Discrete Wavelet Transform and gray-scale image analysis for real time estimation of calorific energy in food products

1st Dr. A. SenthilSelvi
Assc. prof. Dept. of Computer Science
SRM Institute of Science and Technology

2nd Ishmeet Singh Kalra
Dept. of Computer Science
SRM Institute of Science and Technology

3rd Sarthak Pandita
Dept. of Computer Science
SRM Institute of Science and Technology

4th Sanchay Mishra
Dept. of Computer Science
SRM Institute of Science and Technology

5th MD Kazmi Alam
Dept. of Computer Science
SRM Institute of Science and Technology

***Abstract*—Computer vision techniques have demonstrated great success in being able to classify complex food products. The challenge however, lies in extracting any further information, such as the nutritional value in these products. We propose a method using grayscale features of the image to classify and extract information from an image. The discrete wavelet transform is used to create 4 combinations of high pass & low pass filters through which we are able to enhance the features by edge detection and fine-grain analysis at different levels. Outputs from the filters are coupled together to achieve a compressed image with enhanced features from which we extract our required grayscale feature data. The various vectors obtained are coupled as a singular tensor which is fed into a neural network to perform the classification task in form of a supervised learning, where we make use of the ingredients as our labels. Our solution assumes that the targets are clearly separable from the background which must be plain. Humans describe food items and their differences in terms of the ingredients contained within and the way these ingredients are arranged, so it makes sense that we should be able to learn extra information by extracting features at an intermediate ingredient level instead of just the pixel level. Our goal with this method is to achieve a good balance between classification accuracy as well as the time and processing power required to perform the same. Our largest challenge is the differences in volume of raw ingredients as well as the high intra-class variations within similar groups.**

***Index Terms*—Machine Learning, Food, Calorie, Computer Vision, GLCM, Discrete Wavelet Transform, Neural Networks**

I. INTRODUCTION

In this paper we demonstrate a challenging classification problem in the domain of Computer Vision. Unlike other objects like faces or cars, food products have a very high intra-class variation, and can have different methodologies of naming. Various methods have been previously employed for this task however they struggle in balancing accuracy and performance such that higher score requires a lot of power and time both for training as well as prediction. In order to tackle the problem of performance, we make use of discrete wavelet transform to compress our input through a combination of high-pass & low-pass filters, such that we are also able to enhance the features required for the task. At each step, we down-sample the images by half in terms of rows or rows first and columns later. While the complete low pass combination provides us the compressed image, other filters provide information visible at original resolution such as the edges. Through the various ray Level co-occurrence matrix(GLCM) features such as co-occurrence, energy, entropy, contrast, correlation coefficient & homogeneity, we obtain vectors that describe our image in various forms. The combination of these vectors is used as input to a neural network for the classification task. We rely on limited level of ground truth ingredient data to distinguish between similar types of food, the spatial features of which are represented by the vectors from GCLM features, as such, differentiating between very similar foods like sauces, burgers or pasta, can be extremely difficult, since their description lies in method of preparation rather than the appearance of final product. Our method is able to achieve a good amount of increase in accuracy while reducing the amount of time required for classification. The main limitation within the approach is that we assume that the food & the background are clearly distinguishable, which requires that the background be plain and have a moderate to high contrast in terms of color & intensity as compared to our target food product. Once identified and labeled, we can make use of the output to fetch calorie value from a database which lists the same against ingredient names, which is provided to the user for a certain fixed quantity of the food product, assuming the ratio of ingredients contributing the most calories remains constant.

II. Previous Work

Our work is mainly based on past work by Yang et al. [11] which classified food products through pairwise statistics of local features, introducing the idea of random sampling of the pixels from the image & using histograms of the statistics like distance & orientation as features in a Support Vector Machine classifier. Pixel-level ingredients labels have also been used as intermediate features in [12], however, this work has focused only on global ingredient histogram instead of using the pairwise statistics. Some other works have attempted a combination of both global and local features, such as global histograms, which is followed in [1]. Some less related work has focused on extracting various different features, like bag of SIFT & text-ons [5]. Joutou and Yanai have combined these features within a single classifier using multi-kernel learning [6].Kong and Tan have shown that food is classified much better with having multiple viewpoints [7], although this approach is feasible only if the we take a video of the food product instead of just an image, using a mobile device. Our aim is to expand on these works by integrating the pairwise statistics with other approaches from various sources.

