FOOD CALORIE ESTIMATION USING FASTER R-CNN

author
Ayça KULA

1 INTRODUCTION

Obesity is an increasing public health problem in many countries in the world (WHO). In order to stop the obesity epidemic, people must control their daily calorie intake and eat more of the healthier foods. Moreover, many people are interested in following what they eat to help them lose weight or manage their diets. The significant reason behind this problem roots from not being able to estimate the calories of their food. In this paper, we propose a work that will help people plan their diet easier by estimating food calories.

2 RELATED WORK

There are many similar studies have been conducted to serve a similar purpose. An image-based method was proposed which has used Pin-hole modelling; algorithms to determine the focal length of the camera, the distance between a given pair of points, and a certain surface area on the food (Sun et al., 2008).In this study, the use of the use of a calibration card is necessary when the food picture is taken. In addition, the calibration object may be the user's thumb on a suitable position on the dish. Beside image processing algorithms SVM was used to feed food portion characteristics such as color, size and shape into the classification procedure (Villalobos et al., 2012). Moreover, Meyers et al. (2015) uses deep learning algorithmns in order to recognize foods. CNN-based approaches were done to run the algorithms in a conventional mobile phone (Meyers et al., 2015). This paper was based Liang & Li (2017)'s work which have used Faster Region-based Convolutional Neural Networks (Faster R-CNN) to detect objects, and GrabCut algorithm as the segmentation algorithm. The authors chose Faster R-CNN rather than using semantic segmentation such as Fully Convolutional Networks (FCN). The calibration object, which is a coin, is used to estimate images scale factor (Liang & Li, 2017).

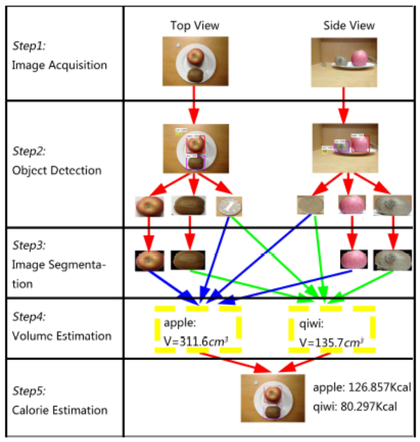


Figure 1: Food Calorie Estimation (Liang & Li, 2017).

3 METHOD

Briefly, our method for food analysis includes the following steps:1) Image acquisition, 2) Object detection, 3) Image segmentation, 4) Volume estimation, 5) Calorie estimation. The steps of the progress are shown clearly in figure 1.

3.1 OBJECT DETECTION

Standard convolutional neural network could not be used, since the length of the output layer is not constant due to the varying number of objects of interests (Gandhi, 2018). Therefore, many methods such as R-CNN, Fast R-CNN, Faster R-CNN, YOLO etc. have been developed for object detection. R-CNN training, which has a multi-stage pipeline, is expensive in both space and time (Girshick, 2015). The main significant problem of an R-CNN model is that it independently

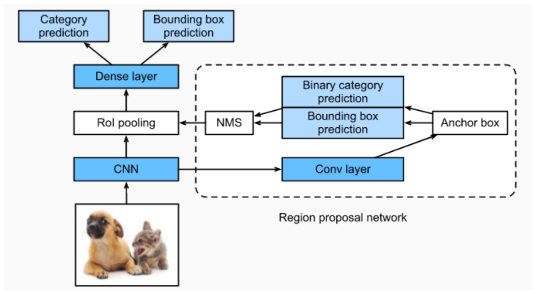
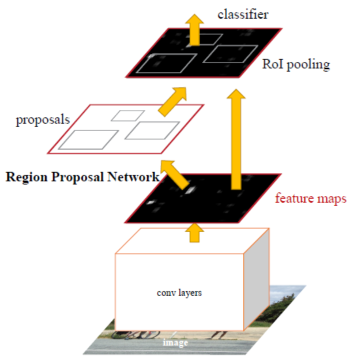


Figure 2: Faster R-CNN model (Aston Zhang, 2021).

