Computer Science 4490Z Research Project Thesis: Enhancing Food Image Recognition for Nutrition Analysis

Hanxiang Pan
Student Number: 250608428
Department of Computer Science
Western University
London, Ontario, Canada

Email: hpan33@uwo.ca
March 30, 2016

The supervisors for this thesis are Dr. Charles Ling, and Dr. Michael Katchabaw

Abstract

The task of reading images taken by users and classifying into different categories is used widely in many domains. However, the challenge lies in the varying quality of images and how to best identify what is a food item and what is not. This research project takes a practical approach by using the best-practices in academia and industry as a main direction. I started with simple models such as linear regression and moved on to more complex models such as the Convolutional Neural Network (CNN) and obtained the results through a trial-and-error approach. The final classifier is a combination of GoogLeNet's CNN architecture for extracting features from images and training/classifying them using Support Vector Machine, achieving top-1 error rate of 1.93% for 10-class classification. This discovery lays a great foundation for nutrition analysis based on the classified food item. However, there is still room for improvement given more data for training and cross-validation, and a custom CNN trained from our own data.

Acknowledgement

I would like to thank my supervisor Charles Ling for giving me this opportunity to gain hands-on experience with computer vision, an application of machine learning, specifically image classification. Also, I would like to appreciate the help that Tracy Wang provided throughout the whole research project from giving directions on my next steps to providing consultation on specific problems I encountered along the way. I also want to mention Andrew Ng's machine learning course that laid a solid foundation for understanding how different machine learning models work from a mathematical and intuitive perspective.

Table of Contents

List of Figures

List of Tables

List of Equations

List of Abbreviations, Symbol, Nomenclature

- **One-vs-all:** For multiclass classification, selecting among N different classes is hard to do, but it would be easier if we convert the problem in N different binary problems. This way, each problem requires the classifier to distinguish between being in the given class or being in any of the other (N-1) classes (Aly, 2005).
- **CNN:** Convolutional Neural Network, an application of neural network used widely in computer vision.
- **Convolutional Layer:** In CNN, this kind of layer is used a lot. It basically transforms the previous layer to a new size by performing matrix multiplication. The convolutional layer consists of a set of small learnable filter that extends through the full depth of the image. To perform convolution, the filters are slid across the image with a given step size, and as we slide across, dot products between the input and the filter is calculated.
- **Pooling Layer:** In CNN, the pooling layer transforms the previous layer to a new size, usually reducing the amount of parameters in the previous layer to save computations in the network. This layer is usually inserted between convolutional layers. MAX is one of the most used operations in this layer, discarding a portion of the previous input.
- **Fully-connected layer:** Every neural on the previous layer connects to every neural on the next layer. It is usually used near the end of the architecture since it is computationally expensive.
- **Principal Component Analysis:** An efficient way to reduce multidimensional data into lower dimensions to save computational power and reveal more insights.
- **Supervised and unsupervised learning:** Supervised learning requires the user to specify the input features and output results, usually it involves a predefined function. Unsupervised learning means that the user does not know the output beforehand, and the learning model is able to unveil hidden structure in the dataset.
- **Underfit:** Due to insufficient data, the model that is used does not fit the training data, thus will not perform well with classifying the test data.
- **Overfit:** When the input data contains noise, and the model tries to fit every data point in the training data, as a result, the model will not be able to generalize and fit the test data well.
- **C (cost) in SVM:** Controls the tradeoff between how much error we are willing to accept to gain more stability of the model. It is also known as the cost of the soft margin cost function. A large C means the model penalizes misclassification a lot.
- **γ (gamma) in SVM:** With a large gamma, the model will try to fit most of the training data well, leading to overfitting. With a small gamma, the model might not fit the training data well, leading to underfitting.

1 INTRODUCTION

Nowadays people have different eating habits and people, especially diabetes patients, are increasingly cautious about their diet. Current research in the department of computer science led by Dr.Charles Ling's research spin-off project GlucoGuide allows diabetes patients to keep track of their food consumption throughout the day to measure their blood sugar level. Traditional ways of recording one's food consumption is tedious and might need to go through dozens of hierarchical menus before adding in the right food. GlucoGuide offers users to take pictures of their food for instant nutrition identification and eating habit analysis. The biggest challenge for food image recognition is the different environments in which the images can be captured, as well as the variations of the same food category prepared by different people. These factors greatly increase the difficulty for accurate food recognition and analysis.

In this research project, I have investigated the different ways of extracting useful features from food images and compared different models that gave the best accuracy for classifying a variety of food items. In addition, I have conducted experiments on different food class datasets, ranging from apple to sashimi with 10 classes in total. Despite the limited hardware resources and limited time, I was able to achieve a 1.97% top-1 error rate with 10-class classification.

1.1 Personal Learning

My motivation for this research project was to gain a deeper understanding in machine learning and to experience different training models both from an implementation perspective and from a practical perspective. With limited background knowledge in food recognition and computer vision, I started by reading research papers gain a high-level understanding of different aspects in food recognition, to better direct my future research efforts. Upon identifying factors that contribute to the identification of food items (Zhang, Yu, Siqqiquie, Divakaran & Sawhney, 2015), I had clearer direction of where to start. I started by learning more about the different classifier models, how the input data is interpreted by the different models, and what kind of techniques I should apply to increase the overall accuracy.

Technically, I had little knowledge of the tools in machine learning and computer vision coming into the research project. I boosted my knowledge in the field by following the Machine learning course on Coursera (Ng, n.d.), which laid a great foundation for learning complex models later in the research process. To gain practical experience, I completed all the assignments for the Machine Learning course, to try out many of the most prominent learning models on both hypothetical and real-world data. At the same time, it was a great opportunity to practice Matlab through those assignments. Although there are many libraries out there for training and classifying images, I wanted to know how the models work technically, so as to have more flexibility in choosing a combination of models or customizing them to fit my research. Several research papers and courses (Zhang et al., 2015; CS231n: Convolutional Neural Networks for Visual Recognition, 2016; Zeiler, M.D. & Fergus, 2014) guided me along the way in studying these architectures of neural networks and investigating in different approaches to boost accuracy in food recognition.

My approach was to start with simple models to match my own learning curve of machine learning. Also, this approach enabled myself to see how the accuracy of image recognition is improved overtime with different models used. Having this trial and error approach provided a good guidance for my next steps.

