Semantic Segmentation

Jay Prajapati
University of Maryland
College Park
jayp@umd.edu

Akash Parmar
University of Maryland
College Park
akasparam@umd.edu

Naveen Anil
University of Maryland
College Park
nvnanil@umd.edu

Aditi Arun Bhoir
University of Maryland
College Park
abhoir@umd.edu

Sarin Ann Mathew
University of Maryland
College Park
sarinann@umd.edu

Abstract

This project aims to develop a deep-learning model for semantic food segmentation. Semantic segmentation visualizes the information about the various image regions by analyzing the images and videos fed to the model on a pixel level. Food segmentation is crucial for developing and understanding the different food regions within the given input and is used in applications such as contamination and quantity detection. The report provides an overview of the project's problem statement, the work behind building and training the models, testing and validation of the datasets and the evaluation of the model's performance.

1. Introduction

This project aims to construct a deep-learning model for semantic food segmentation. This initiative aims to address the difficulties associated with food quality control and inspection by automating the process of identifying defects and contaminants in food products. The section also emphasizes semantic segmentation as a potent technique for analyzing images at the pixel level and extracting meaningful information about the visual properties of various image regions.

Food segmentation utilizing semantic segmentation is an emerging discipline that holds great promise for addressing several of the challenges facing the food industry. Deep learning is a subfield of machine learning that involves training neural networks to recognize image patterns. Semantic segmentation aims to classify each pixel in an image, such as by the type of food or the presence of contaminants. This level of granularity provides more information than traditional image classification, which only identifies the image's dominant class.

The significance of food segmentation based on semantic segmentation lies in its potential to enhance the effectiveness and precision of food quality control and inspection. The food industry is subject to stringent food safety and quality regulations and standards. Nevertheless, conventional inspection techniques are frequently time-consuming, subjective, and prone to human error. By automating the examination process and yielding more objective and precise results, semantic food segmentation can offer a solution to these problems. Microorganisms and chemical residues are examples of difficult-to-detect defects or contaminants in food products that can be identified with this technology.

Food classification using semantic segmentation can reduce food waste, which is another essential aspect. An estimated 1,3 billion tons of food are squandered yearly, making food waste a significant global problem. A substantial portion of this waste is attributable to defective or contaminated products discarded during production or recalled after reaching the market. By identifying defective products early in production, semantic segmentation can reduce the likelihood of contaminated products reaching the market, thereby reducing food waste.

Several requirements must be met to create a model for food stratification using semantic segmentation. Initially, a large dataset of high-quality culinary images is required. The images should be annotated to distinguish food product types and their corresponding regions. Additionally, the dataset should be diverse enough to account for various illumination conditions, angles, and backgrounds. Additionally, the data should be

preprocessed to eliminate noise, rectify distortion, and standardize color and brightness.

Semantic segmentation, a popular deep-learning technique, has the potential to revolutionize the food industry by improving food inspection, reducing waste, and enhancing supply chain management. This project aims to develop a real-time semantic segmentation model to classify food items.

2. Methodology

Our methodology comprises several essential stages to ensure accurate and efficient food item segmentation. We employ a model already trained on a large dataset for our initial segmentation procedure. This pre-trained model facilitates transfer learning because it possesses a high level of comprehension and generalizability. Using this pre-trained model, we can efficiently extract meaningful features from our input data, laying the groundwork for the following stages.

Following the initial stage, we created our custom model. This model is tailor-made for our product segmentation task. It is akin to developing a tool that comprehends the distinct characteristics of our dataset and knows what to search for when separating various food items. This custom model enhances the precision and efficacy of our segmentation procedure.

Once our custom model is complete, we employ it to segment our culinary images. This indicates that the model thoroughly examines each image and identifies the various parts of each food item. Comparable to having a trained expert who can recognize and classify food items based on their appearance. Using our custom-built model produces more precise and accurate results than the initial pre-trained model.

To further improve our model, we employ a “transfer learning” technique that focuses explicitly on food segmentation. We incorporate a second food-image-trained model already trained on a large dataset. This enhances our model's understanding of food-related characteristics and its ability to identify and segment food items in the images we provide.