extracts features for each proposed region which will yield to increased volume of repeated computations (Aston Zhang, 2021).Then, fast R-CNN, which is a new improved training algorithm, was proposed to fix the disadvantages of the R-CNN. Compared with R-CNN the speed and accuracy was improved (Girshick, 2015). However, both of these algorithms use selective search to find out the region proposals. Selective search is an algorithm which combines both the exhaustive search and segmentation algorithms (Uijlings et al., 2013). Ren et al. (2015) came up with faster R-CNN algorithm that replaces selective search with a region proposal network and lets the network learn the region proposals. Therefore, faster R-CNN approach will be used to solve the object detection problem.

3.1.1 Faster R-CNN

Faster R-CNN is the combination of two modules. First one is the deep fully convolutional network that proposes regions and the second is the fast R-CNN detector (Ren et al., 2015) as seen in figure 2. The Faster R-CNN architecture is a single, unified network as shown in figure 3.



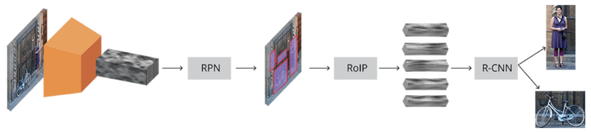


Figure 3: LEFT: Unified Network of Faster R-CNN (Ren et al., 2015).RIGHT: The different layers of Faster R-CNN (ale).

Region Proposal Network (RPN) Region Proposal Network (RPN) is a fully convolutional network, which takes an image of any dimension as input and it outputs a set of rectangular object proposals each with an objectness score (Ren et al., 2015). Objectness score defines how likely the region is to contain an object. The “grid” shown in figure 4, is the convolutional feature map which is the input of RPN and we convolve the input feature layer with an n x n sliding window (kernel). At each sliding window position, we generate k anchor boxes with different sizes and aspect ratios which are centered at the center of the sliding window.

Anchors Anchors, as seen in figure 5, are fixed bounding boxes that are placed throughout the image with different sizes and ratios used for training the Region Proposal Network (RPN) (MADALI, 2020). They are going to be used as a reference when first predicting object locations. There are 2k classification scores and 4k regression coordinates for k anchor boxes at each sliding window location as seen from figure 4. The classification is simply “object or no object” and this classification layer is implemented as a two-class SoftMax layer. The 4k outputs are the coordinates of the box. For a convolutional feature map with width W, height H and size WxH, there are WxHxk anchors in total.

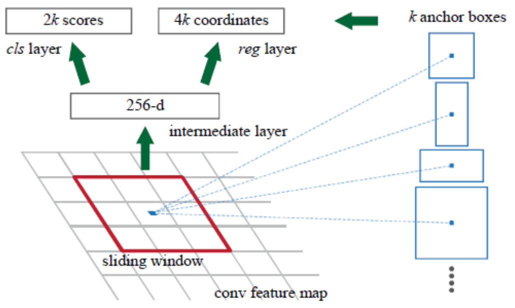


Figure 4: Region Proposal Network(RPN) (Ren et al., 2015).

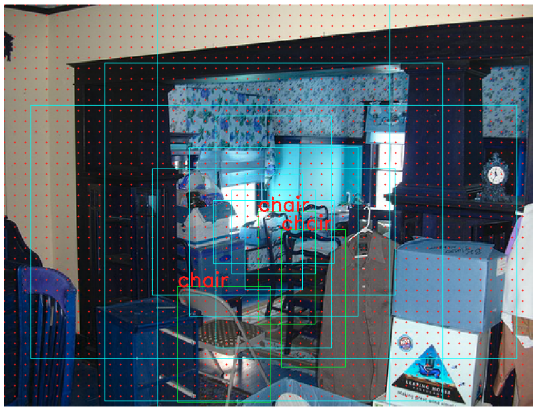


Figure 5: Anchors (red dots), 9 out of thousands of anchor boxes (light blue), and ground truth boxes (green) overlaid on the original image (Goswami, 2018).

