to the server. The server returns the result in JSON format which includes the information about the identified food object and its location in the image. Our application works and we can recognize images from the dataset with some exceptions. We can make this better by training the model with larger food-image datasets. All in all, our method works with those categories of food for which we have enough samples. This way we partially achieved our initial goal. After this work, we are more convinced that research on automatic food recognition and localization can improve better calorie-estimation.

We had to compromise quality of experiments and results because of data. We strongly like to suggest any future researchers working on similar problems to use tools like Mechanical Turk to collect data and/or label data. A proper data to work on paves way for towards the goal more clearly.

For future work, we can also develop applications that can ‘invent’ new kind of food based on the food images and ingredients. That is, it may be possible to develop an AI chef that augments the traditional recipe with new machine-generated ones. Moreover, such application may also be able to predict replacement ingredients in recipes according to the tastes and dietary restrictions of the users.

Bibliography

- [1] ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., IRVING, G., ISARD, M., ET AL. Tensorflow: A system for large-scale machine learning. In *OSDI* (2016), vol. 16, pp. 265–283.
- [2] BASTIEN, F., LAMBLIN, P., PASCANU, R., BERGSTRA, J., GOODFELLOW, I., BERGERON, A., BOUCHARD, N., WARDE-FARLEY, D., AND BENGIO, Y. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590* (2012).
- [3] BAY, H., ESS, A., TUYTELAARS, T., AND VAN GOOL, L. Speeded-up robust features (surf). *Computer vision and image understanding* 110, 3 (2008), 346–359.
- [4] BENGIO, Y. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning* (2012), pp. 17–36.
- [5] BISHOP, C. M. *Neural networks for pattern recognition*. Oxford university press, 1995.
- [6] BROSANAN, T., AND SUN, D.-W. Improving quality inspection of food products by computer vision—a review. *Journal of food engineering* 61, 1 (2004), 3–16.
- [7] BRYNJOLFSSON, E., AND MCAFEE, A. The big data boom is the innovation story of our time. *The Atlantic* 21 (2011).
- [8] CHEN, M., DHINGRA, K., WU, W., YANG, L., SUKTHANKAR, R., AND YANG, J. Pfid: Pittsburgh fast-food image dataset. In *Image Processing (ICIP), 2009 16th IEEE International Conference on* (2009), IEEE, pp. 289–292.

- [9] CHOLLET, F., ET AL. Keras: Deep learning library for theano and tensorflow. *URL: <https://keras.io/>* 7 (2015), 8.
- [10] CISCO. Cisco visual networking index: Forecast and methodology, 2016 to 2021 (white paper), 2017.
- [11] DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K., AND FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 248–255.
- [12] DENG, Y., AND MANJUNATH, B. Unsupervised segmentation of color-texture regions in images and video. *IEEE transactions on pattern analysis and machine intelligence* 23, 8 (2001), 800–810.
- [13] DESELAERS, T., PIMENIDIS, L., AND NEY, H. Bag-of-visual-words models for adult image classification and filtering. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on* (2008), IEEE, pp. 1–4.
- [14] DYER, C. Notes on adagrad. *School of Computer Science, Carnegie Mellon University 5000* (2013).
- [15] ERICKSON, B. J., KORFIATIS, P., AKKUS, Z., KLINE, T., AND PHILBRICK, K. Toolkits and libraries for deep learning. *Journal of digital imaging* 30, 4 (2017), 400–405.
- [16] EVERINGHAM, M., VAN GOOL, L., WILLIAMS, C. K., WINN, J., AND ZISSERMAN, A. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88, 2 (2010), 303–338.
- [17] FELZENSZWALB, P. F., GIRSHICK, R. B., MCALLESTER, D., AND RAMANAN, D. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence* 32, 9 (2010), 1627–1645.
- [18] FRIEDMAN, J., HASTIE, T., AND TIBSHIRANI, R. *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.
- [19] GIRSHICK, R. Fast R-CNN. *arXiv preprint arXiv:1504.08083* (2015).
- [20] GIRSHICK, R., DONAHUE, J., DARRELL, T., AND MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 580–587.