1.2 Data Source

The food image files were collected from the internet, with a total of 11,129 images and organized into respective food category folders. The quality of the images varies a lot. Some images contain the food image itself with a blank background, some had cluttered background, some were deformed by shape due to zooming, and some had very poor lighting. One motivation for having variations within the image set is to prevent the model from underfitting on these similar image quality, since it would be very likely that users of the classifier will be taking pictures under different circumstances, most of the images will also vary in quality. In a later section, we will investigate how the different qualities of images contribute to the accuracy of image classification which could be found in the discussion in Section 3.3 Error Analysis.

1.3 Input Selection

For simplicity and efficiency reasons, initially, I started with only two classes - Apple and Burger - since I believed that these two classes vary significantly in terms of their characteristics, which may be easier to achieve a descent accuracy with simple models. Gradually, I moved on to 3 classes, 5 classes and 10 classes as the classification accuracy increased with different models used. The maximum number of classes that's investigated in this research project is 10. This is due to the limited computing power, and to ensure a large number of input images per class to ensure variety.

2 FEATURE EXTRACTION

2.1 Pixel

The most naïve way of choosing features from images is by their raw pixels. Indeed, it did not give much info on the characteristics of a particular food. Having varying sizes and resolutions in input images required uniform resizing for pixel feature extraction. However, reducing the image size may cause images to lose important detail that might be crucial for classification. The results of using pixels as the features can be seen in Section 5.1.1 Pixel Features Evaluation.

2.2 Edge Detection

Instead of using raw pixels as features for images, I thought that edges would be more representative of what the object is. My hypothesis was that objects in the same class tend to have

similar edges. As for implementation, I have applied three most widely used edge detection methods (Sobel, Canny, Prewitt) as feature extractors over all the images in each class. Different edge detectors produced different edges which can be viewed in Figure 1. The output of the edge function is logical 1's and 0's, therefore, this feature extraction method does not provide lots of detail of the image for further processing. Each edge detection method has advantages and disadvantage over the other. The results of using edge detection as the feature extractor is shown in Section 5.1.2 Edge Detectors Evaluation.

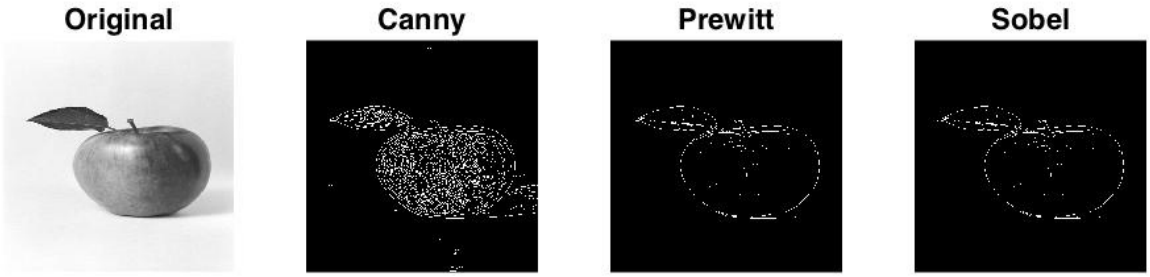

Figure 1 - Edge Detectors Comparison

2.3 SURF

Compared to edge detection, Speeded Up Robust Features (SURF) performs better in terms of efficiency and provides better feature accuracy. This method uses interest points that represents repeatable points under different viewing conditions. After interest points have been generated, a feature vector can be formed by neighboring interest points. SURF ensures that the descriptors are robust to handle varying deformations to the image. For implementation, a SURF extraction is performed on all images to create the feature vector for each image. In the process, I noticed that some images tend to have less features extracted than others. If we decide to include them, it may introduce inaccuracies into the training and classification process. These images tend have a very low resolution. Statistically, I have decided to leave out the bottom 5% images that have the least features. The results of how SURF features actually perform in representing food items can be seen in Section 5.1.3 SURF Features Evaluation.

2.4 SURF with Bag of Features

Traditional SURF gave different features for every image. But how can we visualize or combine features extracted from images in the same class to form a more generic feature for the entire class.

I decided to combine SURF with the bag-of-features approach. This is a two-step process. The first step is to use SURF as the feature extractor, similar to the previous section. The second step is to use the bag-of-features approach to group relevant features together to eliminate useless features and to save computational memory. One important characteristic of bag-of-features is that it does not retain the order or spatial position of the features in the image. This can be useful if the image is deformed or taken under different angles, as long as the food item itself contain the characteristics/features. In terms of implementation, we run SURF to generate interest points from the images, then we use k-means clustering to define 500 cluster points randomly. By gathering the interest points (extracted features) by their Euclidean distance closest towards any one of the 500 cluster points, we can form large clusters around the cluster points. After running k-means clustering over numerous iterations, the cluster boundaries will converge, and the resulting features will be formed. The results of combining bag of features with SURF for classification is shown in Section 5.1.3 SURF Features Evaluation.

2.5 Convolutional Neural Network as Feature Extractor

My next approach in feature extraction was through the Convolutional Neural Network (CNN). Compared to the previous approaches in feature extraction, this approach is unsupervised that it allows the CNN to pick out different types of features by applying different convolutions to the original image and intermediate layers. Whereas in the previous approaches, we tend to have an algorithm that extracts features based on a predefined routine. For example, some convolution filters may be sensitive to vertical edges, some filters may be sensitive to green colors and so on. With varying input images, one specific features could be learned at different spatial locations within any given image. The resulting features would go through numerous combinations of the following layers to get to the final features. In my solution, I used GoogLeNet, the ImageNet ILSVRC competition winning CNN, due to its highly efficient architecture, its well known accuracy, and with limited computing power, a pre-trained CNN was more favorable than training one myself. The input images were passed through GoogLeNet and the features were extracted from the second last layer right before going the fully-connected layer. In total, 1024 features were extracted from each food image, which were constructed into 10 sets, each representing a different food category. Another similar attempt was with VGG-F, another CNN architecture. This attempt

extracted a total of 4096 features, then constructing into respective food category sets. The actual experiment can be found in Section 5.1.4 Convolutional Neural Network Features Evaluation.