Training RPNs and Loss Functions First of all, Non-Maximum Suppression (NMS) that removes boxes that overlaps with other boxes that has higher scores is used for reduction operation. For training, we take all the anchors and put them into two different categories as foreground and background. If "foreground" or "positive samples", anchors overlap a ground-truth object with an Intersection over Union (IoU) bigger than 0.7. But if, IoU is smaller than 0.3 is considered as "background" or "negative sample" (Konushin). Then these IOUs are used to label the ROIs (RPN classification). These labels are then used to calculate the cross-entropy loss. Also, the RPN tries to tighten the center and the size of the anchor boxes around the target which is called the bounding box regression. But for this to happen, targets need to be obtained and losses are needed to be calculated for back propagation. RPN loss function for an image is the sum of classification loss and bounding box regression loss called the Smooth L1 Loss.

$$L\left(\{p_i\},\{t_i\}\right)=\frac{1}{N_{cls}}\sum_i L_{cls}\left(p_i,\ p_i^*\right)+\lambda\frac{1}{N_{reg}}\sum_i p_i^*\ L_{reg}\left(t_i,\ t_i^*\right) \tag{1}$$

For bounding box regression, we define the equations given below:

$$t_x=(x-x_a)/\omega_a, t_y=(y-y_a)/h_a, t_w=\log(\omega/\omega_a), t_h=\log(h/h_a) \tag{2}$$

$$t_x^*=(x^*-x_a)/\omega_a, t_y^*=(y^*-y_a)/h_a, t_w^*=\log(\omega^*/\omega_a), t_h^*=\log(h^*/h_a) \tag{3}$$

Training the Faster R-CNN Joint training which means that training RPN, Fast-RCNN as one network is faster for faster R-CNN. This one network consists of four losses for RPN classification (good/bad anchor), RPN regression (anchor-proposal), Fast R-CNN classification (over classes), Fast R-CNN regression (proposal-box). Changing from VGG-16 to ResNet-101 model on Microsoft COCO dataset will give much more gain as seen from table 1.

Table 1: Detection results on Microsoft COCO dataset (Konushin)

Faster R-CNN Baseline	mAP@.5	mAP@.5:.0.95
VGG-16	41.5	21.5
ResNet-101	48.4	27.2

3.2 Image Segmentation

After detecting the object,it has to be accurately segmented from the background using the Grab-Cut algorithm.In order to complete foreground extraction, a bounding rectangle alone is sufficient (Rother et al., 2004).

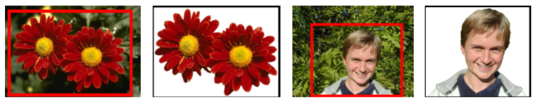


Figure 6: Examples of Grabcut (Rother et al., 2004).

3.3 Volume Estimation

First of all scale factors has to be calculated.For the side view it is calculated as in Equation 4.

$$\alpha_S = \frac{2.5}{(W_S + H_S)/2} \tag{4}$$

Also,for the top view it is calculated as in Equation 5.

$$\alpha_T = \frac{2.5}{(W_T + H_T)/2} \tag{5}$$

If dataset is taken into account, there are different types of shapes of objects. Therefore, volume must be calculated in reference with Equation 6 regarding these shapes.

$$v = \begin{cases} \beta \times \frac{\pi}{4} \times \sum_{k=1}^{H_S} \left(L_S^k\right)^2 \times \alpha_S^3 & \text{if} \quad \text{the shape is ellipsoid} \\ \beta \times \left(s_T \times \alpha_T^2\right) \times \left(H_S \times \alpha_S\right) & \text{if} \quad \text{the shape is column} \\ \beta \times \frac{\pi}{3/4} \times \alpha_S^3 & \text{if} \quad \text{the shape is sphere} \end{cases} \tag{6}$$

3.4 Calorie Estimation

The simple equations are used after the getting the food volume.The mass for each food is obtained with Equation 7.

$$m = \rho \times v \tag{7}$$

Then, the calorie of the food is calculated with Equation 8.

$$C = c \times m \tag{8}$$

4 Experimental Settings

The ECUSTFD dataset (Liang-yc) which consists of 19 types of food has been used.There are total of 2978 images.These images contain both the top and side views of each food and there exists a calibration object(Yuan coin) in every picture.80 % of the dataset has been used for training and the rest for testing. In order to easily train, deploy and obtain object detection models with an open source framework the TensorFlow Object Detection API has been used.Resnet-101 has been used as a pretrained model.In order to calculate the accuracy/performance of the object detection approach IoU has been used.And the final results are compared with the previous work done.

