SeeFood

Okan Alan, Gokberk Sahin, Emre Yazici
Hacettepe University
Department of Computer Engineering
okanalan97.14@gmail.com | sahingokberk@gmail.com | emreyazicihumb@hotmail.com

Abstract

People want to know how much calories they eat. For this purpose, we propose a new Machine Learning approach. Our dataset contains 2870 images of 19 different types of food with a single China coin. We used Faster R-CNN for object detection and classification. Using the GrabCut algorithm we get the contour of each food and coin. We predict the volume with a single coin as a reference point and calculate the mass in grams. Finally, we calculate the calorie of the given food. Also, proposed models can directly estimate the calorie with the same set of features which we discovered later.

Index Terms—- *Calorie Estimation, Object Detection*

1. Introduction

Obesity is a medical condition and means you have too much body fat. When your Body-Mass Index(BMI) is between 25 and 29.9 it means that you are carrying excess weight. If your BMI is over 30 then it means you have obesity.

Obesity can have multiple reasons. One of these reasons is consuming too much calorie. Consuming too many calories means that amount of calories that you are taking is bigger than amount of calories you burn. The body stores the excess calorie as body fat.

To lose weight or maintain the healthy weight people needs to keep track of the calorie they take. But this process can be difficult and tiring. Because people tend to avoid difficult and tiring things, they often do not track how much they eat. And this might lead to obesity.

In recent years, there is several studies[5, 6, 8] in this area(see Section 2). Among these studies, two main factors of the accuracy change are object detection algorithm, volume and calorie estimation method. For example, Support Vector Machine(SVM) is used for object detection and classification. For volume and calorie, the estimation reference point is what makes the difference. Using different reference points(coin, thumb, playing card etc.) effects the practicality of the application. Purpose of this study is to make this tracking easier. For this, we come up with a Machine Learning Base approach. With only two pictures(one is from the side and one is from the top) of the food and a single coin, people will be able to know the calorie of the food that they are eating.

In this study we find and classify the food and the coin as reference point(see Section 3.2 and 3.3.) and make a prediction about the volume of the food. Finally, we calculate the calorie of the food based on the volume that models have predicted(see Section 3.4 and 3.5). However, we discovered that estimating the calories directly was giving us much accurate results.

2. Related Works

Pouladzadeh *et al*. [6] proposed an approach that segments the image and predicts the calorie of the food in the given image. They used K-Mean clustering for the image segmentation and find the food portions in the image. For classification Support Vector Machines(SVM) is used. The SVM model was trained in order to classify the food that given. SVM model took Color Features(10 categories), Size Features(6 categories), Shape Features(5 categories), Texture Features(5 categories) and classify the food. They needed two images(one is from the side and other is from the top) which are contains food and the user's thumb for the volume estimation. They reached 92.21% of accuracy.

Flows of below approach

Using Thumb: For the volume prediction using users, the thumb is not an efficient way. Because other than the registered user whoever uses this than the accuracy will drop.

Only Works on Simple Food Images: This approach does not work on complex foods like soup, sandwich etc. it only works on simple foods like apple, banana etc.

Chokr *et al*. [5] used a dataset of fast-food images based on the Pittsburgh Fast-Food Image Dataset[2]. Mathworks Image Processing Toolbox is used for extracting features. Total of 11,868 raw features extracted from RGB representation of the image. Raw features reduced using Principal Component Analysis(PCA) and Information Gain(InfoGain) to 23 features. Using these features and Sequential Minimal Optimization(SMO) they classified the food. Size prediction is done by using Random Forest in grams. Finally, the calorie of the food is predicted using Multilayer Perceptron. They experiment on different types of representation of an image such as Averaged RGB, Gray Scale, BW 0.7 and BW 0.5 but they get the best results with RGB representation.

Flows of this approach

Size estimation: For the size estimation, they used the ground truth value of the food from its manufacturer. Because of that this method give a low accuracy on the foods from other manufacturers.

