**FOOD CALORIE ESTIMATION USING  
 INTRINSIC SEGMENTATION MODELS**

Darsh Kachroo1

1. Vellore Institute of Technology, SENSE

***Abstract:***

*This study introduces a novel system for estimating the calorie content of food using a combination of deep learning and a comprehensive food database. Our work takes inspiration from existing models like ResNet50, YOLOv3, ImageNet, and more, to create new building blocks for new models. We put our blocks to the test by incorporating them into models for the prediction of Food Calories and Food Volume from the given image.*

*By leveraging data from a calorie database, the system is able to accurately predict the total number of calories in a given image based on the calculated volumes and labels. This approach effectively tackles common challenges such as varied images and lighting conditions, thanks to its robust segmentation and database-driven prediction methods.*

*An extensive evaluation, including measuring segmentation accuracy, volume estimation error, and overall calorie estimation error, is planned. This innovative system has the potential to greatly assist individuals in monitoring their dietary intake by providing accurate calorie estimates from food images.*

***Keywords: calorie estimation, image recognition, image segmentation***

**1. Introduction:**

Having a healthy diet is vital for overall well-being, but accurately keeping track of calorie consumption can be difficult. Despite the existence of calorie labels, manual tracking can be tedious and prone to mistakes, resulting in underestimated intake. As food photography and smartphone cameras become more popular, the opportunity arises to utilize technology for automated calorie estimation, giving individuals the power to make informed dietary decisions.

**1.1.** **The Importance of Automated Calorie Estimation:**

Precise calorie tracking is crucial for weight management, preventing chronic illnesses, and improving overall health. Nevertheless, traditional methods such as manual logging or calorie-counting apps have their limitations.

**1.2. Challenges and the Gap:**

Creating an effective automated calorie estimation system comes with its unique set of obstacles. The varying appearance of food images caused by lighting, camera angles, and presentation poses a challenge in accurately identifying and segmenting food items. Furthermore, accurately estimating portion sizes proves to be a daunting task due to the diverse physical characteristics of different foods. Despite advancements in computer vision and deep learning technologies, there remains a lack of readily available solutions to effectively address these challenges, leaving a significant gap in the path towards healthy eating through convenient calorie tracking.

**1.3.** **The Promise of Deep Learning and Segmentation:**

This research aims to bridge this gap by leveraging the power of deep learning and image segmentation techniques. By developing a model that accurately identifies and segments individual food items within an image, we can estimate their volumes and subsequently, their calorie content. This approach eliminates the need for manual portion size estimation, offering a more objective and convenient method for individuals to track their calorie intake.

**1.4. Significance and Impact:**

The successful development of such a system can significantly impact individual and public health. By empowering individuals to make informed dietary choices through accurate calorie tracking, this technology has the potential to:

* Promote weight management and reduce obesity rates.
* Prevent diet-related chronic diseases like diabetes and heart disease.
* Encourage healthier eating habits and mindful food choices.
* Facilitate personalized dietary interventions and nutritional guidance.

**2. Literature Review:**

While software estimating food calories from pictures holds promise, no current options are accurate or user-friendly. Over 2000 people estimated calories for 20 diverse food pictures. Even experts only averaged 5 correct out of 20. Surprisingly, even small crowds (10 people) were more accurate than individuals or experts. Women and younger people did better, while people overestimated calorie-dense foods and struggled with pictures containing reference objects like credit cards. These findings show crowdsourcing could improve accuracy, but limitations remain. Understanding biases is crucial for better software and analysis. [4,5]

Other works proposed a layered deep learning approach for automated calorie estimation which included techniques like

Faster R-CNN, Canny edge detection, Grab Cut, and Deep learning models. [1]

Convolutional neural networks (CNN) were also used. The network, trained on labeled food images and calorie data, directly predicts the total calorie content of the food in the image.[2]

In the various studies conducted for food calorie prediction, two key concepts remained:

* People individually have poor judgment capabilities for calorie estimations due to personal biases and beliefs.
* Use of machine learning techniques gave promising results up-to a certain extent.