3 IMAGE PREPROCESSING & ADJUSTMENTS

3.1 Principal Component Analysis (PCA)

When it comes to finding the best classifier to express the data, the actual focus should be verifying how good the data is. The entire data collection process inevitably introduce noise into the data set, and even produce redundant features due to repetitive sampling from the same dimension (Shlens, 2014). With some of my early feature extractors, each image had too many features extracted, which introduces computational complexity to the problem. There is a need to reduce the number of features to speed up the overall training and classification process. Principal component analysis (PCA) greatly reduces the number of features for a given image, by removing insignificant features or representing correlated features with the one combined feature. Another benefit of PCA was its improvement in accuracy, since insignificant features were removed from causing bias. Furthermore, there is no need to resize the images with less features left after PCA. An example of how PCA contributed to classification accuracy can be seen in Section 5.3 Data Pre-processing.

3.2 Duplication Removal

Since the input images were collect from the internet and there was no verification of the quality of those images. One thing that I noticed was that some images had duplicates with different file names collected from different sources. I decided to remove the duplicates since I did not want to give more weight to certain images because they have more copies.

3.3 Error Analysis

Although error has decreased significantly throughout the research process, I was still interested in unveiling insights from the misclassified food items. I wanted to know what the computer thinks versus what the actual food category a given image belongs to. A detailed experiment and analysis on misclassified images can be found in Section 5.3 Data Pre-processing/Adjustment Evaluation.

3.4 Pre-classification Randomization

When performing training and classification I applied two layers of randomization to minimize bias due to fixed image ordering within the folder and the training/testing set. The first layer is getting random input images from each food category. After the images are extracted from the respective folders and constructed into different images sets (one for each class), a training and testing set was formed using random images from every category being examined. I performed the second layer of randomization to shuffle the images within the training set and testing set to ensure the images from the same class are not adjacent. For example, without this layer of randomization the training set could have been "apple", "apple", "apple", "burger", "burger", "burger". The results of how randomization affected the accuracy can be found in Section 5.3 Data Pre-processing/Adjustment Evaluation.

3.5 Mean Subtraction

In most classifiers, features with greater numeric values will dominate the results. Therefore, I subtracted each feature in the dataset by the mean value of all features to get an adjusted value. At first, I only performed mean subtraction after the training set and testing set were constructed. But after accidently performing mean subtraction before constructing any training sets, which is twice in total, I noticed an increase in accuracy. The results and experiments with mean subtraction can be seen in section in 5.3 Data Pre-processing/Adjustment Evaluation.

4 MODEL TRAINING & CLASSIFICATION

Initially, when I researched on comparing regression models and classification models, the main difference is regression models are used to predict a real value given some input values, whereas, classification models are used to predict a class given some input values. In this research project, classification models are more relevant to achieving our goal. But I will include regression models since it was part of my learning process.

4.1 Simple Model Comparison

4.1.1 Linear Regression

The linear regression was more suitable for cases where we know the exact features and these features are very representative of the food images. Therefore, we can use these features to accurately predict a result using the regression function. The objective of linear regression is to

minimize the Euclidean distance of input data from the predicted linear function by constantly adjusting the linear function's parameters. But usually, the relationship between/among features with the output class is not linear and could have a very complex relationship.

4.1.2 Perceptron Batch

Perceptron Batch is designed as a binary classifier, and it will either output a number greater than zero or less than zero. After conversion, the output will either be one of the two class values. Therefore, I would be hesitant to use this model as the final classifier since we will be performing multiclass classification rather than just binary classification.

4.1.3 K-nearest Neighbor (KNN)

K-nearest Neighbor (KNN) is a special model where training does not take place. For fast experimentation, I would use this model as it is fast to implement and efficient to get results right away. Also, this model requires storing all the input data in memory while performing classification, which limits how much input images and the variety of classes the research can cover. Therefore, I decided not to use this model for classifying a wide variety of food images.

4.2 Simple Model Classification

For simple models, I started with two classes, namely Apple and Burger to see the accuracy of each model on binary classification. In terms of implementation, since different classes are represented as different textual food categories (i.e. "apple", "burger"), therefore, before classification, we must convert all categories into numerical classes for the classifier to correctly process the input. After comparing the different results of various feature extractors, for simple models, I have decided to go with SURF as the feature extractor, since edge detection in Matlab only outputs a logical 0 or 1, which provides less value for classification. After running Linear regression and perceptron batch for binary classification, we obtained results which can be seen in Section 5.2 Classifier Evaluation. As for KNN, by using different feature extractors, I have obtained a wide range of accuracies. The results and discussion can also be found in Section 5.2 Classifier Evaluation.

4.3 Support Vector Machines (SVM)

After performing classification with the simple models, I attempted to use the Support Vector Machine as one of its biggest advantages and characteristic is to find a separating plane that gives the largest margin (Hsu, Chang & Lin, 2003). This characteristic enables the learning model to perform well on new data (test data) since it will not likely to overfit the model compared to traditional models like Linear Regression. My approach was to compare different feature extraction approaches and evaluate the results with SVM classifier.

With limited time for this project, I decided to use the LibSVM library for implementation. Initially, I attempted to do binary classification between Apple and Burger. There are various kernels that are supported by LibSVM (Hsu, Chang & Lin, 2003). I decided to apply the Radial Basis Function (RBF) kernel which is the default kernel for the library. This kernel is favorable over other kernels when the number of features is not too large. Also, compared to linear regression models, this kernel is able to handle nonlinear relationships between the class labels and the input data. Other kernels are also applied for comparison. The results of SVM binary classification can be found in Section 5.2 Classifier Evaluation.

Fortunately, LibSVM provided the option to have probabilities for multi-class classification, which is used as my one-vs-all multiclass classification approach. The model calculated all the probabilities for a given image across all categories. A highest probability in class A means the food image, based on the given features, most likely belongs to class A. Another important detail to note is that SVM imposes a margin requirement that the highest probability has to be higher than all other probabilities by this margin value to be chosen as the predicted class. This implies the accuracy of SVM is much higher than other simple models and provides a another layer of information (probabilities for all predictions) for error analysis on misclassified food items. To verify how each feature extractor performed, I used SVM as the classifier for every feature extractor. The actual classification results can be found in Section 5.2 Classifier Evaluation.