III. Background

A. Discrete Wavelet Transform

In numerical & functional analysis, a discrete wavelet transform (DWT) is any wavelet transform for which the wavelets are discretely sampled. DWT has many applications in science, engineering, mathematics and computer science. Within the field of imaging, the discrete wavelet transform is used an algorithm for compression both lossy & lossless. It can be performed within $O(n)$ operations and captures both the frequency content as well as the temporal content of the input, by examining it at different scales & the times at which these frequencies occur. Most notably, it is used to represent a discrete signal in a more redundant manner, often as a precondition for compression. It has been shown that DWT (discrete in scale & shift, and continuous in time) has been successfully used as an analog filter bank in biomedical signal processing for designing low-power pacemakers as well as in certain ultra-wide band (UWB) wireless communications.

Wavelets are used to denoise 2-dimensional signals, like images. The first step involves determining a wavelet type, and level of decomposition(N). Bi-orthogonal wavelets have been commonly used in image processing for detection and filtering of the White Gaussian noise, due to the high contrast in the neighboring pixel intensity values. After the decomposition of image, the next step involves determining the threshold values for each level of decomposition. Birgé-Massart strategy is a common method for calculating the thresholds. Using this process, individual thresholds are made for all levels of decomposition. The filtering is composed mainly of applying these thresholds to the signal. The final step is reconstruction of the image from these modified levels, which is accomplished through an inverse wavelet transform. The most important part of filtering any form of data is quantifying the signal- to-noise-ratio(SNR) of the result. Choosing different wavelets, levels, and threshold strategies can result in different outputs of filtering.

The DWT of any signal x is calculated by passing the same through a series of filters. First, these samples are passed through a low pass filter with an impulse response g, which results in convolution of the two:

$$y[n] = (x * g)[n] = \sum_{k=-\infty}^{\infty} x[k]\,g[n-k] \qquad (1)$$

Simultaneously, the signal is also decomposed by using a high-pass filter h. The outputs of which give the detail coefficients (from high-pass filter) & the approximation coefficients (from low-pass). It is integral that the two filters be related to each other. This is known as a quadrature mirror filter.

However, since half of the frequencies from x have been discarded, according to Nyquist's rule, half of the samples may also be discarded. The output of low-pass filter g is sub-sampled by 2 & can be processed further by passing it through a new low-pass filter g and a high-pass filter h with half the cut-off frequency of the previous, i.e.:

$$y_{low}[n] = (x * g)[n] = \sum_{k=-\infty}^{\infty} x[k]\; g[2n-k]\,,$$

$$y_{high}[n] = (x * g)[n] = \sum_{k=-\infty}^{\infty} x[k]\,h[2n-k] \qquad (2)$$

This decomposition has now halved the time resolution, due to only half of each filter output characterizing the signal. However, since each output has half the frequency band of input x, the frequency resolution has now been doubled. This decomposition process is repeated for further increasing both the frequency resolution and the approximation coefficients, which have been decomposed with high & low pass filters & then down-sampled. The process is represented as a binary tree, with the nodes representing a sub-space with a different time-frequency localization. This tree is known as the filter bank.

B. GLCM Features

The Gray Level Co-occurrence Matrix (GLCM) and associated texture feature calculations are image analysis techniques that are used to extract the spatial relationship between a set of pixels within an image. Given an image composed of pixels, each with a specific intensity (gray level), the GLCM describes how often different combinations of gray levels co-occur in an image or section of the image. Texture feature calculations use contents of a GLCM to provide a measure of the variation in intensity, at any pixel of interest. Each element (i,j) of the resultant GLCM is the sum of the number of times the pixel with value i occurred in the specified spatial relationship to another pixel with value j within the input image.The GLCM is able to reveal certain properties about the spatial distribution of the gray levels in the image. For example, if most entries in the GLCM for a given image are concentrated along the

diagonal, the texture is described as coarse with respect to the specified offset. The various features analyzed to construct the GLCM are:

Energy, which is a measure of homogeneity of the image and can be calculated from the normalized COM. It is a suitable measure for detecting disorder in the image & is calculated as:

$$J = \sum_{i=1}\sum_{j=1} (p(i,j))^2 \quad (3)$$

Entropy, which is a measure of complexity of image. Complex textures tend to have higher entropy, which is given by:

$$S = \sum_{i=1}\sum_{j=1} (p(i,j)) \log(p(i,j)) \quad (4)$$

Contrast measures local variations and the texture of shadow depth in the GLCM & is given by:

$$J = \sum_{i=1}\sum_{j=1} (i-j)^2 (p(i,j)) \quad (5)$$

Where, $(p\ (i,\ j))$ is the GLCM

Correlation Coefficient is used to measure the joint probability occurrence of the specified pixel pairs as:

$$\sum \cdot \sum \cdot \frac{(x-\mu x)(y-\mu y)\,p(x,y)}{\sigma_x \sigma_y}^{\Sigma\Sigma} \quad (6)$$

Homogeneity measures the closeness of the distribution of elements in the GLCM in comparison to the GLCM diagonal by equation:

$$\sum \cdot \sum \cdot \frac{p(x,y)}{1+[x-y]}^{\Sigma\Sigma} \quad (7)$$

The number of gray levels in the image determines size of the matrix, which in most cases would be $length \times width \times features$ and may be padded with arbitrary values to make convert into a square matrix if the provided input itself is not of equal dimensions within a 2-dimensional space.

C. Backpropagation Neural Networks

Neural networks are a type of machine learning algorithms, modeled after the biological network of neurons, which constitute the nervous system. They consist of a series of nodes, known as neurons, which are arranged into layers. The algorithm aims to derive the underlying relationships within a finite set of data, which is described as the learning process. Each connection between the neurons is assigned a weight which describes its relative importance, along with bias of its constituent layer. The first layer is called input layer and the final layer is known as the output layer,while all the others in the middle are known as hidden layers. Each neuron within a layer may be connected to every single neuron in the following layer, which is known as a dense connection. As such, the combination of outputs from all neurons in a layer are passed to all the neurons in the following layer in a process called feed-forward & the network is known as a feed-forward neural network.

At a given time, only a subset of the outputs from a layer may be important in the operation, which we can define by the use of a specialized function known as the 'Activation Function'. With $a_i^{(j)}$ denoting a set of neurons, where i is the i^{th} neuron in $(j-1)^{th}$ layer and θ_{ik} being weights for the i^{th} neuron and k^{th} target in the current layer and 'g' as the activation function, the activation for any neuron can be given by:

$$a_i^{(j)} = \sum_{k=0}^{n} g(\theta_{ik}^{(j-1)} x_k) \quad (8)$$

Where, n is the total number of neurons in the j^{th} layer. Outputs from (8) are then used with the weight vector $\theta^{(j)}$ as x from current layer. Calculating similarly for all layers gives us our hypothesis function $h_\theta(x)$. In a simplified sense, the activation function g creates the threshold for a neuron, below which it's output is not used in the calculations of following layer.

A large amount of concurrent floating point operations are performed, as such it is possible to estimate the extent of difference between our desired outcome and the one produced by the neural network. The set of all outputs is plotted over a multidimensional space along with the expected set of outputs, which is defined as the global optima within a convex function. The estimate of difference is given by the cost function, which gives the average of the square of distances between the set of outputs & the optima and is given as:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) \quad (9)$$

However, our activation is of the form of a non-convex function having multiple local optima, as such (9) is not guaranteed to converge on global minimum. For this reason, we make use of the log function:

$$Cost(h_\theta(x^{(i)}), y^{(i)}) = \begin{matrix} -log(h_\theta(x)) & \text{if y=1} \\ -log(1-h_\theta(x)) & \text{if y=0} \end{matrix} \quad (10)$$

which can be simplified as:

$$Cost(h_\theta(x^{(i)}), y^{(i)}) = -y\,log(h_\theta(x)) - (1-y)\,log(1-h_\theta(x)) \quad (11)$$