- [21] GOODFELLOW, I., BENGIO, Y., COURVILLE, A., AND BENGIO, Y. *Deep learning*, vol. 1. MIT press Cambridge, 2016.
- [22] HE, K., ZHANG, X., REN, S., AND SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1026–1034.
- [23] HINTON, G., SRIVASTAVA, N., AND SWERSKY, K. Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural networks for machine learning, Coursera lecture 6e* (2012).
- [24] HOASHI, H., JOUTOU, T., AND YANAI, K. Image recognition of 85 food categories by feature fusion. In *Multimedia (ISM), 2010 IEEE International Symposium on* (2010), IEEE, pp. 296–301.
- [25] HORNIK, K., STINCHCOMBE, M., AND WHITE, H. Multilayer feed-forward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.
- [26] HUBEL, D. H., AND WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. *The Journal of physiology* 160, 1 (1962), 106–154.
- [27] HUBEL, D. H., AND WIESEL, T. N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology* 195, 1 (1968), 215–243.
- [28] IOFFE, S., AND SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (2015), pp. 448–456.
- [29] JIA, Y., AND SHELHAMER, E. Caffe model zoo, 2015.
- [30] JIA, Y., SHELHAMER, E., DONAHUE, J., KARAYEV, S., LONG, J., GIRSHICK, R., GUADARRAMA, S., AND DARRELL, T. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093* (2014).
- [31] JOUTOU, T., AND YANAI, K. A food image recognition system with multiple kernel learning. In *Image Processing (ICIP), 2009 16th IEEE International Conference on* (2009), IEEE, pp. 285–288.

- [32] KAWANO, Y., AND YANAI, K. Real-time mobile food recognition system. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on* (2013), IEEE, pp. 1–7.
- [33] KAWANO, Y., AND YANAI, K. Food image recognition with deep convolutional features. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication* (2014), ACM, pp. 589–593.
- [34] KAWANO, Y., AND YANAI, K. Foodcam: A real-time food recognition system on a smartphone. *Multimedia Tools and Applications* 74, 14 (2015), 5263–5287.
- [35] KE, Y., AND SUKTHANKAR, R. Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* (2004), vol. 2, IEEE, pp. II–II.
- [36] KHOSO, M. How much data is produced every day? <http://www.northeastern.edu/levelblog/2016/05/13/how-much-data-produced-every-day>, 2016. [Online; accessed 01.06.2018].
- [37] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [38] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.
- [39] KUMAR, N., BELHUMEUR, P. N., BISWAS, A., JACOBS, D. W., KRESS, W. J., LOPEZ, I. C., AND SOARES, J. V. Leafsnap: A computer vision system for automatic plant species identification. In *Computer vision–ECCV 2012*. Springer, 2012, pp. 502–516.
- [40] LECUN, Y., BENGIO, Y., ET AL. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361, 10 (1995), 1995.
- [41] LECUN, Y., BENGIO, Y., AND HINTON, G. Deep learning. *nature* 521, 7553 (2015), 436.

- [42] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324.
- [43] LI, F.-F., AND KARPATY, A. Convolutional neural networks for visual recognition, 2015.
- [44] LOWE, D. G. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on* (1999), vol. 2, Ieee, pp. 1150–1157.
- [45] LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60, 2 (2004), 91–110.
- [46] MATSUDA, Y., HOASHI, H., AND YANAI, K. Recognition of multiple-food images by detecting candidate regions. In *Multimedia and Expo (ICME), 2012 IEEE International Conference on* (2012), IEEE, pp. 25–30.
- [47] MEZGEC, S., AND KOROUŠIĆ SELJAK, B. Nutrinet: A deep learning food and drink image recognition system for dietary assessment. *Nutrients* 9, 7 (2017), 657.
- [48] MINSKY, M., AND PAPERT, S. A. Artificial intelligence progress report.
- [49] NESTEROV, Y. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. In *Soviet Mathematics Doklady* (1983), vol. 27, pp. 372–376.
- [50] PAN, S. J., AND YANG, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.
- [51] PARK, J., AND SANDBERG, I. W. Universal approximation using radial-basis-function networks. *Neural computation* 3, 2 (1991), 246–257.
- [52] REN, S., HE, K., GIRSHICK, R., AND SUN, J. Faster rcnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (2015), pp. 91–99.
- [53] RUBLEE, E., RABAUD, V., KONOLIGE, K., AND BRADSKI, G. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on* (2011), IEEE, pp. 2564–2571.