White Background They used a white background for the foods in this projects dataset. Because of that, this project is not efficient for daily usage.

Liang *et al*. [8] used a dataset of 2978 images of 19 different food with a single yuan coin. They used Faster R-CNN for object detection. Each bounding box that created by Faster R-CNN is classified. They used the GrabCut algorithm for image segmentation. They divided food into 3 categories ellipsoid, column and irregular. Using the coin as a reference point they calculate the volume of the food depending on its shape. Knowing the volume mass can be easily calculated using the density of the food. With the mass is known they calculated the calorie of the food.

3. The Approach

3.1. DATASET

3.1.1 Description

We used ECUSTFD [9] (ECUST Food Dataset) in the SeeFood. ECUSTFD is a free public food image dataset. In ECUSTFD, foods volume and mass records are provided, as well as one China coin is used as the calibration object.

ECUSTFD has 19 types of food: apple, banana, bread, bun, doughnut, egg, fried dough twist, grape, lemon, litchi, mango, moon cake, orange, peach, pear, plum, kiwi, sachima, tomato. There is also mix object that is two foods in one image but we did not use them. There are 2978 images in the dataset but we delete mix images and we used 2870 images of 19 different foods. The dataset contains a top view and a side view of photos.

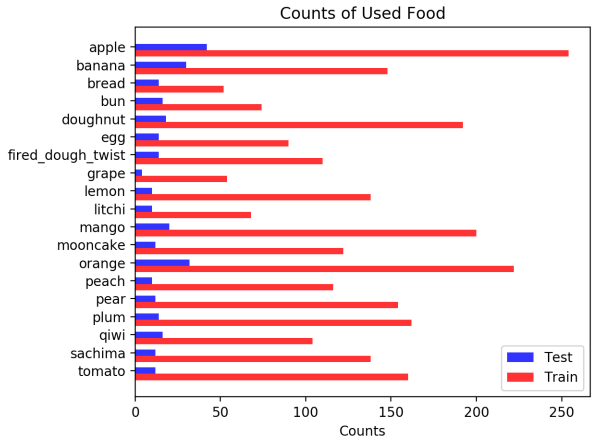


Figure 1. Number of used images for object detection training and testing

ECUSFTD provide other information for each image as follows:

(1) Annotation We did not deal with labeling pictures because ECUSFTD has the images that are labeled. We use labeled images while training object detection models. These labeled images have 2 bounding box label, one for food and one for coin.

(2) Mass and Volume Our dataset provides real mass and volume for each food in the image.

Images shooted different conditions. For example, some photos were taken in a dark place without using a flash light. Another example is background, in the majority of photos have used 2 different plates, there is not in other.

Every image has China coin. The diameter of the coin is 25 millimeter. The coin is used for the reference point. We use this to estimate volume.

3.2. OBJECT DETECTION

We used Tensorflow Object Detection API[3] for our project. We experiment on several object detection model[4].

Model Name	Speed (ms)	COCO mAP[^1]
ssd_mobilenet_v1_coco	30	21
faster_rcnn_inception_v2_coco	58	28
faster_rcnn_resnet50_coco	89	30
faster_rcnn_resnet101_coco	106	32
faster_rcnn_nas	1833	43
mask_rcnn_inception_v2_coco	79	25

We trained this models on our dataset of 312 test images and 2558 train images.

ssd_mobilenet_v1_coco: We trained this model for several days but we could not drop its loss to the level we want

so we gave up on this model.

faster_rcnn_nas: Using this model was not very efficient. Complexity of this model makes it consume so much memory and works much slower than other models. For these reasons it's not practical for us to use this model.

faster_rcnn_inception_v2_coco: This model was the one that worked for us. When we tested this model it gave us high accuracy on finding coin and food objects in the images. It could easily find the coin and food with high accuracy.

Because of these reasons we chose to use Faster R-CNN Inception v2 Coco model in our project. Faster R-CNN Inception model can find objects in the image and classify them correctly. It puts the objects that it found into the bounding boxes. These bounding boxes will be used for volume estimation.