4.4 Convolutional Neural Network (CNN)

Initially, I tried an interesting test on how accurate my input images were by classifying using GoogLeNet's original output classes. For the "apple" class, since there were no "apple" class among the 1000 classes for the "ILSVRC" challenge, therefore, the majority of predicted classes

was “Granny Smith”, which is a specific kind of apple. To accommodate my own class labels, I decided to use GoogLeNet as the feature extractor combined with the SVM classifier. The actual experiments and evaluations can be seen in Section 5.1.4 Convolutional Neural Network Features Evaluation. When studying CNN, I attempted to train my own neural network from scratch using custom layers, but due to limited computational power, I directed my focus on using pre-trained CNN to boost food image recognition performance for this research project.

4.5 Cross-validation

Some learning models require the researcher to modify parameters to achieve the best result on new data (test data), which leads to another important step in classification – cross-validation. This step involves separating the original data into 3 different sets, training set, validation set and testing set. After training is performed on the training set, we will use the trained classifier on the validation set. The purpose of this step is to have an initial view of the model’s performance on unknown data. An improved model is achieved by adjusting the parameters of the classifier to achieve a lowest possible error rate on the validation set. After numerous iteration of trying different parameters, we will obtain a set of parameters that maximizes the classification accuracy. Once a fairly high accuracy rate is obtained, we can test the improved classifier on the third set – testing set. This experiment can be found in 5.2.1 Cross-validation to Optimize the SVM Classifier.

5 EXPERIMENTATION & EVALUATION

5.1 Feature Extractor Evaluation

5.1.1 Pixel

Starting with the simplest feature extractor, the naïve pixel features. They were grouped into sets for each image, which was plugged into a SVM classifier for classification. Due to limited computational power, I resized the images to 128x128 and 200x200 for experiments which, in effect, gives 16,384 features and 40,000 features per image. For the purpose of rapid testing, I used binary classification with the Apple and Burger class. The classifier managed to have an accuracy rate of 62.98% for both sizes of the images. [Table 1] This accuracy rate is fairly low, given that “random guessing” would give 50% accuracy rate. I doubt it would scale better with more classes.

Table 1 - Pixel and Edge Features Results (Two classes)

Binary Classifier (SVM[1]) with Apple and Burger class				
Image Size[2]	**Feature Extractor**	**Training Size (per class)**	**Test Size (per class)**	**Test Accuracy**
128x128	Raw Pixels	700	524	62.98%
200x200	Raw Pixels	700	524	62.98%
128x128	Prewitt	700	524	71.37%
200x200	Prewitt	700	524	73.66%
200x200	Canny	700	524	83.59%
200x200	Sobel	700	524	71%

5.1.2 Edge Detectors

As for edge detectors, my hypothesis was it should give higher accuracy than plain pixels, since edges gives more information about the object than pixels. Indeed, the experiments I ran with three different edge detectors, namely Prewitt, Canny and Sobel, gave promising but not extraordinary results. I resized the images to 128x128 and 200x200 formats before extracting the edges. Then I plugged the extracted edge data into the SVM classifier for binary classification. Out of the three edge extractors, Canny gave the highest accuracy of 83.59%. [Table 1] To verify its scalability on more classes and more input data, I ran multiclass classification using 900 training images and 600 testing images. The results were not favorable, the best edge extractor Canny had an accuracy of 58.83% over three classes. [Table 2] Recall that "random guessing" would give 33% accuracy, which implies that edge extractors would not be a good feature extractor in the long-term. Prewitt and Sobel edge detection algorithms tend to perform poorly with noisy images, which we cannot avoid in practice. The Canny edge detection algorithm depends on the the size of the Gaussian filter to smooth out noise. However, having larger Gaussian filters will render inaccurate localization of the edge (Maini & Aggarwal, 2009).

Table 2 - Edge Features and SURF Results (Multiclass)

Multi-class Classifier (SVM[3]) with Apple, Burger and Coffee				
Image Size	**Feature Extractor**	**Training Size**	**Test Size**	**Test Accuracy**
200x200	Sobel	900	600	56.50%
200x200	Canny	900	600	58.83%

[1] Default RBF kernel
[2] Resized due to limited computational power
[3] Default RBF kernel with one-vs-all for multiclass classification

200x200	Canny	1500	600	58.50%
300x300	SURF (9+ features)	1050	450	48.22%
500x500	SURF (9+ features)	1050	450	52.67%
500x500	SURF (15+ features)	1050	450	53.56%
500x500	SURF (27+ features)[4]	1050	450	47.78%
500x500	SURF (27+ features)	1500	600	52.83%

5.1.3 SURF

SURF was another attempt as a feature extractor. Input images were also resized to 300x300 and 500x500 for efficiency reasons. Since different images resulted in different number of features, I set several thresholds for leaving out images that outputted feature numbers less than the set minimum. For all experiments, I used 1050 to 1500 training images and 450 to 600 testing images. The best result came from a minimum of 15 features per image that had a 53.56% accuracy rate. [Table 2] Compared to edge detection, SURF did not seem to perform very well. This led to my next experiment of combining SURF extractor with the Bag of Features approach. My initial motivation and hypothesis was SURF extracted many weak features that are not representative of the food class. Having another layer of grouping features together, synthesizing and filtering useful features, I believed accuracy would increase as a result. In my next experiment, since extracted SURFs would go through a "filtering" stage and would not be computational expensive, I decided to keep the image size as original to retain as much detail as possible. With three classes, a total of 1272 images were used as the training set, and 849 images were used as the testing set. The SVM classifier was used again as an evaluator of the feature performance with the target of converging to 500 bags (groups) of features in the end. SURF with Bag of features achieved a test accuracy of 79% and training accuracy of 85%, and 81% test accuracy with resized 300x300. [Table 3] This result was a huge improvement over plain SURF extractors, which motivated me in scaling the classes from Apple, Burger and Coffee to 10 classes. 10-class classification achieved a test accuracy of 46%. Although results show that this feature extractor was not acceptable to be considered as a practical one, it was better than previous feature extractors. In addition, to gain a

[4] Removing the bottom 5% images that contain the least features results in keeping images with more than 27 features

better understanding of how bag of features work, I decided to visualize the features in the process of forming 500 bags of features. [Figure 2 & 3]