When this is written fully with summation, we get:

$$j(\theta) = \frac{-1}{m} \sum_{i=1}^{m} [-y^{(i)} log(h_\theta(x^{(i)})) - (1-y^{(i)}) log(1-h_\theta(x^{(i)}))] \quad (12)$$

Equation (12) gives us the case for only one node. If we generalize this to multiple output nodes for multi-class classification, we obtain:

$$j(\theta) = \frac{-1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} [-y_k^{(i)} log((h_\theta(x^{(i)}))_k) - (1-y^{(i)}) log(1-(h_\theta(x^{(i)})_k))] \quad (13)$$

The cost from (13) is conveyed backwards, in a process called backpropagation. The neural network aims to optimize the

distance over this convex function by adjusting the weights in small steps wherein a value close to 0 emphasizes presence of the gradient, while a value close to 1 denotes approaching convergence. Various different methods of optimizations have been developed, which describe the descent over the convex hyperplane, among which the most commonly used are 'gradient descent(and stochastic gradient descent)' & 'adam'. The process is described as complete when successive iterations do not cause any significant reduction in the observed error rate. Learning rate is used as a parameter to denote the magnitude of alteration the while adjusting weights across the network.

IV. METHODOLOGY

A. The Food-101 Dataset

We make use of food-101 dataset for our work. The said dataset contains 101 categories of images where each category consists of 750 training samples and 250 test samples, thus making a total of 101,000 images. A large number of the samples are filled with intense color or noise along with mislabeling in a few of the samples. We corrected the labels manually by ourselves for this work. The images have been scaled down to resolution of 256x256 for inputs to the wavelet transform.

B. Pre-processing

Food classification is a challenging task due to very-high intra-class variations. Before the data is provided to classifier, we need to ensure that it is capable of dealing with more realistic scenarios including but not limited to rotation, angle and lighting. We also need to identify variations in textures of similar looking food like curries.

In order to both optimize to performance as well as obtain the relevant features, we process our images through a combination of high-pass and low-pass filters within a discrete wavelet transform. We configure a 9-tap/7-tap integer wavelet transform by 2 layers each having a high-pass filter(H) & a low-pass filter(L), thus providing us 4 possible combinations of outputs as LL, LH, HL & HH. The following outputs are obtained after the procedure:

- *LL:* Image with lossless compression. Used as input to speed up the operations.
- *LH:* Edge detection at original resolution.
- *HL:* Highlighted edges and important features.
- *HH:* Smoothed boundaries for important features and edges.

The outputs from various filters have been shown in [13]. These are superimposed to isolate only the required features from our compressed images, leaving us with only our relevant data free of noise. The remaining areas are filled with pitch black pixels, as shown in Fig. 1 to maintain the 256x256 size standard throughout the dataset.

Fig. 1. Left: sample from database. Right: After superimposition

We follow-up with various image transformations to introduce more variance. The following operations are performed:

- *Rotation(45°, 90°, 180°& horizontal):* Images are rotated in random direction at various angles to increase the diversity, as well as to ensure we can classify at any angle. 180°rotation provides a vertically flipped output.
- *Wide Shift(0.2):* The images are shifted horizontally with this margin, allowing incomplete inputs for classification
- *Height Shift(0.2):* The images are shifted vertically with this margin, allowing incomplete inputs for classification
- *Brighten(20%):* Brightness of images is increased to allow variations in lighting for classification. We chose the 20% value as any further increase added no value & only increased white intensity, which is not a real-world scenario.
- *Darken(15%):* Images are darkened to allow classification indoors and under artificial lighting. We fixed the maximum 15% value as any further only reduced white intensity and removed focus from our images, leading them to be perceived as a different class.

We thus have a total of 12 transformations, increasing the size of dataset to 12,12,000 images total with 12000 images in each category. We use the same ratio of 75% training images & 25% validation images.