3.3. GrabCut[1]

GrabCut is an algorithm that extracts an object to the foreground. If it needs to simplify, background in the image is deleted by GrabCut. Deleting means in here is changing background pixels with black.

Firstly, we detect an object then we bound the object with bounding boxes. Using bounding box, we put the images into GrabCut algorithm. We can see what GrabCut do in Figure 2

In our project we used GrabCut algorithm to count pixel detected foods. These pixels are the numerical form of an image. Then we are using them to estimate volume.

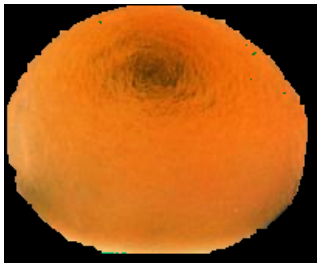


Figure 2. GrabCut example on orange from our dataset

3.4. K Nearest Neighbors(K-NN)[7]

While we estimating the volume of given foods, we used K nearest neighbors algorithm. It can be used for both classification and regression problems. In our project, we are making a prediction with regression.

K nearest neighbors is a simple algorithm. We choose nearest K point to the test point and determine the class which is most recurrent. However, when we regress, after selecting the nearest K neighbor, we estimate value that is calculated by dividing the sum of the neighbor's values by K. It means our prediction value for the object is average of the value of its K nearest neighbors.

3.5. Random Forest

3.5.1 Motivation

In this problem we mainly preferred a decision tree since our number of features are limited and decision trees are very convenient to use in this case. The reasoning is we have two separate photos of a given food, one taken from top and the other one taken from side of the food. Consequently we have some features that are seemingly independent at first thought however some of them highly correlated as a result of this two image approach. We can see the correlation matrix below in Figure 3. Random forest randomly selects from these set of features and provides the most accurate results. Also if we look at the formulas in our baseline work there are some non-linear relationship between these features. Since decision trees don't assume a linear relation between features this approach will provide us some interesting feature engineering options.

3.5.2 Exploring the Data set

Let's explore the correlations between our features so we can figure out how to combine them to get more interesting features.

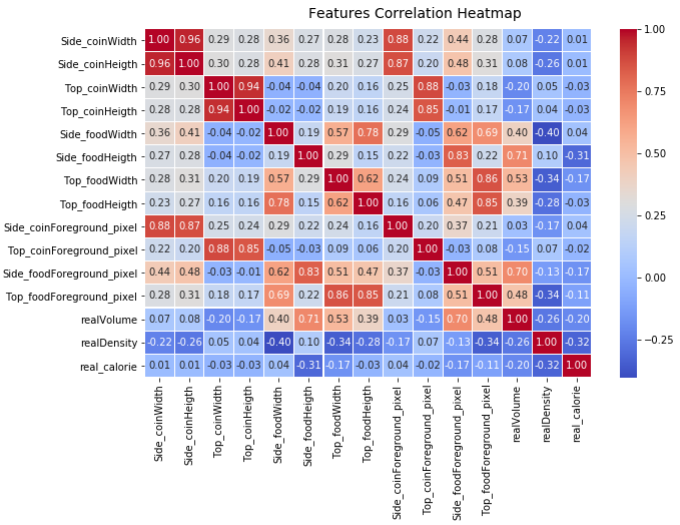


Figure 3. Correlation Matrix of the features

As we can see there is a high correlation between side width and top height features. This correlation makes sense because the width dimension of the side image is the same dimension with a height of the top image. Furthermore, both dimensions from both images of the coin are also highly correlated. This is because of the shape of the coin

which is round. In this set of features, Random Forest gives us an ability to randomly sample these features and build multiple decision trees out of them which makes the model more stable and accurate.