5 Experimental Results

In order to run the faster R-CNN model, it was importnat to choose the appropriate hyper parameters such as learning rate, epoch, batch size. As the number of epochs increases, more number of times the weight are changed in the neural network and the curve goes from underfitting to optimal to overfitting curve (Sharma, 2019).Therefore, the epoch is chosen higher than 1. The learning rate is selected as approximately 0.001. As you can see from figure 7, initially the learning rate is higher and then learning rate becomes more smaller by the shorter size of steps.

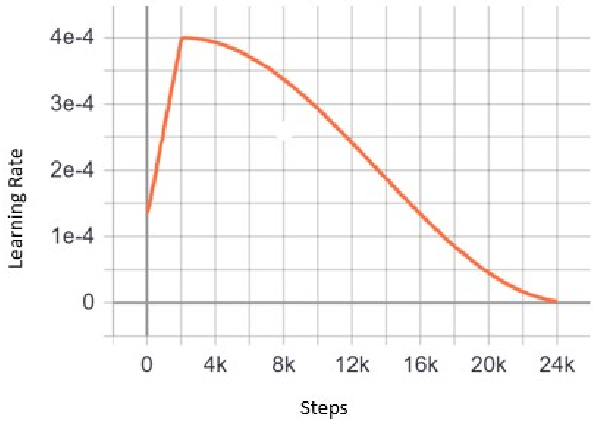


Figure 7: Learning rate with respect to training steps

Moreover, when using GPUs, it is common for power of 2 batch sizes to offer better runtime. Typical power of 2 batch sizes range from 32 to 256, with 16 sometimes being attempted for large models (Goodfellow et al., 2017).The presented results confirm that using small batch sizes achieves the best training stability and generalization performance, for a given computational cost, across a wide range of experiments. In all cases the best results have been obtained with batch sizes m = 32 or smaller (Masters & Luschi, 2018).In theory, this hyper-parameter should impact training time and not so much test performance, so it can be optimized separately of the other hyperparameters, by comparing training curves (training and validation error vs amount of training time), after the other hyper-parameters (except learning rate) have been selected.B = 32 is a good default value for starting the training (Bengio, 2012). Therefore, the batch size is decreased to 2.

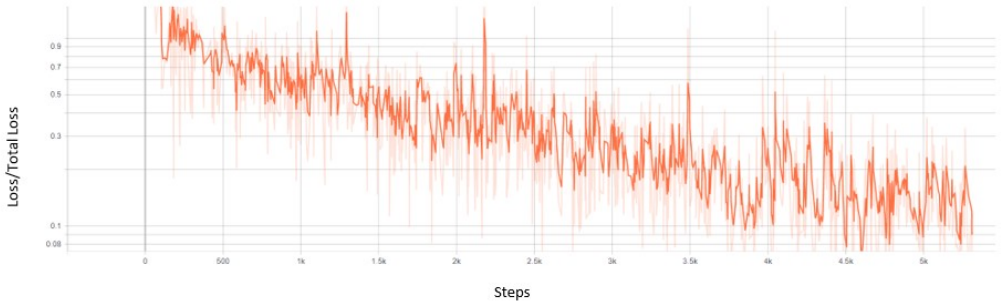


Figure 8: Loss function with respect to training steps

In figure 8, the loss function decrements can be seen. The aim was to decrease our loss function while training the network. Therefore, this procedure has been successful.After running the whole model the bounding boxes have been found as seen from figure 9.