- [54] RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATY, A., KHOSLA, A., BERNSTEIN, M., ET AL. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.
- [55] SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural networks* 61 (2015), 85–117.
- [56] SHIN, H.-C., ROTH, H. R., GAO, M., LU, L., XU, Z., NOGUES, I., YAO, J., MOLLURA, D., AND SUMMERS, R. M. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging* 35, 5 (2016), 1285–1298.
- [57] SHOTTON, J., JOHNSON, M., AND CIPOLLA, R. Semantic texton forests for image categorization and segmentation. In *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on* (2008), IEEE, pp. 1–8.
- [58] SHROFF, G., SMAILAGIC, A., AND SIEWIOREK, D. P. Wearable context-aware food recognition for calorie monitoring. In *Wearable Computers, 2008. ISWC 2008. 12th IEEE International Symposium on* (2008), IEEE, pp. 119–120.
- [59] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- [60] SONNENBURG, S., RÄTSCH, G., SCHÄFER, C., AND SCHÖLKOPF, B. Large scale multiple kernel learning. *Journal of Machine Learning Research* 7, Jul (2006), 1531–1565.
- [61] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
- [62] SURYANI, D., DOETSCH, P., AND NEY, H. On the benefits of convolutional neural network combinations in offline handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on* (2016), IEEE, pp. 193–198.

- [63] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCHE, V., RABINOVICH, A., ET AL. Going deeper with convolutions. *Cvpr*.
- [64] UIJLINGS, J. R., VAN DE SANDE, K. E., GEVERS, T., AND SMEULDERS, A. W. Selective search for object recognition. *International journal of computer vision* 104, 2 (2013), 154–171.
- [65] UN. Tenfold increase in childhood and adolescent obesity in four decades: new study by imperial college london and who. <http://www.who.int/en/news-room/detail/11-10-2017-tenfold-increase-in-childhood-and-adolescent-obesity-in-four-decades-r> 2017. [Online; accessed 01.06.2018].
- [66] VAN HOOSER, S. D., HEIMEL, J. A. F., CHUNG, S., NELSON, S. B., AND TOTH, L. J. Orientation selectivity without orientation maps in visual cortex of a highly visual mammal. *Journal of Neuroscience* 25, 1 (2005), 19–28.
- [67] WIDROW, B. Adaline and madaline-1963. In *Proc. IEEE 1st Int. Conf. on Neural Networks* (1987), vol. 1, pp. 143–57.
- [68] YANAI, K., AND KAWANO, Y. Food image recognition using deep convolutional network with pre-training and fine-tuning. In *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on* (2015), IEEE, pp. 1–6.
- [69] YANG, J., JIANG, Y.-G., HAUPTMANN, A. G., AND NGO, C.-W. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval* (2007), ACM, pp. 197–206.
- [70] YANG, S., CHEN, M., POMERLEAU, D., AND SUKTHANKAR, R. Food recognition using statistics of pairwise local features. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 2249–2256.
- [71] YOSINSKI, J., CLUNE, J., BENGIO, Y., AND LIPSON, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems* (2014), pp. 3320–3328.
- [72] ZEILER, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).

- [73] ZEILER, M. D., AND FERGUS, R. Visualizing and understanding convolutional networks. In *European conference on computer vision* (2014), Springer, pp. 818–833.
- [74] ZHENG, L., YANG, Y., AND TIAN, Q. Sift meets cnn: A decade survey of instance retrieval. *IEEE transactions on pattern analysis and machine intelligence* (2017).