We can improve upon these key ideas by:

* Educating the mass about calorie intake and healthy lifestyle ways.
* Improving machine learning techniques by incorporating new features and better models.

**3. Dataset Details:**

The chosen dataset, IEEE’s FOODD, for this task comprises of images of 16 types of food items out of which we are using 7, namely Apple, Banana, Carrot, Cucumber, Onion, Orange and Tomato. These categories were taken for the creation of a Proof-of-Concept model as these categories were the simplest to predict in terms of calorie count.[3]

**3.1. Exploratory Data Analysis:**

The images corresponding to various food items are not labeled and hence we had to implement crude pre-processing to correctly obtain the calorie values.

It was observed that each of the images were comprising of 4 parts:

* White plate
* The food item
* The background
* The finger of the user holding the plate.

We implemented layers of OTSU thresholding, named after Nobuyuki Otsu, along with contouring to obtain the individual components. Further, we specify the size of the finger to scale each component. This allows us to have a measure of the relative size of the food item.

Using the class of the food item, we split the items via shape for the volume derivation via the relative pixel area. With the help of density of table for the common food items, we are able to generate approximate answers for the calories present in the food item. [7]

**4. Methodology:**

We are employing a variety of deep learning architecture based on convolutional neural networks. *Our process is unique as we attempt to train the model using the preprocessing function and test the accuracy based on extrapolation capabilities.*

The core of this pipeline lies in the following blocks:

**4.1. Feature Extraction Block (FEB):**

This block is specifically designed to extract informative features from the input image. It employs n convolutional layers with a 1x1 convolution encompassing them.

This architecture efficiently extracts low level features along with maintaining the input shape through the 1x1 convolution.

Here, the Input and Output shape remain the same through the use of padding before convolution.

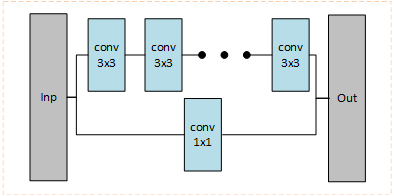


Fig 1- Feature Extraction Block

**4.2. Feature Localization Block (FLB):**

The FLB refines the features extracted from the preceding block. It employs pooling layers, followed by up sampling layers, to focus on the most critical regions within the extracted features. This step helps to localize the informative parts of the feature maps, potentially improving the model's ability to identify key aspects of the input image. This allows for better spatial localization of the features.

Here, the Input and Output shape remain the same as the pooling reduces the dimension and up sampling increases it.

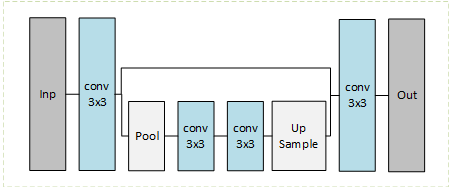


Fig 2- Feature Localization Block

**4.3. Feature Reduction Block (FRB):** The FRB aims to reduce the dimensionality of the features it obtains. This is achieved through dimensionality reduction layers like pooling, ensuring efficient processing by the subsequent layers.

This architecture represents the encoder half of an autoencoder, allowing this block to reduce the input features into a latent space for the subsequent blocks to use.

Here, the Input and Output are of different shapes because of the pooling layers used.

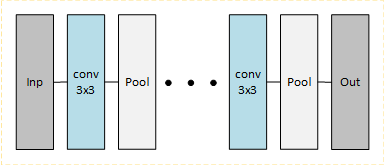


Fig 3- Feature Reduction Block

**4.4. Feature Distillation Block (FDB):** The FDB serves a crucial role. It leverages the knowledge learned throughout the network to create a compressed yet informative representation of the features. This is done by employing ‘n’ convolutions like in FEB but without padding and skip connections.

This archetype is observed in other FDB like Residual Feature Distillation Blocks, IMDB, and more [6]. However, they incorporate multiple overlaps which increases model complexity.