We further construct the GLCM matrix for all of these images. The various feature vectors are combined into a singular tensor, which is to be used as an input to our neural network. For each image we obtain a 256*256*5 tensor, which is split into 5 matrices, input at different points in our neural network. We can also flatten the tensor to a 1280*256 matrix, which however increases the size of our network and makes it unsuitable for smaller systems like smartphones.

C. Architecture

Our neural network consists of inputs are various layers for different set of features obtained from our inputs. Compressed image post superimposition as the input. We allow a buffer of 3 layers before providing any further inputs, in order to adjust the weights & allow our classifier to extract the features for which the GLCM has been constructed. The reasoning for the same is explained below in table IV, which shows the input distance between compressed image and GLCM features input in number of layers. The buffer in between allows the network to predict boundaries for the relevant features. This allows us to completely ignore the pitch black pixels, which in turn reduces the number of nodes required in the layers,

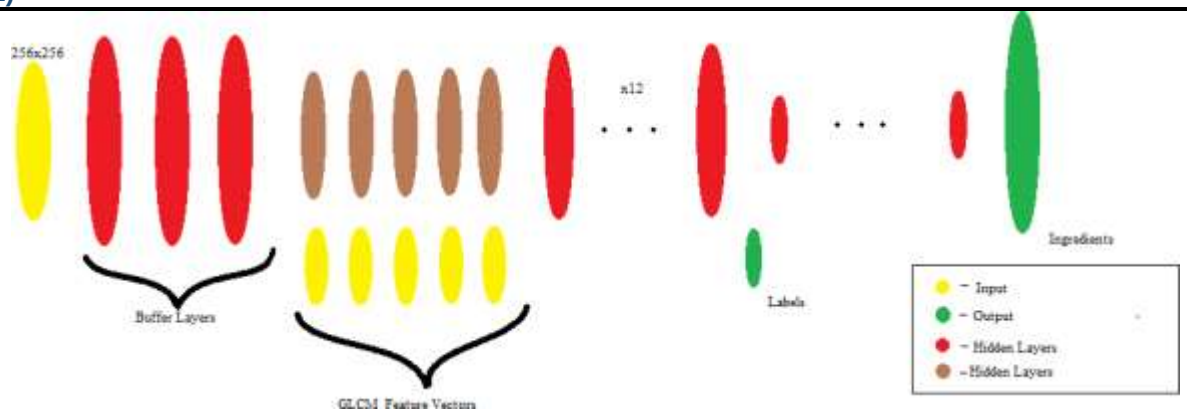

Fig. 2. Architecture of the network

reducing the size of our network. The optimization allows us to improve the processing efficiency by working only on the features deemed relevant.

Our network consists of 56 layers including the input & output layer. The input layer is a flattened 256*256 nodes, which is the size of our inputs. The first 3 hidden layers, which are also the buffer between image & GLCM inputs consist of 2048 densely connected nodes. This configuration was performed as the final step after completion of the rest of the network. We tested with powers of 2 ranging between 256 nodes to 4096 nodes, and their effect on classification accuracy vs. performance is given in table I. Note that our goal is not to obtain the highest possible classification accuracy, but to achieve a high efficiency in terms of accuracy vs. time taken to classify the image. The 5 layers after the buffers consist of 512 hidden nodes and 256*256 flattened input nodes for every GLCM feature. In these layers we are not concerned with adjusting weight for the image, but only relating the features to the input. As a consequence, the next 12 layers consist of 1024 nodes each. All the rest of the layers are composed of 256 nodes. The output layer has a softmax activation, which provides the confidence level for each label.