With our enhanced custom model and the advantages of transfer learning, we are finally prepared to make predictions. Our model is fed new, unseen images and performs its wizardry to accurately separate and identify the various food items within those images. These predictions can be used for various purposes, including food recognition, portion estimation, and diet analysis. Our methodology incorporates the strengths of pre-trained models, custom model development, transfer learning, and accurate predictions to achieve high-quality food image understanding and analysis results.

2.1. Pre-trained Model

To instate the project, we have implemented SAM semantic segmentation module. The Segment Anything Model (SAM) from Meta AI is an AI model that can "cut out" any object in an image with a single click. SAM is promotable and exhibits zero-shot generalization, requiring no additional training for unfamiliar objects. Trained on a dataset of 11 million images and 1.1 billion masks, SAM excels at segmenting objects of all shapes and sizes, even in cluttered scenes. It has applications in image editing, object detection, and scene understanding, allowing for easy removal of unwanted objects, object identification and tracking in videos, and generating masks for training other AI models. SAM has the potential to revolutionize image interaction and analysis.

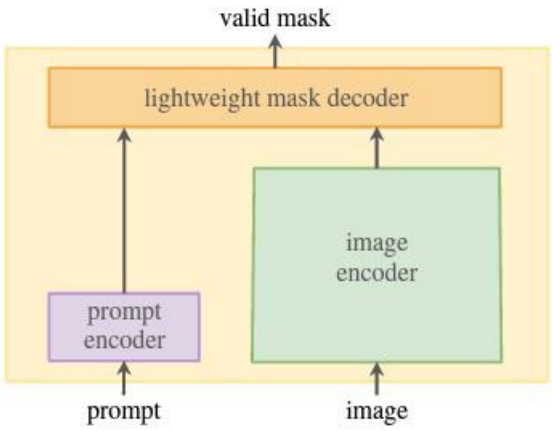

Figure 1: Architecture for SAM

The refinement network is crucial in a Segmentation Anytime Model (SAM). It uses techniques like dilated convolutions, spatial pyramid pooling, and encoder-decoder structures to capture fine details, contextual information, and connections between different parts of an image. SAM models sometimes use an iterative approach to improve the refinement process. In each iteration, the refined segmentation mask is fed back into the refinement network, allowing it to improve the segmentation results gradually. This can happen for a fixed number of iterations or until the results reach a desired level of accuracy. SAM models also take advantage of multi-scale processing, which means they analyze the input image or the segmentation mask at different levels of detail. This helps

the model understand both the smaller details and the bigger picture, leading to more precise and reliable segmentation outcomes.

Additionally, SAM models may include attention mechanisms. These mechanisms help the model focus on important regions or features of the image or the segmentation mask. By assigning more weight to relevant areas, attention mechanisms enhance the overall segmentation performance. A Segmentation Anytime Model's specific choices and details can vary depending on the task, available resources, and dataset characteristics. Different combinations of network architectures, modules, and techniques are used to find the right balance between accurate segmentation and efficient computation. It's important to note that SAM models are designed to be flexible. They allow users to obtain segmentation results at any process stage according to their needs. This flexibility makes them suitable for real-time or interactive segmentation tasks and situations where there are limitations on computational resources.

The output images using this module are shown in Figure-2.

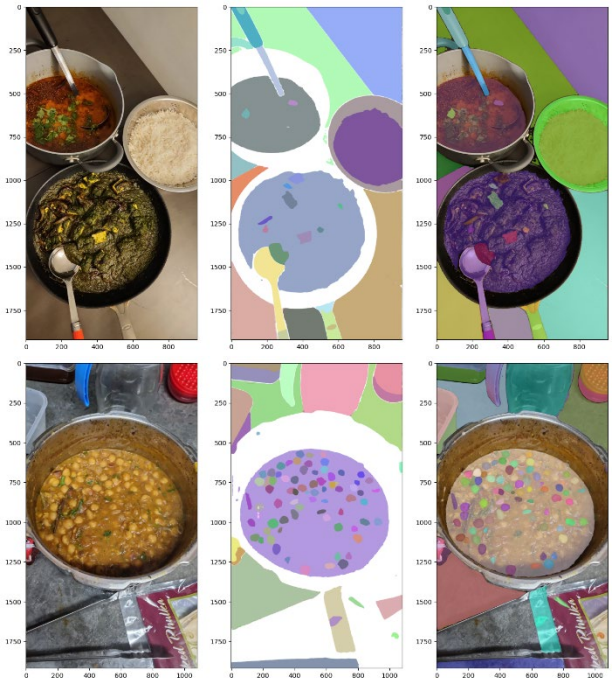

Figure 2: Output using SAM.