Our FDB reduces the complexity and allows the model to pass on the features found in previous blocks effectively.

Here, the Input and Output shapes don't remain the same as the convolutions are not padded.

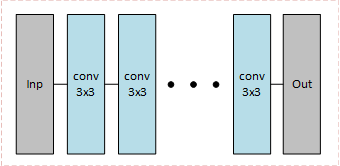


Fig 4- Feature Distillation Block

**4.5.** **Model Architecture**

The order of these blocks is crucial for optimal performance. Feature extraction blocks and feature localization blocks are employed at the beginning of the pipeline. These blocks are responsible for extracting raw features from the input data and refining their spatial information, respectively. Subsequently, feature reduction blocks and feature distillation blocks are placed.

FRBs aim to reduce the dimensionality of the features obtained from the FLBs, promoting computational efficiency for subsequent layers. FDBs leverage the knowledge accumulated throughout the network to create a compressed yet informative representation of the features.

This intrinsic order ensures a logical flow: initial feature extraction and localization followed by dimension reduction and knowledge distillation for a compact and informative feature vector feeding the prediction model.

An example architecture is shown as follows:

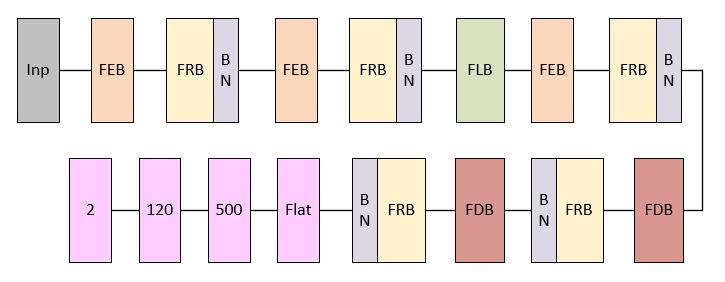


Fig 5- Example Architecture using custom blocks

**5. Data Analysis**

For our work, we needed to address a few challenges to accurately predict the food calories. First, we needed the dataset to be consistent in terms of object localization or the dataset be labeled. Since there was little to no record of labeled and segmented singular foods, we went with the FooDD dataset by IEEE [3]. Since the segmented masks were not provided, we had to develop a robust preprocessing step which leveraged the object localization for accurate mask generation.

Our approach to handle the FooDD dataset is as follows [7,8] :

1. Segmenting out the finger, plate, and the fruit as a combined contour.
2. Separate the plate from the finger and fruit.
3. Separate the finger and fruit mask.

This was possible as all the images were similarly taken with a hand/finger holding a white plate with a fruit on the plate.

This separation of the finger mask played a crucial role in establishing a scaling reference. The inherent two-dimensional nature of images eliminates depth perception, potentially causing fruits of varying sizes (e.g., watermelon vs. apple) to appear identical.

To address this challenge, our approach compared the relative size of the fruit to the finger, mitigating errors caused by solely relying on pixel-based measurements.

Further, the following assumptions were made about the fruit shapes:

1. **Circle:** Fruits like apple, orange, tomato, and onion were assumed to be perfect circles.
2. **Rod:** Fruits like carrot, cucumber, and banana were assumed to be a perfect cylinder.
3. **Disk:** Cut fruits like carrot and tomato were assumed to be a perfectly circular disk of 0.5cm uniform thickness.

These categories simplified our label generating process as compared to trying to accurately define the shapes.

Taking into consideration the projective nature of images, to calculate the volume of the fruit, we multiplied the area of the fruit by 2 as we can only see half of the fruit.

Subsequently, the volume can be found based on the established assumptions.

This plays a significant role in determining the complexity the predicting model should possess. Since the task requires calorie density, the model will first have to segregate the class of fruit and segment it to predict the calories. However, this is a limiting approach, and**, in our work, we have decided to skip these steps and allow the model to self-learn the features**.