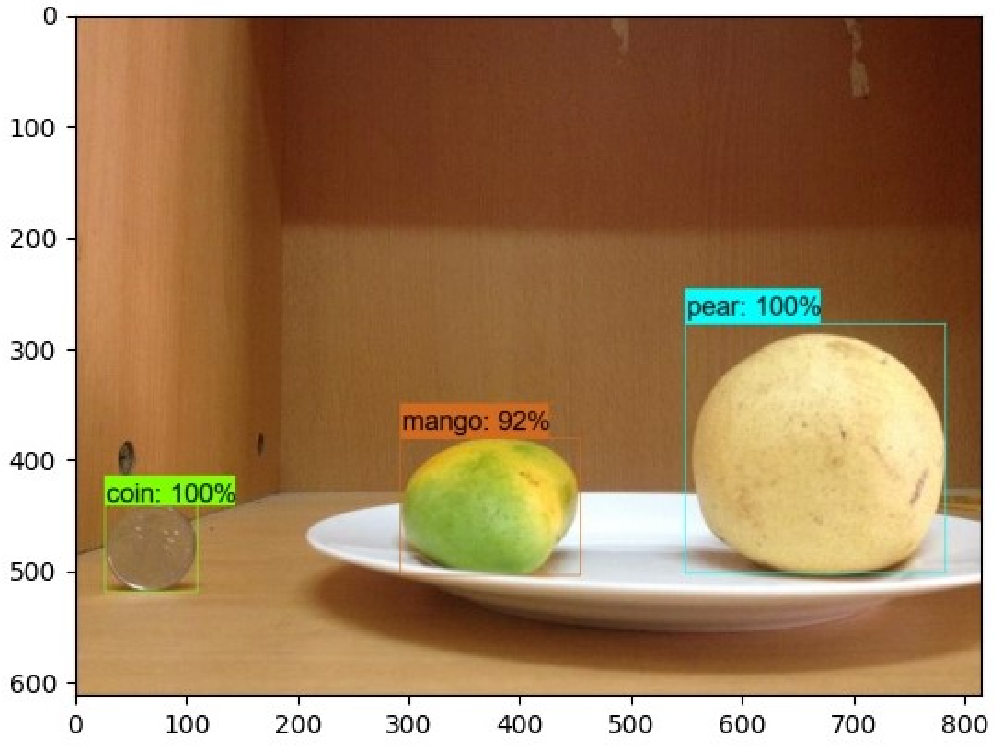


Figure 9: Running the Faster R-CNN Model

After, obtaining our Resnet-101 Faster R-CNN method. We have segmented the images using Grab-cut algorithm. The mango and the pear are segmented from the image which is shown in figure 9. The detection accuracy's are shown in the figure.

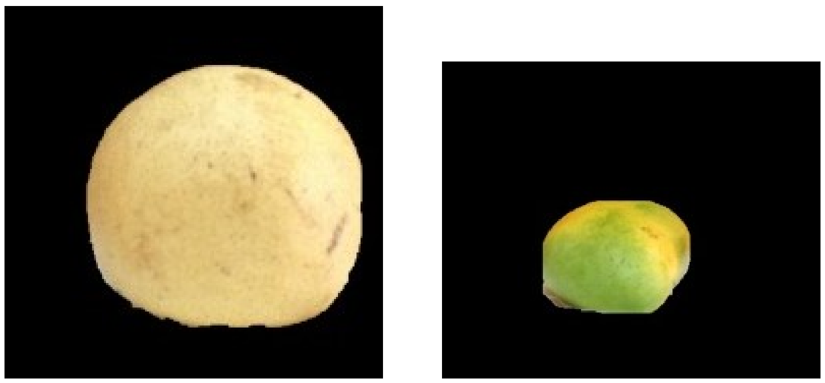


Figure 10: Detected pear and mango are segmented by Grab-cut algorithm.