We use a modified form of Sparse Categorical Crossentropy(SCCE) as our loss function, in the form of a supervised learning task. We are not concerned with the variance in confidence(which is expected to be high between the most confident & 2nd most confident prediction for a given input, with a softmax activation). With this, we are able to drop the task of one-hot encode at final layer, which although minor, still provides a performance improvement in less powerful systems.The final obtained vector is not label of the food itself, but the possible ingredients that are present in the same. We modified the SCCE loss function to output 1 for every ingredient that it predicts might be present, based on the input. This is done by clipping the probabilities to the minimum obtained value among ones with highest confidence. To identify the same, we look at the value after which there is a sharp drop in confidence, which shows that a certain ingredient is less likely to be present. For a cake, this can be seen in the form of just high enough confidence in sugar, which has least effect on texture(but still has some), while the confidence for sweet potatoes which may be the next highest value, is extremely low in comparison, thus all the values are clipped at sugar. This method has its demerits since we have a high chance of introducing unwanted ingredients, which are not present. To tackle this, we introduce a correlating factor *s*, which penalizes the network for introducing these ingredients and is calculated as:

$$s = mean \cdot \sqrt{(\Sigma(np))^{\Sigma}} \quad (14)$$

Where *np* is the list of confidence values of ingredients not present, and is added to the error for every mistake in the encoding vector. For ex. if our classifier predicts presence of 7 ingredients, out of which only 4 are correct, we multiply the penalty by 3 and add it to the error obtained. We, however, also need the the label of food in order to carry out the calorie estimation task. Since we can identify the food much easier than the ingredients present in the same, we can obtain the output somewhere in the middle. Through trial & error, we were able to obtain our food label, without any further alterations to our loss function, at layer 32, with high accuracy. We attach 101 output nodes with the same softmax activation and our customized loss function at this point with the hypothesis that if the labels are incorrect, then so will be the ingredients.

D. Calorie Estimation

The task of calorie estimation is simple. We can either scrape the web for estimated calories, or we can have them stored in a database against the ingredients. We go with PostgreSQL for this task and store the calorie values for every 100g of given ingredient. For any given recipe, we have a standard ratio of ingredients that must be present, we apply the same ratio to 100g and obtain result for the ingredient for specified weight, thus keeping our target weight for final product at 100g as a standard measure. In simpler terms, if a recipe has 2:3:5 ratio of ingredients a, b & c, which gives us

TABLE I
COMPARISON BETWEEN NO. OF NODES AND EFFICIENCY

Nodes per buffer layer	Accuracy(Max.)	Initial Processor Speed(GHz)	Final Processor Speed(GHz)	Time(s)	Efficiency
256	68.37%	0.70	1.42	29	3.17
512	74.88.%	0.74	1.51	30	3.2
1024	81.46%	0.81	1.65	31	2.30
2048	84.36%	0.93	1.78	38	2.61
4096	88.92%	0.83	2.10	47	1.48

TABLE II
COMPARISON BETWEEN NO. OF NODES AND EFFICIENCY

Model	Accuracy(Max.)	Initial Processor Speed(GHz)	Final Processor Speed(GHz)	Time(s)	Efficiency
SVM	51.87%	0.93	1.47	36	2.66
Multiple Kernel Learning	76.58.%	0.93	2.18	45	1.36
Convolutional Neural Networks	86.97%	0.93	2.38	41	1.46
Ours	84.36%	0.93	1.78	38	2.61
Random Forest	81.98%	0.93	2.21	38	1.68

a 20% for ingredient a, 30% for b & 50% for c, we obtain 0.2*(calories of a) + 0.3*(calories of b) + 0.5*(calories of c) to obtain the calories of our dish per 100g. We keep this ratio as a fixed value for every category in our dataset and use it to calculate the total calorie value after values for all ingredients has been obtained from the database.

V. EXPERIMENTAL RESULTS

The problem of food classification is an extremely hard task & as such, demands a very complex approach with high processing requirement. Our goal here is to achieve the best possible balance between classification accuracy, resources consumed and time taken. Each of our experiment is run on the same system with i7-8750h processor with 6 cores & 12 threads, 2x8(16) Gigabytes of DDR4 RAM & Nvidia GTX 1070 GPU. The efficiency of the task is calculated as:

$$\frac{A}{(S_f - S_i) \times T} \quad (15)$$

Where, A is the accuracy in percent, S_f is the maximum processor speed during classification task in GigaHertz(GHz), S_i is maximum processor speed before classification task in GHz & T is the time taken in seconds. We present the results of our methods and a few others in table II.