Table 3 - SURF with Bag of Features Results

Multi-class Classifier (SVM[5]) with Bag of Features			
Image Size	**Training Size**	**Test Size**	**Accuracy**
3 classes			
Full	669	669	Train: 86%
Full	669	1557	Test: 77%
Full	1272	1272	Train: 85%
Full	1272	849	Test: 79%
300x300	1332	1332	Train: 85%
300x300	1332	888	Test: 81%
10 classes			
300x300	4440	2960	46%

Sample 1

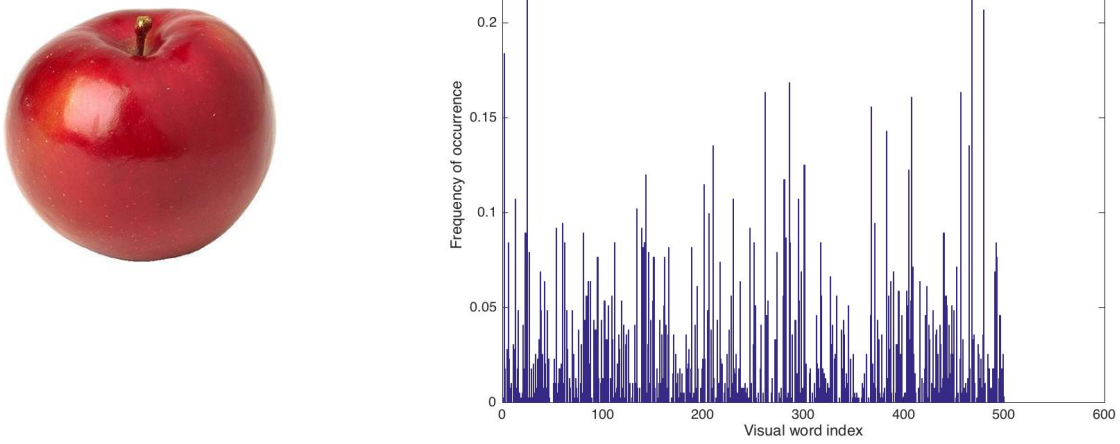

Figure 2 - Bag of Words Visualization (a)

[5] Linear kernel with multiclass option

Sample 2

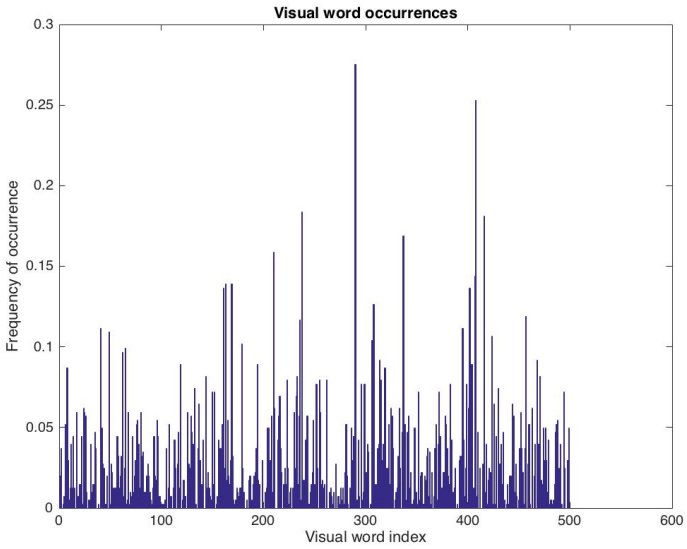

Figure 3 - Bag of Words Visualization (b)

5.1.4 Convolutional Neural Network Features Evaluation

Our next experiment uses Convolutional Neural Network (CNN) to extract various small patches of features based on the GoogLeNet architecture. I was curious to how the intermediate layers looked like when a food image was passed through 22 parameter-generating layers (Szegedy et al., 2015). In Figure 4, the upper left image is the input image of an apple that was used for this experiment. The upper right image is 64 patches of 112x112 output from the first convolutional layer. Each small patch shows how the original image reacts to various convolutional filters, each filter is "interested" in a certain part of the image. Together they form the activation map of the original apple image. The bottom left image shows the output after the second convolutional layer. There are 192 56x56 patches. As humans, it is already very difficult to see how each patch relates to the original apple image. But this is merely the second parameter-generating layer. The bottom right image is the second last parameter-generating layer, which provides the 1024 features for the given apple image.