All of the features are self-explanatory however we are trying to estimate the volume of the object so we don't expect this model with this set of features to be the most accurate one. There is one feature worth mentioning here which is the real density of the food. Now we can't use this feature directly since the density of a food can vary based on the percentage of water and other ingredients but we can take the average of the density since variation is not big. If we give all of these features without any feature engineering with hand-tuned hyper-parameters we get a mean error of **20.76**. To increase this accuracy we can combine our features such a way that it, they will align with the formulas mentioned in our baseline work.

3.6. Feature Engineering

Firstly to make sense of the width and height of the food and coin we multiply the width and height of the food to get area of the bounding box. We do the same thing for coin and we get the "top ratio" and "side ratio" by dividing food area to coin area. All results mentioned in this section obtained from %20 percent of our data set which we used for testing.

$$\beta_T = \frac{F_h F_w}{C_h C_w}$$

$$\beta_S = \frac{F_h F_w}{C_h C_w}$$

β_T and β_S are top and side ratios respectively. F_h and F_w are height and width of the food respectively. When we are calculating the top ratio we are using top width and height when we are calculating the side we are using side width and height.

After adding these two features our model's mean error decreases to **14.56**. Since we now provided some ratio values for both top and side images it's no surprise our model performed much better. Because before adding these features there is no way our model can distinguish the relationship between coins in the top and side images and their relevance to the food's bounding box. Except there can be a case when one of the decision tree in the random forest accidentally samples just top or side features of a image which is almost not possible considering the other features. In any case these features will help the model distinguish the top and side images.

We can apply the same logic to number of foreground pixels of coin and food for both side and top images.

$$\theta_T = \frac{F_p}{C_p}$$

$$\theta_S = \frac{F_p}{C_p}$$

θ_T and θ_S are top and side foreground pixel ratios respectively. F_p and C_p are foreground pixels of the food and coin respectively. The purpose of this feature is very similar to previous one so we don't expect a huge decrease in mean error as we initially saw. After adding these two features, our mean error is decreased to **14.18**.

To estimate the volume we should know the real size of the coin and use it to get some ratio. We are going to follow a basic formula similar to our baseline work, we are just going to divide the radius of the coin to sum of width and height of its bounding box.

$$\alpha_T = \frac{2.5}{(C_w + C_h)}$$

$$\alpha_S = \frac{2.5}{(C_w + C_h)}$$

α_T and α_S are the top and side coin ratios respectively. C_t and C_h are the width and height of coin's bounding box. If we were to calculate the volume of the food by using formulas these ratios will determine the volume because they give the relationship between real life sizes and image sizes. Note that the constant 2.5 will change if we change the coin to be a different size. We can plug in any coin radius instead of this constant value given all images in our data set uses the same coin. After adding these two features we can see a small decrease in our mean error which is now **14.17**.

Now we can combine our new features to get another feature which resembles the formula in our baseline work. Since the model don't assume features have linear relationship we can just multiply all ratios with the number of foreground pixels. The purpose of this feature is to give the model a tangible metric to use when estimating the volume.

$$\sigma_T = F_h F_w \theta_T \alpha_T$$

$$\sigma_S = F_h F_w \theta_S \alpha_S$$

σ_T and σ_S are the symbolic top and side area of the food respectively. These features are highly correlated with real volume as it resembles the real volume formula from our baseline work. After adding these two features our mean error decreases to **13.97**.

Lastly, we can use "kcal" (energy per grams) as a feature since it's a constant value for specific type of food. Our data set don't provide us this value so we find it by dividing the total calories to food's mass.

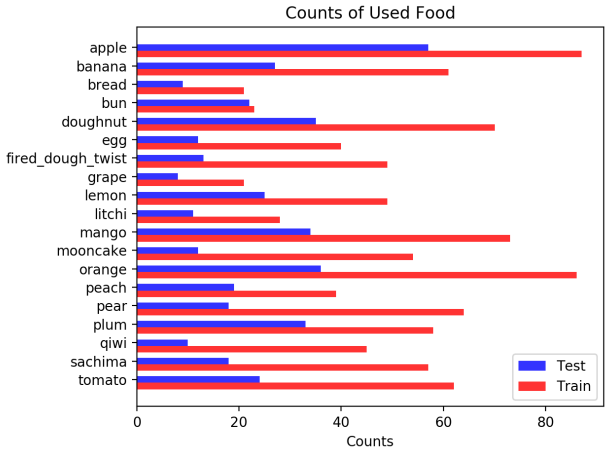
$$kcals = \frac{C_f}{V_f \rho_f}$$

C_f, V_f, ρ_f are total calories, real volume and density of the food respectively. After adding this feature we see a drastic decrease in error which is now **11.63**.