The results of identification of ingredients with and without our customized loss function are presented in the table III

TABLE III
RESULTS WITH & WITHOUT CUSTOM LOSS FUNCTION

Model	Accuracy(Max.)
Without Loss	43.74%
With Loss	87.95%

We start by training our network only to label the food images, in order to determine the buffer between the input image and the GLCM features. The task is carried out as a trial & error. As we can see in table IV, a buffer of 3 layers works the best, as the network is able to recognize the edges of features for which the GLCM has been constructed.

TABLE IV
BUFFER LAYERS VS. ACCURACY

Buffer	Accuracy(Max.)
0	71.57%
1	78.83%
2	81.55%
3	**84.36%**
4	77.43%
5	74.82%
6	69.34%

The loss function for our network in labeling task is presented & compared with a few other solutions in Fig. 3 The optimal number of nodes in terms of efficiency in these

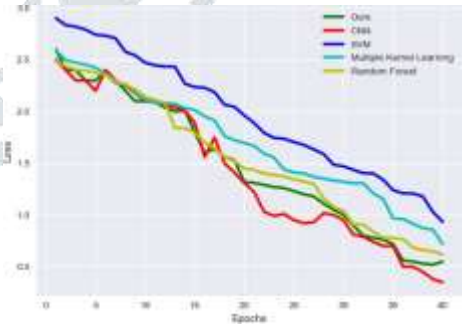

Fig. 3. Observed loss in different methods

buffer layers were determined afterwards. We use powers of 2 starting from 256 nodes, up to 4096 nodes in each layer. The results are presented, for a 3 layer buffer, in table I

The optimal number of layers for labeling the food product & ingredients have been shown in tables V & VI. The number of nodes used in every layer are kept a standard power of 2 for simplicity purposes & the optimal count per layer is only for performance purposes. While we were able to obtain similar results with lesser or more nodes per layer, the number of

TABLE V
NO. OF LAYERS VS. ACCURACY(FOOD LABELING) AT 35 EPOCHS

Layers	Accuracy(Max.)	efficiency
25	65.84%	3.63
30	72.39%	3.18
35	79.64%	2.41
36	80.17%	2.28
37	82.63%	2.21
38	**84.36%**	**2.30**
39	85.19%	2.26
40	85.38%	2.17
42	87.21%	1.89

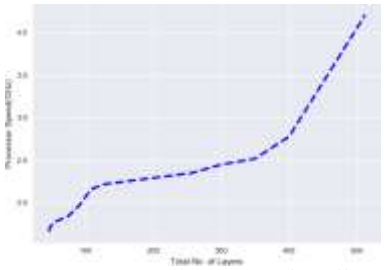

Fig. 5. Processing speed at classification task for different no. of layers

layers grew in a logarithmic curve as shown in fig. 4, however this affected our performance in a exponential curve as shown in Fig. 5.

TABLE VI
NO. OF LAYERS VS. ACCURACY(INGREDIENT LABELING) AT 35 EPOCHS

Layers	Accuracy(Max.)	efficiency
40	59.61%	2.69
45	66.18%	2.43
50	72.93%	2.31
52	76.48%	2.33
54	79.24%	2.19
56	**81.76%**	**2.28**
58	82.14%	2.15
60	82.97%	2.06
65	84.54%	1.84

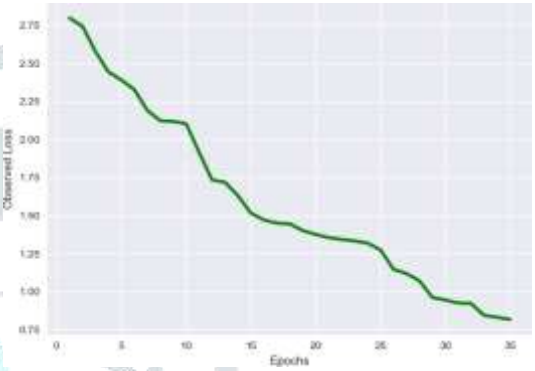

Fig. 6. Loss Through training epochs

The loss observed through training of the entire network, consisting of both labeling of products & ingredients in 35 epochs is shown is fig. 6.