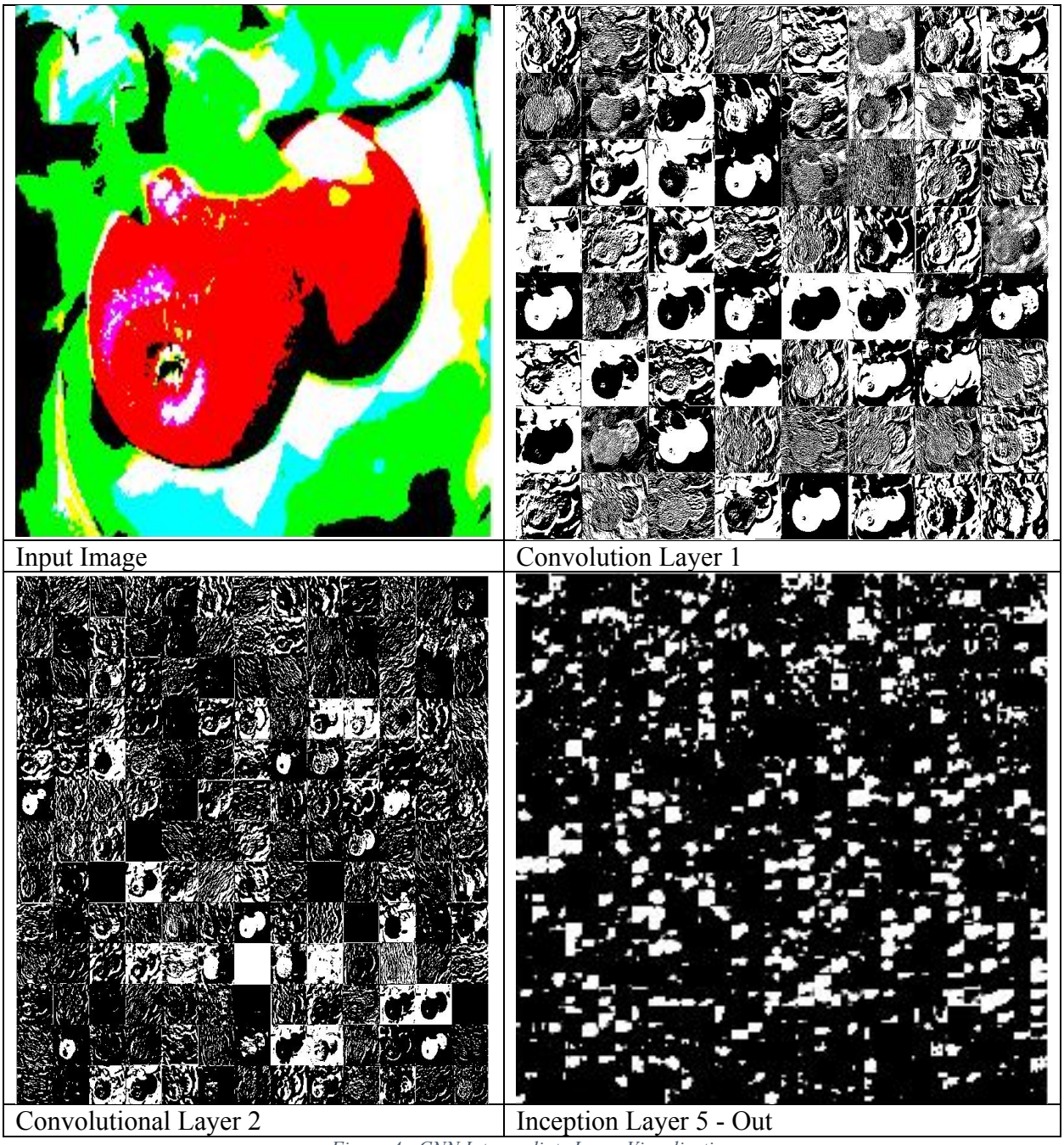

Figure 4 - CNN Intermediate Layer Visualization

5.2 Classifier Evaluation

To evaluate our classifiers, I started by testing the simple models first. My first experiment was using linear regression as the classifier on two classes, Apple and Burger, and using SURFs and extracted GoogLeNet features as the input. The results show 55.75% accuracy for the SURFs only roughly 6% better than "random-guessing" and CNN features had 61% accuracy which is still not a favorable result as a good classifier. [Table 4]

Table 4 - Linear Regression Results

Linear Regression (MSE) for Apple and Burger				
Image Size	**Feature Extractor**	**Training Size**	**Test Size**	**Test Accuracy**
Full	SURF (Top 9)	800	400	55.75%
Full	CNN - GoogLeNet	800	400	61%

My next experiment was with perceptron batch with the same two classes for binary classification. The accuracy was low for SURF extractor at 66% with a batch size of 1000. [Table 5] The accuracy was extremely low for CNN features with the accuracy at 41.5% with a batch size of 10 and the accuracy decreases as the batch size increases. This result aligns with the fact that perceptron is a linear classifier and the nature of food image classification is not linear.

Table 5 - Perceptron Batch Results

Perceptron Batch Normalized for Apple and Burger					
Image Size	**Feature Extractor**	**Training Size**	**Test Size**	**Test Accuracy**	**Batch Size**
Full	SURF (Top 9)	800	400	63.75%	10
Full	SURF (Top 9)	800	400	62.75%	100
Full	SURF (Top 9)	800	400	66%	1000
Full	CNN - GoogLeNet	800	400	41.5%	10
Full	CNN - GoogLeNet	800	400	20.75%	100
Full	CNN - GoogLeNet	800	400	21.50%	10000

As for KNN classifiers, it performs better than simple linear models since it is non-linear, thus being able to generate dynamic boundaries based on the input features. Using SURF as the feature extractor, the accuracy with the three-nearest neighbors produced an accuracy of 59.5% which is not very favorable compared to "random-guessing". [Table 6] However, with GoogLeNet's features, the classifier achieved an accuracy of 95.5% with three-nearest neighbors. This discovery proved that KNN classifiers depend hugely on the quality of the features. But considering the storage requirement of KNN while performing the classification, it might not be ideal if the amount of data increases exponentially in the future.

Table 6 - K-nearest Neighbor Results

KNN for Apple and Burger					
Image Size	**K**	**Feature Extractor**	**Training Size**	**Test Size**	**Test Accuracy**
Full	3	SURF (Top 9)	800	400	59.50%

Full	5	SURF (Top 9)	800	400	54.75%
Full	3	GoogLeNet	800	400	95.50%
Full	5	GoogLeNet	800	400	95%

Up until now, the SVM classifier has been used widely across my research to test the performance different feature extractors and to verify how different preprocessing methods are able to produce better features. This SVM classifier analysis will focus on its various parameters, and the process of discovering the most suitable set of parameters for food image recognition. GoogLeNet's features has proved to produce the best accuracy, so it will be used in this experiment. The first experiment will be using LibSVM's default kernel, the Radial Basis Function (RBF) [Equation 1], and it's default γ (gamma) value of $\frac{1}{\#\ of\ features}$. The results were very favorable with a test accuracy of 99.33% for binary classification. [Table 7] Switching the kernel to the polynomial function [Equation 2], with a default d (degree) of 3, and the default γ (gamma) value of $\frac{1}{\#\ of\ features}$, and the default r (coefficient) of 0, the classifier yielded a slightly lower accuracy rate of 98.67% for binary classification. To verify the scalability of the the two kernels on multiclass classification, the next experiment is based on multiclass classification with different kernels.