There are two features we can't directly use in our model which are image name i.e food name and food's shape classified as ellipsoid, irregular or column. We have to encode these strings so model can make sense of them. We used a encoding method called One hot encoding which denotes the binary state of each class, "1" if it is present or true, "0" otherwise.

4. Experimental Results

Our object detection model is Faster R-CNN Inception v2 Coco. This model helps us to bound the object and classifying. We trained our model with 2558 images and tested with 312 images. Faster R-CNN creates almost perfect bounding boxes for food and coin. We use these bounding boxes' numerical form on volume estimation.



Number of used images for training and testing on volume estimation

4.1. Dataset Description

We have 2870 images in our dataset.

4.1.1 Object Detection

We split 2870 images for object training and testing. Our used dataset is not big. Therefore we used 2558 images for training and 312 images for object detection testing models.

4.1.2 Volume Estimation

After we got high accuracy on finding coin and food in given images, we saved our object detection model. Then we obtain numerical form of images with using saved model. We added extra knowledge from our data set such as real volume, real density, per calorie in gram and so on. These numerical form names are in column of Figure 3. We used 2002 images' numerical form for training and 862 images' numerical form for testing in our estimation methods.

4.2. K Nearest Neighbors(KNN)

We used all the images' processed numerical form and then when we determine best K. Finding the value of best K is not easy. We try from 1 to 100 for K value. We calculate RMSE(Root Mean Square Error) for every value of K. Then we compared these RMSEs we calculated. We found the value of K that has minimum RMSE value. The best value of K is 7 for K nearest neighbor method. RMSE gives us how similar, on average, are two foods. Minimum RMSE is the most similar food. Therefore when we estimate the volume of food we select nearest 7 foods that are numerical the most similar foods. Our estimating volume is the average value of these 7 foods real volume.

When we saw these results, it surprised us because the results are better than we expect.

4.3. Volume Estimation

Total mean volume error is **21.06**. Our volume error result is in Table 1 for every class of food. Most of the foods' estimated volumes are relatively good. Some food not such as kiwi, moon cake and mango. This high error is probably due to our inability to receive good data from GrabCut.

Food Type	Estimation Food Count	Mean Volume	Mean Estimation Volume	Volume Error (%)
apple	38	321.32	327.03	1.78
egg	12	51.67	52.26	1.15
lemon	15	96.67	97.62	0.99
orange	28	222.14	226.89	2.14
peach	14	102.14	100.61	-1.5
plum	31	111.29	110.55	-0.66
kiwi	9	160.0	127.94	-20.04
tomato	21	183.81	182.79	-0.56
bread	8	150.0	143.93	-4.05
grape	13	304.62	304.29	-0.11
mooncake	19	52.11	46.99	-9.81
sachima	17	152.35	152.61	0.17
banana	17	174.71	174.03	-0.38
bun	11	236.36	224.55	-5.0
doughnut	26	198.08	197.97	-0.06
fired dough twist	13	70.0	70.22	0.31
litchi	11	42.73	40.65	-4.86
mango	34	100.88	109.54	8.58
pear	18	242.22	251.9	4.0

Table 1. Volume Estimation Error Results

4.4. Calorie Estimation

We calculate total calories using estimated volume in mathematical calorie calculation formulas.

$$C = c \times \rho \times v$$

C will estimate calorie where c is calories per gram, ρ is average density of food and v is our estimated volume.

Total mean calorie error is **45.79**. Our calorie error result is in Table 2 for each different food.