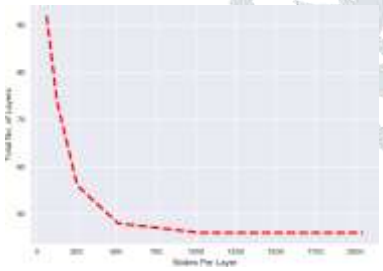

Fig. 4. No. of nodes vs layers required for similar accuracy

The classification accuracy as well as efficiency for every network is gauged in table II, after a standard 35 epochs of training, with no observed overfitting in any solution. There are no tasks in background that require any computational resources during our experiment.

CONCLUSION

In this paper, we have presented a method to solve the highly complex problem of food classification with a good efficiency in terms of accuracy vs time & performance. Our network requires high amount of resources for the training, however is highly efficient afterwards in the prediction phase. We have presented our results and comparison of our method against a few popular ones. We have been able to accurately classify images in various lighting conditions with varying dynamic ranges & at different angles. There is a possibility of increasing accuracy by extending the network further, which however is not the goal here. We see that our model is suitable for lower end systems with less processing power despite its inherent complexity of architecture and an acceptable accuracy in classification. We can improve the model further with a more complex dataset, which however is yet not available in literature, for benchmarking purposes. While our loss function has displayed good results for ingredient identification, there is also a possibility of creating a more complex one to further improve the efficiency of classification.

REFERENCES

[1] M. Bosch, F. Zhu, N. Khanna, C. Boushey, and E. Delp.Combining global and local features for food identification in dietary assessment. In 18th IEEE International Conference on Image Processing (ICIP), 2011, pages 1789–1792.IEEE, 2011.

[2] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available athttp://www.csie.ntu.edu.tw/cjlin/libsvm.

[3] M. Chen, K. Dhingra, W. Wu, L. Yang, R. Sukthankar, andJ. Yang. Pfid: Pittsburgh fast-food image dataset. In IEEE International Conference on Image Processing (ICIP), 2009 16th , pages 289–292. IEEE, 2009.

[4] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray.Visual categorization with bags of key points. In Workshop on statistical learning in computer vision, ECCV, volume 1,page 22, 2004.
[5] H. Hoashi, T. Joutou, and K. Yanai. Image recognition of 85food categories by feature fusion. In IEEE International Symposium on Multimedia (ISM), 2010, pages 296–301. IEEE,2010.
[6] T. Joutou and K. Yanai. A food image recognition system with multiple kernel learning. In 16th IEEE International Conference on Image Processing (ICIP),2009 , pages 285–288. IEEE, 2009.
[7] F. Kong and J. Tan. Dietcam: Regular shape food recognition with a camera phone. In International Conference on Body Sensor Networks (BSN), 2011, pages 127–132. IEEE, 2011.
[8] D. Lowe. Distinctive image features from scale-invariant key points. International journal of computer vision,60(2):91–110, 2004.
[9] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008. , pages 1–8. IEEE, 2008.
[10] J. Shotton, M. Johnson, and R. Cipolla. Semantic text on forests for image categorization and segmentation. In Computer Vision and Pattern Recognition, 2008. CVPR 2008.IEEE Conference on, pages 1–8. IEEE, 2008.
[11] S. Yang, M. Chen, D. Pomerleau, and R. Sukthankar. Food recognition using statistics of pair wise local features. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pages 2249–2256. IEEE, 2010.
[12] M. Zhang. Identifying the cuisine of a plate of food.
[13] Ahmed A. Nashat, and N.M. Hussain Hassan. Image Compression Based Upon Wavelet Transform and A Statistical Threshold. International Conference on Optoelectronics and Image Processing (ICOIP), 2016
[14] David J. Attokaren, Ian G. Fernandes, A. Sriram, Y.V. Srinivasa Murthy, and Shashidhar G. Koolagudi. Food classification from images using convolutional neural networks. TENCON 2017 - 2017 IEEE Region 10 Conference