Equation 1 - Radial Basis Function (RBF) Kernel

$$Radial\ Basis\ Function\ (Hsu, Chang\ \&\ Lin, 2003)(RBF):$$

$$K(x_i, x_j) = exp(-\gamma \parallel x_i - x_j \parallel^2), \gamma > 0.$$

Equation 2 - Polynomial Kernel

$$Polynomial\ (Hsu, Chang\ \&\ Lin, 2003):$$

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$$

Table 7 - SVM Binary Classifier Kernel Comparison

Binary Classifier (SVM) with GoogLeNet Feature Extractor				
Training Size	**Test Size**	**Test Accuracy**	**# of Classes**	**Kernel**
900	300	99.33%	2	RBF

900	300	98.67%	2	Polynomial

This second experiment focuses on multiclass classification with three different kernels. [Table 8] The default RBF kernel performed consistently well, and resulted in 98.1% accuracy for 5-class classification and 96.93% accuracy for 10-class classification. Unfortunately, the polynomial kernel did not scale well to multi-class and yielded an accuracy of 81.47% for 10-class classification. The sigmoid kernel performed poorly with 26.53% which proves to not be suitable for multidimensional non-linear food image data, and will not be further investigated in this research project. In the next section, I will use cross-validation to adjust the parameters of the SVM classifier to achieve the highest accuracy on multiclass classification.

Table 8 – SVM Multiclass Classifier Kernel Comparison

Multi-class Classifier (SVM) with GoogLeNet Feature Extractor				
Training Size	**Test Size**	**Test Accuracy**	**# of Classes**	**Kernel**
1250	800	95.2%	5	RBF
2000	1000	98.1%	5	RBF
3500	1500	95.07%	10	RBF
4500	1500	96.93%	10	RBF
4500	1500	81.47%	10	Polynomial
4500	1500	26.53%	10	Sigmoid

5.2.1 Cross-validation to Optimize the SVM Classifier

Using cross validation for the SVM classifier, I will focus on the RBF kernel with adjusted γ (gamma) and c (cost). My first cross-validation was with 3 folds, which means that the initial training data set is divided into 3 subsets, in each iteration, one of the subsets will be used as the test set and the other two subsets are combined into the training set. For this experiment, I used 10 classes of food images as the input, and the GoogLeNet's features. The cost parameter $c = 0 \pm step\ size$ and gamma $\gamma = \frac{1}{\#\ of\ features} \pm step\ size$. Multiple step sizes were tried to balance efficiency and precision. After running cross-validation over three iterations, the two parameters are finally set, which is then used to perform classification on the actual test set. The results show that using default LibSVM parameter values yielded 96.93% for 10-class classification. With cross-validation, the same classifier, using adjusted parameters of $c = 32\ \ \gamma = 0.00097656$, achieved 98.07% for the same 10-class classification. Due to limited image data, cross-validation might not perform well after dividing the training data into subsets, which is something to take

into consideration in the future. Despite the increased accuracy, another purpose of cross-validation is to avoid having the same data being used to adjust the parameters and verifying the result of the adjustments, thus separated testing and validation data. To visualize the classification results with fine-tuned parameters, refer to Figure 5, and its confusion matrix in Figure 6.

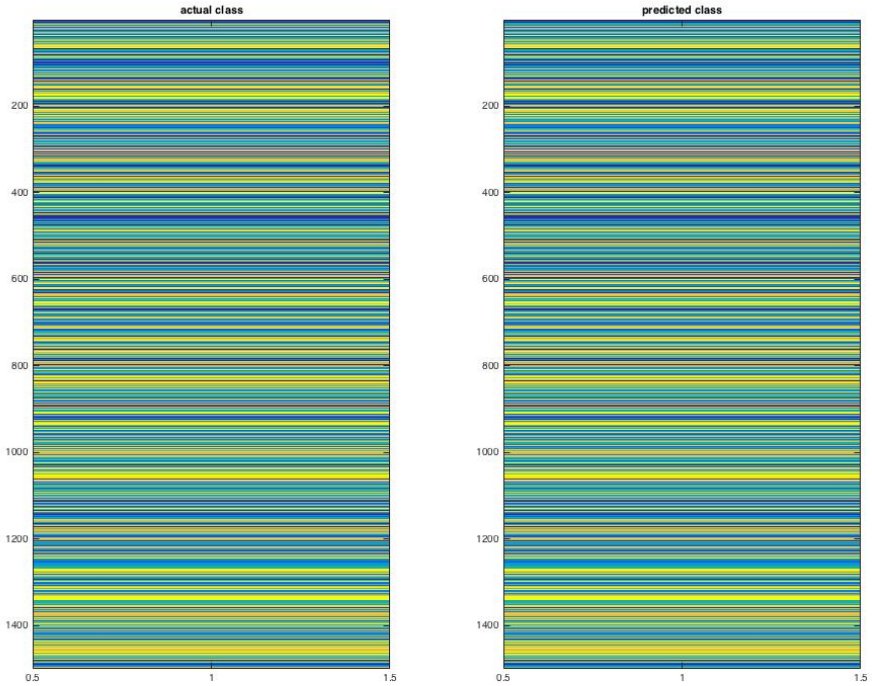

Figure 5 - SVM Multiclass (Actual vs Predicted)

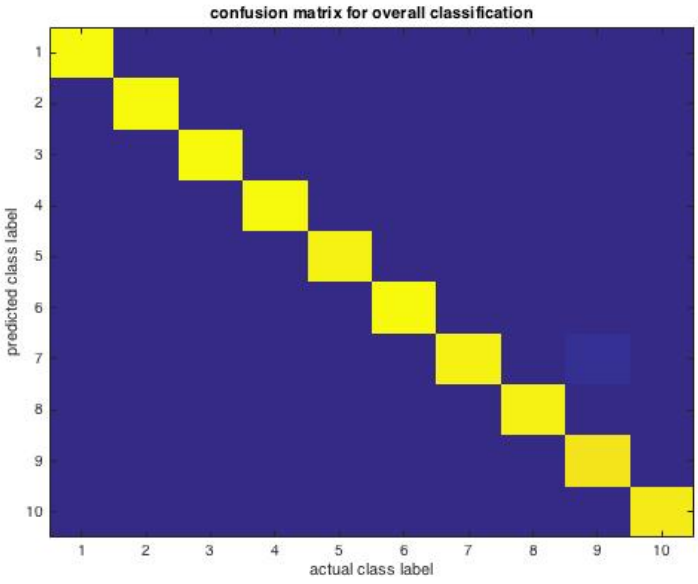

Figure 6 - Confusion Matrix (10 class)