Appendix A

List of ‘top90’ food categories

SN	Synset	Image Count	Real Name
1	n07584110	691	consomme
2	n07836838	633	chocolate sauce OR chocolate syrup
3	n07874780	611	porridge
4	n07579787	604	plate
5	n07920052	601	espresso
6	n07615774	586	ice lolly OR lolly OR lollipop OR popsicle
7	n07565083	580	menu
8	n07873807	577	pizza OR pizza pie
9	n07875152	559	potpie
10	n07711569	558	mashed potato
11	n07583066	540	guacamole
12	n07590611	524	hot pot OR hotpot
13	n07614500	520	ice cream OR icecream
14	n07613480	511	trifle
15	n07742313	505	Granny Smith
16	n07768694	504	pomegranate
17	n07871810	498	meat loaf OR meatloaf
18	n07697537	485	hotdog OR hot dog OR red hot
19	n07892512	480	red wine
20	n07880968	478	burrito
21	n13133613	475	ear OR spike OR capitulum
22	n07747607	475	orange
23	n07697313	461	cheeseburger
24	n07930864	454	cup
25	n07932039	444	eggnog
26	n07829248	442	cider vinegar

SN	Synset	Image Count	Real Name
27	n12267677	435	acorn
28	n07860988	434	dough
29	n07749582	429	lemon
30	n07831146	429	carbonara
31	n07693725	423	bagel OR beigel
32	n12620546	419	hip OR rose hip OR rosehip
33	n07802026	417	hay
34	n07753275	414	pineapple OR ananas
35	n07760859	411	custard apple
36	n07739125	409	apple
37	n11879895	407	rapeseed
38	n07684084	407	French loaf
39	n07754684	396	jackfruit OR jak OR jack
40	n07695742	379	pretzel
41	n07583197	377	soup
42	n12144580	373	corn
43	n07753592	364	banana
44	n07745940	363	strawberry
45	n12768682	355	buckeye OR horse chestnut OR conker
46	n07753113	346	fig
47	n07749969	334	grapefruit
48	n07874995	315	oatmeal OR burgoo
49	n07712959	308	nacho
50	n07858978	306	honey
51	n07893642	297	champagne OR bubbly
52	n07935288	286	hyson
53	n07576781	249	barbecue OR barbeque
54	n07769584	247	quince
55	n07612632	246	pudding
56	n07642933	239	jam
57	n07762913	223	durian
58	n07772935	223	coconut OR cocoanut
59	n07769731	221	rambutan OR rambotan
60	n07557434	208	dish
61	n07765361	201	guava
62	n07920349	192	cappuccino OR cappuccino coffee OR coffee cappuccino
63	n07631926	192	ice-cream cake OR icebox cake
64	n07764155	187	mango
65	n07585557	186	chicken soup
66	n07863374	184	pasta
67	n07807922	184	fruit salad
68	n07593004	183	pot-au-feu

SN	Synset	Image Count	Real Name
69	n07808904	183	tabbouleh OR tabooli
70	n07772274	183	chestnut
71	n07934373	182	sun tea
72	n07690431	182	bran muffin
73	n07746186	182	persimmon
74	n07872593	181	moussaka
75	n07704205	181	congee OR jook
76	n07700003	180	spaghetti
77	n07933799	180	cuppa OR cupper
78	n07621497	180	zabaglione OR sabayon
79	n07808479	180	chicken salad
80	n07587023	180	potage OR pottage
81	n07924834	179	orange juice
82	n07615460	179	strawberry ice cream
83	n07935043	179	souchong OR soochong
84	n15092650	179	biotin OR vitamin H
85	n07861813	178	chicken and rice
86	n07611267	178	flan
87	n07585758	178	gazpacho
88	n07806221	178	salad
89	n07926442	177	orangeade
90	n07864934	176	chili OR chili con carne