Food Type	Estimation Food Count	Mean Calorie	Mean Estimation Calorie	Calorie Error (%)
apple	38	131.79	134.24	1.86
egg	12	86.96	87.06	0.12
lemon	15	27.29	27.38	0.32
orange	28	129.12	131.8	2.08
peach	14	60.05	59.4	-1.09
plum	31	51.73	51.38	-0.66
kiwi	9	98.46	78.1	-20.68
tomato	21	49.06	49.85	1.6
bread	8	89.66	88.87	-0.88
grape	13	199.97	199.06	-0.45
mooncake	19	982.53	971.82	-1.09
sachima	17	711.76	712.62	0.12
banana	17	137.18	141.38	3.06
bun	11	185.8	170.52	-8.22
doughnut	26	259.63	272.14	4.82
fired dough twist	13	1007.84	1057.79	4.96
litchi	11	28.19	26.76	-5.09
mango	34	66.86	73.78	10.35
pear	18	90.59	93.89	3.64

Table 2. Calorie Estimation Error Results

4.5. Random Forest

We used this model for both estimating the volume and calories. The first approach was estimating the volume and multiplying this estimated volume with food's density and kcal (energy per grams). However, this approach scaled up with volume error linearly so we tried a second approach. With same set of features, we tried to estimate the total calories directly which resulted in a much better mean error value. We are going to discuss both of these approaches in the sections below.

4.5.1 Volume Estimation

We can see the mean error of volume estimation per each class in the figure below.

Food Type	Estimation Image Number	Mean Volume	Mean Estimation Volume	Volume Error
peach	14	108.57	119.57	-11.00
egg	15	51.33	52.15	-0.81
sachima	21	150.48	150.14	0.33
apple	33	335.15	330.71	4.44
mango	31	100.00	106.40	-6.40
lemon	17	95.29	106.12	-10.83
doughnut	35	188.29	188.60	-0.31
plum	32	114.06	115.16	-1.10
qiwi	11	150.91	146.73	4.18
bun	6	226.67	228.90	-2.24
tomato	20	181.50	177.51	3.99
orange	24	240.42	233.22	7.19
litchi	13	42.31	42.20	0.11
mooncake	21	49.05	50.11	-1.06
pear	17	249.41	247.27	2.14
banana	16	163.75	167.91	-4.16
fired dough twist	15	71.33	73.71	-2.37
grape	8	371.25	368.35	2.90
bread	6	148.33	149.71	-1.38

Table 3. Volume Error for Random Forest

As we can see some classes' error is close to 0 whereas others are relatively less accurate. This is caused by irregular distribution in our data set. Also, an error is directly affected by the accuracy of the bounding boxes and the performance of object detection model. In addition, some of the foods like orange and lemon are very similar to each other. Some undergrowth oranges' color is similar to lemons which cause object detection model to misinterpret the food.

4.5.2 Calorie Estimation

We obtain the total calories using the estimated volume. If we multiply the volume by the food's density it will give a estimation about its mass. Lastly, we multiply this mass with kcals (Joule per grams) to obtain the total calories. We can see the performance of the model in the Table 4.

We noticed that some of the classes have higher error than others like grape and fired dough wist. This is caused by the irregularity of our data set and the complex structure of the food.

Our second method for calorie estimation was directly estimating the calories with same set of features. This approach reduces the root mean square error **32.52** to **30.37**.