Fig 6- Example Image of Apple



Fig 7- Example Image of Banana

After applying the preprocessing, we performed various visualizations to better understand how our data is distributed.

The steps involved were

1. Outlier Detection and Removal
2. Independence of Output Variables

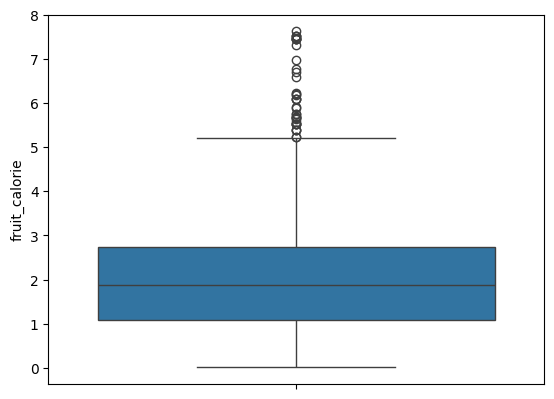


Fig10- Distribution of fruit calories

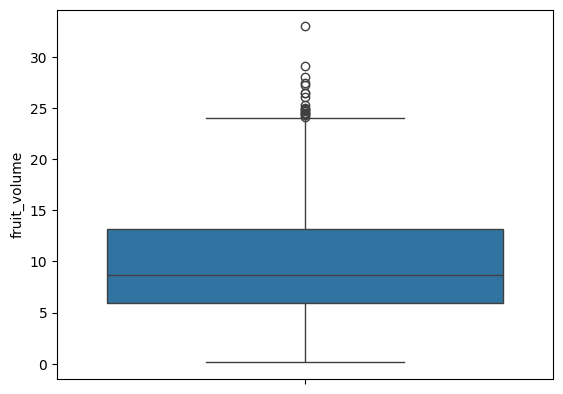


Fig 8- Distribution of fruit volume

|  |  |  |
| --- | --- | --- |
|  | **Fruit volume** | **Fruit calorie** |
| **Fruit volume** | 1.0 | 0.779156 |
| **Fruit calorie** | 0.779156 | 1.0 |

Table 1- Correlation matrix for labels

Though the output labels are highly correlated, we should not take this at face value as on plotting these variables with each other we find that the relationship between these isn’t linear in the classical sense.

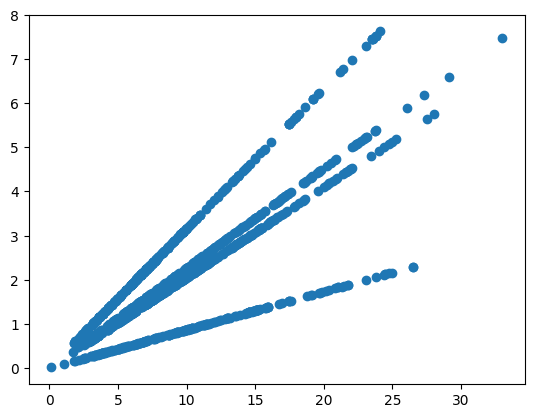


Fig 9- Fruit Volume v Fruit Calorie Scatterplot

It is observed that 4 distinct lines are present. This is because of the classes which we have taken in assumptions. Each represents one ‘type’ of fruit in terms of the shape. In essence, the data encodes the different classes of fruits.

**6. Results**

Vision Transformer’s hyperparameters were optimized to yield the best validation mean absolute error with the help of hyperband optimizer.

* **Patch size** indicates that the input image is divided into patches of the specific size.
* **Projection dimension** determines the dimensionality of the patch embeddings. **Transformer encoder** employs layers, each with attention heads.
* **Transformer units** and **MLP head units** control the complexity of the model's internal representations.