5.3 Data Pre-processing/Adjustment Evaluation

Having too many features can cause computational complexity and reduced accuracy. I have attempted to use principle component analysis (PCA) to reduce the dimension/features of each image data. The first experiment was to apply PCA on the 4096 features extracted from VGG-F. My motivation was some features might be weak and the overall accuracy from the prior VGG-F experiment was fairly low. PCA can only generate (N – 1) principal components, and with 700 images in the coffee set, the top 700 principal components was used for this experiment (Shlens, 2014). The top 700 principal components explains 98.23% of the total variation in the apple image set. 3-class classification after PCA yielded 66.89% accuracy rate, slightly higher than the previous 58.22% in 3-class classification. [Table 9]

Table 9 - Principal Component Analysis Experiment

PCA Result Comparison						
Image Size	**Training Size**	**Test Size**	**Test Accuracy**	**# of classes**	**Feature Extractor**	**PCA**
Full	1350	450	58.22%	3	CNN-VGG-F	N/A
Full	1350	450	63.53%	3	CNN-VGG-F	PCA - Top 700
Full	4500	1500	96.93%	10	CNN-GoogleNet	N/A
Full	4500	1500	79.33%	10	CNN-GoogleNet	PCA- Top 627

Applying PCA on the high accuracy GoogLeNet + SVM classification resulted in a much lower test accuracy of 79.33% for 10-class classification, mostly because PCA reduced the feature number from 1024 to 627 features, which was, again, constrained by the number of images in the coffee set. [Table 9] These results imply that we must gather more images to have the flexibility of keeping more principal components. Also with a base feature count of 1024 for GoogLeNet + SVM, there is no need to do PCA as the dimensions are still manageable.

An important procedure prior to classification is image set randomization and feature normalization. The first experiment was to verify the effectiveness of randomization in datasets. This experiment was utilized GoogLeNet to extract features combined with 5-class SVM classification. Prior to randomizing input images within the training set and testing set, the accuracy was 98.8% and after randomization, the accuracy decreased slightly to 98.53% which

means there was a little bias caused by having classes adjacent to each other, and randomization was better to reflect the actual accuracy of the classifier. [Table 10] As for mean subtraction, a second experiment utilizing the same feature extractor and classifier was performed on 10-class classification, and the accuracy was greatly improved by mean-subtraction from 90.6% to 95.13%. This shows that some features were indeed on different scales, causing certain features to mislead the classification result. With mean subtraction, all features being adjusted to the same scale, resulting in more accurate features and more accurate labeling. The third experiment was similar to the second experiment with minimum 27 SURF features extracted from input images classified using the SVM classifier with 3 classes. The result aligns with improving the overall accuracy from 53.67% to 54.67% after mean subtraction.

Table 10 - Normalization and Randomization Results

Normalization/Randomization Comparison				
Training Size	**Test Size**	**Test Accuracy**	**# of Classes**	**Type**
1750	750	98.8%	5	Before random training/testing
1750	750	98.53%	5	After random training/testing
3500	1500	90.6%	10	Before mean subtraction
3500	1500	95.13%	10	After mean subtraction
1500	600	53.67%	3	Minimum 27 SURF features, without mean subtraction
1500	600	54.67%	3	Minimum 27 SURF features, with mean subtraction
4500	1500	96.93%	10	Only mean subtraction for training/testing set
4500	1500	97.73%	10	Mean subtraction twice for original image data set and training/testing set

To gain a better understanding of areas of improvement for the classifier, I summarized all the misclassified images for 10-class[6] classification from SVM with CNN features. Figure 7 shows the different scenarios where misclassification could take place:

- The image contains more than one food item, creating interclass distraction
- The image is taken at a very close angle, causing parts of the food item to be amplified and focused, resulting in misleading features

[6] Apple, coffee, French fries, ice cream, salad, burger, donuts, fried rice, ramen and sashimi

- The image contains cluttered background, resulting in mixed features that are not representative of the subject food item

This suggests that future users should be advised of the potential pitfalls when taking photos, and we should have some preprocessing stage that cleans up the image to filter out the subject food item prior to feature extraction.

		
Ground Truth – Salad Prediction - Burger	Ground Truth – Burger Prediction - Donuts	Ground Truth – Sashimi Prediction – Ice Cream

Figure 7 - Misclassification Results

6 CONCLUSION

Based on the results of numerous experiments across the research period, simple models tend to underfit the input image data or inefficient for processing huge amounts of data, and complex classifiers must be combined with accurate features. This leads to our final result of using GoogLeNet's CNN to extract features and plugging those features into SVM for multi-class classification. For data pre-processing, mean subtraction and data randomization provided a more accurate result. Finally, cross-validation is used to find the best set of parameters that gave the highest accuracy of 98.07% with 10-class classification.

In the future, it would be best if a custom CNN can be trained with an architecture suitable for food image data. Also, collecting more images for training and cross-validation is necessary to improve the overall accuracy and to reduce bias. An experiment on whether image metadata improves classification accuracy and efficiency can be performed to unveil another different dimension of data. Geotags can be used to identify where the photo is taken to narrow down specific restaurants, then train specialized classifiers for specific types of restaurants to reduce intraclass variability for the same food across different restaurants. Moreover, cluttered images being inevitable in practice, applying saliency detection to extract the subject food item from the

background before feature extraction and classification is an option to improve the classification results. Finally, it would be convenient from the user's standpoint if the classifier is able to extract multiple food items from one image.

REFERENCES

Code used in this research can be found at: https://github.com/workofart/food-recognition

Aly, M. (2005). Survey on multiclass classification methods. *Neural Netw*, 1-9.

Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification.

Maini, R., & Aggarwal, H. (2009). Study and comparison of various image edge detection techniques. *International journal of image processing (IJIP)*,*3*(1), 1-11.

Ng, A. (n.d.). Machine Learning – Stanford University | Coursera. Retrieved from https://www.coursera.org/learn/machine-learning

Shlens, J. (2014). A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-9).

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014* (pp. 818-833). Springer International Publishing.

Zhang, W., Yu, Q., Siddiquie, B., Divakaran, A., & Sawhney, H. (2015). "Snap-n-Eat" Food Recognition and Nutrition Estimation on a Smartphone.*Journal of diabetes science and technology*, *9*(3), 525-533.

CS231n: Convolutional Neural Networks for Visual Recognition. (n.d.). Retrieved from http://cs231n.stanford.edu/