Food Type	Estimation Image Number	Mean Calorie	Mean Estimation Calorie	Calorie Error
peach	14	62.94	69.84	-6.90
egg	15	86.13	86.96	-0.83
sachima	21	707.94	698.27	9.66
apple	33	133.42	132.63	0.79
mango	31	65.40	71.01	-5.61
lemon	17	27.33	30.06	-2.74
doughnut	35	256.15	254.91	1.24
plum	32	52.87	53.60	-0.72
qiwi	11	87.27	85.80	1.47
bun	6	181.49	176.19	5.29
tomato	20	48.59	48.57	0.02
orange	24	130.49	130.15	0.34
litchi	13	27.73	27.99	-0.26
mooncake	21	1002.95	1002.85	0.10
pear	17	91.64	92.69	-1.05
banana	16	136.80	136.86	-0.06
fired dought wist	15	1000.81	1044.22	-43.41
grape	8	216.70	248.67	-31.97
bread	6	90.36	92.03	-1.67

Table 4. Calorie Error for Random Forest

Food Type	Mean Volume	Random Forest Mean Estimation Volume	KNN Mean Estimation Volume	Baseline Mean Volume	Baseline Mean Estimation Volume
peach	108.57	119.39	100.61	110.65	115.52
egg	51.33	52.23	52.26	52.94	62.13
sachima	150.48	149.87	152.61	147.29	129.05
apple	335.15	330.67	327.03	333.64	270.66
mango	100.00	106.27	109.54	81.67	70.43
lemon	95.29	106.01	97.92	96.79	94.03
doughnut	188.29	188.55	197.97	174.75	143.47
plum	114.06	115.18	110.55	100	98.35
kiwi	150.91	147.05	127.94	127.14	123.08
bun	226.67	229.25	224.55	245.36	235.39
tomato	181.50	177.50	182.79	174.22	162.28
orange	240.42	233.10	226.89	234.42	235.93
litchi	42.31	42.23	40.65	43.33	49.25
mooncake	49.05	50.07	46.99	67.58	52.52
pear	249.41	247.18	251.9	260.00	225.93
banana	163.75	167.96	174.03	162.0	204.16
fired dough twist	71.33	73.48	70.22	65.0	54.50
grape	371.25	369.31	304.29	240.0	323.57
bread	148.33	149.74	143.93	155.0	180.62

Table 5. Comparison with Baseline Work

5. Conclusions

We are going to conclude our work comparing our results with the baseline work in terms of volume estimation since the results for calorie estimations are not shared in the paper.

If we use 30% for testing Random Forest model is the most optimal model for our problem with the mean error of **13.12** whereas KNN has a mean error of **21.06**. We can see the comparison between baseline results and both of our models in Table 5.

As we can see both of our models outperforms the baseline work. Since volumes calculated with math formulas in our baseline work, some fruit that are close to perfect shape like lemon has higher accuracy than our models, however this is only limited to foods with ellipsoid shapes.

When we compare our volume estimation methods, we can see in Table 6 that random forest model is slightly outperforming the kNN model.

The reason why the K Nearest Neighbors method is as good as the Random Forest method is that the data set and the number of features that we use is small and thus the model doesn't suffer from curse of dimensionality.

Since our data set is relatively small we can expand our data set which will decrease the error even more as a future work. Also removing the imbalanced distribution in our data set will decrease the error as well. However due to uniqueness of some food types we might need more images that are taken from different angles and distances for specific food types. So we can say that hand adjusted imbalance might actually help our models to learn better.

Methods Name	Volume Mean Error	Calorie Mean Error
K Nearest Neighbors	21.06	45.79
Random Forest	13.21	30.37

Table 6. Comparing Volume Estimation Methods

References

[1] A. B. Carsten Rother, Vladimir Kolmogorov. grabcut interactive foreground extraction using iterated graph cuts.

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

[3] Z. Lu. Tensorflow object detection api.

[4] Z. Lu. Tensorflow object detection api models.

[5] S. E. Manal Chokr. Calories prediction from food images.

[6] P. Pouladzadeh, S. Shirmohammadi, and R. Al-Maghrabi. Measuring calorie and nutrition from food image. *Instrumentation and Measurement, IEEE Transactions on*, 63:1947–1956, 08 2014.

[7] M. B. Sadegh Bafandeh Imandoust. Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background.

[8] J. L. Yanchao Liang. Computer vision-based food calorie estimation: Dataset, method, and experiment.

[9] J. L. Yanchao Liang. Computer vision-based food calorie estimation: Dataset, method, and experiment.