The optimized model was found with the following hyperparameters:

* **patch\_size: 50**
* **projection\_dim: 32**
* **transformer\_layers: 5**
* **num\_heads: 8**
* **transformer\_units\_1: 112**
* **mlp\_head\_units\_1: 660**
* **mlp\_head\_units\_2: 56**

For a comparative study we trained model based on the custom blocks. The architecture was then tuned with the help of hyperband optimizer.

It tuned the following hyperparameters of the model architecture:

* **Number of blocks:** You can control the model complexity by setting the number of stacked blocks.
* **Dropout rate:** Dropout helps prevent overfitting by randomly dropping units during training.
* **Filter size:** The number of filters used in the convolutions (filters) is optimized.
* **Block selection:** For each block, we choose whether to use pooling reduction, convolutional reduction, or localization. This allows the model to dynamically learn the most effective combination of blocks.
* **Reduction block depth:** The depth of the reduction blocks is also tuned.

This approach allows for a flexible and adaptable model architecture that can be optimized.

* **model\_length:** 3
* **dropout\_rate:** 0.2
* **filters:** 56
* **use\_pooling\_reduction\_0:** True
* **use\_conv\_reduction\_0:** True
* **use\_localization\_0:** False
* **conv\_reduction\_n\_0:** 1
* **pooling\_reduction\_n\_0:** 1
* **use\_pooling\_reduction\_1:** True
* **use\_conv\_reduction\_1:** True
* **use\_localization\_1:** True
* **conv\_reduction\_n\_1:** 2
* **pooling\_reduction\_n\_1:** 1
* **pooling\_localization\_n\_1:** 1
* **use\_pooling\_reduction\_2:** True
* **use\_conv\_reduction\_2:** True
* **use\_localization\_2:** False
* **conv\_reduction\_n\_2:** 2
* **pooling\_reduction\_n\_2:** 2
* **use\_pooling\_reduction\_3:** False
* **use\_conv\_reduction\_3:** False
* **use\_localization\_3:** False

After training all the models the following table is obtained:

|  |  |  |
| --- | --- | --- |
|  | ViT Model | DNN Model |
| Loss | 242.02 | **25.02** |
| MAE | 11.37 | **3.51** |
| Valid loss | **19.33** | 26.4 |
| Valid MAE | **2.43** | 3.41 |
| Test loss | **22.87** | 30.2 |
| Test MAE | **2.70** | 3.76 |

Table 2- Training Results of ViT and DNN

**7. Conclusion**

This study explored the use of intrinsic segmentation models to estimate food calories. We analyzed the FooDD dataset and developed a new deep learning architecture featuring custom-designed blocks for Feature Extraction (FEB), Feature Localization (FLB), Feature Reduction (FRB), and Feature Distillation (FDB). We constructed and trained two contrastive models based on this architecture.

The ViT model shows a lower validation loss and mean absolute error (MAE), but its significantly higher training loss and MAE may be a cause of overfitting. In contrast, the DNN model demonstrated a more balanced performance, with both training and validation losses and MAEs being lower.

These deep learning models, especially the DNN model, present a promising alternative to conventional image processing methods for determining the calorie content of fruits. By learning complex patterns and features directly from the image data, these models can potentially deliver more accurate and reliable calorie estimates. In contrast to specialized image processing functions that often depend on fixed rules and thresholds, deep learning models can adjust to different image variations and lighting conditions. This adaptability allows them to manage a broader range of fruit types and image qualities, making them a flexible solution for calorie estimation in various applications, including food tracking apps and dietary analysis tools.

These findings show the importance of model regularization and careful hyperparameter tuning to prevent overfitting and improve generalization performance.

This research lays the groundwork for further exploration of intrinsic segmentation models paired with feature localization methods for estimating food calories. Future studies may delve into the effects of more advanced FLB architectures or evaluate the effectiveness of this approach on datasets with greater image variability.