Table 2: Food types and the estimated percentage volume errors

Fruits	**Shape**	**Mean Percentage Volume Error**	**Min Percentage Volume Error**	**Max Percentage Volume Error**
apple	sphere	19.14	1.13	67.41
pear	sphere	16.21	12.61	23.55
litchi	sphere	18.21	5.93	37.24
orange	sphere	14.37	3.74	35.82
tomato	ellipsoid	42.65	32.04	50.95
egg	ellipsoid	35.85	2.31	81.70
plum	ellipsoid	20.88	15.19	26.11
kiwi	ellipsoid	24.49	1.43	46.08
lemon	ellipsoid	21	7.54	36.81
mango	ellipsoid	34.18	7.89	66.91
peach	ellipsoid	23.51	13.07	40.27
sachima	column	33.77	10.39	59.84
doughnut	column	64.98	44.71	135.07
bread	column	136.53	80.18	225.44
bun	column	78.03	59.60	108.64
mooncake	column	24.37	5.59	71.23
fired dough nut	column	106.86	2.50	187.06
grape	column	255	255	255
banana	column	144.46	23.68	254.41

6 Discussions/Conclusions

The final estimated volume percentage errors can be seen from table 2. As it can seen from the table, the error increases as the actual shape tries to fit in a column shape and this is a normal result. For example, fitting fired dough nut as a column does not give good results. Therefore, volume of shapes has to be calculated in more detail for the best result. However, the bread volume error percentage should have resulted in a small error since the column is a right fit for the bread's shape. This situation must be examined further.

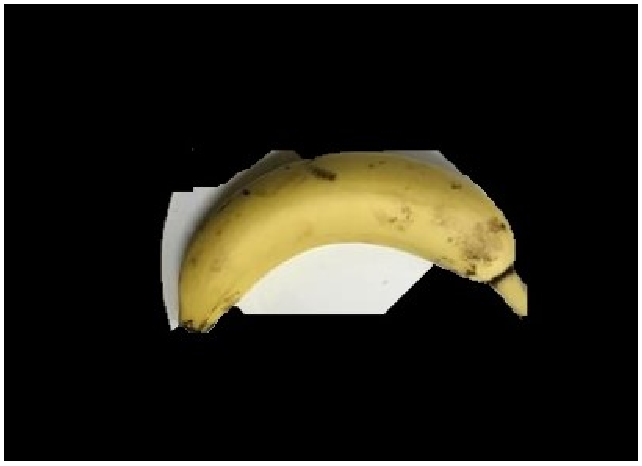


Figure 11: Applying Grab-cut segmentation algorithm for banana.

Moreover, there were problems occured when the grab-cut algorithm was used. At some of the images, the grab-cut algorithm could not been applied.In some of coin images, coin had disappeared from the image. Moreover, in the given banana figure in 11, the segmentation looks problematic.

Table 3: Percentage volume errors comapared with the paper

Fruits	Shape	Mean Percentage Volume Error	Mean Percentage Volume Error (Paper)
apple	sphere	19.14	-11.59
pear	sphere	16.21	-21.42
litchi	sphere	18.21	-26.47
orange	sphere	14.37	2.8
tomato	ellipsoid	42.65	5.36
egg	ellipsoid	35.85	17.56
plum	ellipsoid	20.88	4.78
kiwi	ellipsoid	24.49	-38.86
lemon	ellipsoid	21	3.88
mango	ellipsoid	34.18	-6.05
peach	ellipsoid	23.51	4.35
sachima	column	33.77	-16.57
doughnut	column	64.98	10.06
bread	column	136.53	6.46
bun	column	78.03	-10.05
mooncake	column	24.37	20.22
fired dough nut	column	106.86	8.65
grape	column	255	-3.28
banana	column	144.46	17.11

Arranging the parameters of the function did not helped solving this problem. This problem is especially occured at blurred images.

When we compare this work with the main paper that has been used to start this project (Liang & Li, 2017), volume errors are minimum 60 percent more for the column images.From table 3 the difference can be seen.

7 Default Notation

The notations that were used in the paper are given below.