**8. References:**

**[1]** Caloriemeter: Food Calorie Estimation using Machine Learning. (2021, March 5). IEEE Conference Publication | IEEE Xplore. <https://ieeexplore.ieee.org/document/9397023>

**[2]** Food image recognition and calorie prediction. (2021, April 21). IEEE Conference Publication | IEEE Xplore. <https://ieeexplore.ieee.org/document/9422510>

**[3]**Parisa Pouladzadeh, Abdulsalam Yassine, Shervin Shirmohammadi. (2020). FooDD: Food Detection Dataset for Calorie Measurement Using Food Images. IEEE Dataport. <https://dx.doi.org/10.21227/yvk7-qk38>

**[4]** Zhou, J., Bell, D., Nusrat, S., Hingle, M., Surdeanu, M., & Kobourov, S. (2018). Calorie estimation from pictures of Food: Crowdsourcing Study. Interactive Journal of Medical Research, 7(2). https://doi.org/10.2196/ijmr.9359

**[5]** Dielle Horne, Romina Palermo, Markus F. Neumann, Regan Housley, Jason Bell, Can People Accurately Estimate the Calories in Food Images? An Optimised Set of Low- and High-Calorie Images from the food-pics database,Appetite,Volume 139,2019,ISSN 0195-6663,

<https://doi.org/10.1016/j.appet.2019.04.017>

**[6]** Yang, H., Wang, Z., Liu, X. *et al.* Deep learning in medical image super resolution: a review. *Appl Intell* 53, 20891–20916 (2023). <https://doi.org/10.1007/s10489-023-04566-9>

**[7]** Pouladzadeh, P., Yassine, A., Shirmohammadi, S. (2015). FooDD: Food Detection Dataset for Calorie Measurement Using Food Images. In: Murino, V., Puppo, E., Sona, D., Cristani, M., Sansone, C. (eds) New Trends in Image Analysis and Processing -- ICIAP 2015 Workshops. ICIAP 2015. Lecture Notes in Computer Science(), vol 9281. Springer, Cham. <https://doi.org/10.1007/978-3-319-23222-5_54>

**[8]** Pouladzadeh, Parisa & Kuhad, Pallavi & Peddi, Sri Vijay Bharat & Yassine, Abdulsalam & Shirmohammadi, Shervin. (2016). Food calorie measurement using deep learning neural network. 1-6. 10.1109/I2MTC.2016.7520547.

**9. Appendix:**

For optimizing the architecture of the models, I employed Hyperband Optimization.

In the following sub section are the graphical results of the model hyperparameter optimization.

**9.1: ViT Optimization**

A graph with blue lines and red lines

Description automatically generated

A graph with red and blue lines

Description automatically generated

A graph with blue and pink bars

Description automatically generated

A graph with blue and pink bars

Description automatically generated

A graph with blue and pink bars

Description automatically generated

A graph of different sizes of patches

Description automatically generated with medium confidence

A graph of layers with different colors

Description automatically generated with medium confidence

A graph with blue and pink bars

Description automatically generated

**9.2: DNN Optimization**

**A graph with red and blue lines

Description automatically generated**

**A graph with pink and blue bars

Description automatically generatedA graph with pink bars

Description automatically generatedA graph with numbers and lines

Description automatically generatedA graph with pink and blue bars

Description automatically generatedA graph with pink and white lines

Description automatically generatedA graph with pink bars

Description automatically generatedA graph with pink bars

Description automatically generatedA graph with pink and blue bars

Description automatically generatedA graph with pink and white bars

Description automatically generatedA graph with pink and blue lines

Description automatically generatedA graph with pink and white lines

Description automatically generatedA graph with pink and blue lines

Description automatically generatedA graph with pink and white bars

Description automatically generatedA graph with pink and blue bars

Description automatically generatedA graph with pink bars and white text

Description automatically generatedA graph with pink and white lines

Description automatically generatedA graph with pink and blue bars

Description automatically generatedA graph with pink and blue lines

Description automatically generatedA graph with pink and blue bars

Description automatically generatedA graph with pink and blue bars

Description automatically generatedA graph with pink and blue lines

Description automatically generatedA graph with pink and white lines

Description automatically generated**