Indexing

i	Index of an anchor in a mini-batch
p_i	Predicted probability of anchor i being an object
p_i^*	The ground-truth label where the value is 1 if the anchor is positive, and is 0 if the anchor is negative
t_i	Vector representing the 4 parameterized coordinates of the predicted bounding box
t_i^*	The ground-truth box associated with a positive anchor

Parameters

x, y	Box center coordinates
w, h	Box width and height,respectively
x	Predicated box
x_a	Anchor box
x^*	Ground truth box
W_S	Width of the bounding box from side view
H_S	Height of the bounding box from side view (also number of rows for the side viewed image)
W_T	Width of the bounding box from top view
H_T	Height of the bounding box from top view
α_S	Side view's scale factor
α_T	Top view's scale factor
P_S	Images in side view
P_T	Images in top view
s_T	Number of foreground pixels in top view
L_S^k	Number of foreground pixels in row k$\in \{1, 2, 3, \ldots, H_S\}$
L_T^k	Number of foreground pixels in row k$\in \{1, 2, 3, \ldots, H_T\}$
L_S^{MAX}	Records the number of foreground pixels in P_S
β	Compensation factor
m	Mass(g)
v	Volume(cm^3)
ρ	Density value(g / cm^3)
c	Calories per gram(Kcal / g)

Functions

L_{cls}	Classification log loss over two classes (object vs. not object)
L_{reg}	Regression loss ($L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$)
R	Robust loss function (Smooth L_1)

REFERENCES

Faster r-cnn. URL https://www.alegion.com/faster-r-cnn.

Mu Li Alex J. Smola Aston Zhang, Zack C. Lipton. *Dive into Deep Learning*. 2021.

Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *Lecture Notes in Computer Science Neural Networks: Tricks of the Trade*, pp. 437–478, 2012. doi: 10.1007/978-3-642-35289-8_26.

Rohith Gandhi. R-cnn, fast r-cnn, faster r-cnn, yolo - object detection algorithms, Jul 2018. URL https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2017.

Subrata Goswami. A deeper look at how faster-rcnn works, Jul 2018. URL `https://whatdhack.medium.com/a-deeper-look-at-how-faster-rcnn-works-84081284e1cd#:~:text=Attheconceptuallevel,Faster,afewlast/toplayers`.

Anton Konushin. Faster r-cnn - object detection. URL `https://www.coursera.org/lecture/deep-learning-in-computer-vision/faster-r-cnn-DM0hz?utm_source=gg&utm_medium=sem&utm_content=01-CatalogDSA-ML1-US&campaignid=12490862811&adgroupid=119269357576&device=c&keyword=&matchtype=b&network=g&devicemodel=&adpostion`.

Yanchao Liang and Jianhua Li. Deep learning-based food calorie estimation method in dietary assessment. 06 2017.

Liang-yc. Liang-yc/ecustfd-resized-. URL `https://github.com/Liang-yc/ECUSTFD-resized-`.

Nabil MADALI. Demystifying region proposal network (rpn), May 2020. URL `https://medium.com/@nabil.madali/demystifying-region-proposal-network-rpn-faa5a8fb8fce`.

Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. 04 2018.

Austin Meyers, Nick Johnston, Vivek Rathod, Anoop Korattikara, Alex Gorban, Nathan Silberman, Sergio Guadarrama, George Papandreou, Jonathan Huang, and Kevin P Murphy. Im2calories: towards an automated mobile vision food diary. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1233–1241, 2015.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 06 2015. doi: 10.1109/TPAMI.2016.2577031.

Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. ” grabcut” interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.

Sagar Sharma. Epoch vs batch size vs iterations, Mar 2019. URL `https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9`.

Mingui Sun, Qiang Liu, Karel Schmidt, Ning Yao, John Fernstrom, Madelyn Fernstrom, James Delany, and Robert Sclabassi. Determination of food portion size by image processing. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2008:871–4, 02 2008.

Jasper Uijlings, K. Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 09 2013. doi: 10.1007/s11263-013-0620-5.

Gregorio Villalobos, Rana Almaghrabi, Parisa Pouladzadeh, and Shervin Shirmohammadi. An image processing approach for calorie intake measurement. 05 2012. doi: 10.1109/MeMeA.2012.6226636.

WHO. Obesity and overweight. URL `https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight#:~:text=Oftheseover650millionadultswereobese.,tripledbetween1975and2016`.