2.2. E-Net Architecture

The ENET (Efficient Neural Network) architecture was chosen to build and train a model that balances accuracy and computational efficiency. This architecture was trained over the FoodSeg103 annotated dataset with over 104 ingredient classes. The architecture employs depth-wise separable convolutions that reduce the parameters and time needed for computation. Many mobile applications built to improve health and provide the right dietary requirements will require semantic segmentation to identify the food ingredients and calculate the calories. These applications cannot run on heavy architectures that require heavy computational resources. The ENET architecture is well-suited for these resource-constrained environments and real-time applications that expect outputs of fast and precise results.

Key Components of ENET Architecture

Initial Block:

The input image is first passed through an initial block that performs initial feature extraction. It consists of a convolutional layer, a batch normalization layer, and a PReLU (Parametric Rectified Linear Unit) activation function. This block helps capture low-level features. The level performs a 3x3 convolution filter and a maxpool2d operation to reduce the spatial dimensions of the input. This block captures the image's or video frame's most important features.

Downsample Blocks:

ENET utilizes a downsampling strategy to progressively reduce the spatial resolution of feature maps. Each downsamples block consists of a series of convolutional layers, batch normalization, and PReLU activation. Downsample blocks employ a 3x3 convolution with a stride of 2 to reduce the feature map size while increasing the number of channels. The downsampling bottleneck in ENET combines max pooling, convolutional operations with different kernel sizes, activation functions, padding, and regularization to process the input features. It performs dimension reduction and non-linear transformations, ultimately helping capture important information and prepare the features for further processing in subsequent layers of the ENET model.

Upsample Blocks:

ENet employs upsampling blocks to increase the spatial resolution of feature maps that were lost during the downsampling in the encoding pathway. The block consists of a series of convolutions, transpose convolutions, max unspooling, regularization for preventing overfitting and passing the resultant output of the convolutions, and the maxunpool2d operations through a PReLU activation function. ENet incorporates two regularization techniques: dropout and spatial dropout. Dropout randomly sets a portion of feature map activations

to zero during training to prevent overfitting. Spatial dropout randomly sets entire channels of feature maps to zero to encourage feature diversity and prevent co-adaptation. They also include skip connections to retain, reuse and integrate information from earlier layers.

The architecture also employs an asymmetric bottleneck combining convolutions, padding, and regularization. This architecture allows the model to capture features with different spatial extents in the horizontal and vertical directions, enhancing its ability to represent complex patterns and structures. The block helps reduce dimensions, introduce non-linearity, and prepare the features for further processing in subsequent layers of the ENet model.

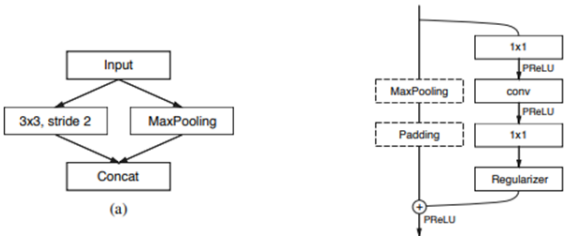

Figure 3: The initial block and basic architecture of bottleneck blocks

The next part of building the model implements a training loop using a Cross-Entropy loss function with class weights. This task can be broken down into the following steps:

- Determining the class weights. Each dataset is collected and classified using certain classes. Since each class does not necessarily have an equal number of images to be trained, the class weights help address class imbalance issues by assigning higher weights to underrepresented classes during training. The weights are inversely proportional to the logarithm of the propensity score of each class.
- Next, a criterion is used for the optimization process. The cross-entropy function considers the class weights that were calculated earlier and measures the dissimilarity between the predicted probability distribution and the true probability distribution of the actual class labels and the ground truth images.

Mathematically, the cross-entropy loss is defined as:

$$CrossEntropyLoss = -\Sigma(y * log(p))$$

- Finally, the model is trained over multiple epochs taking a few batches of the input test data and their respective annotated images with the different classes of labels. After evaluating the losses from the entropy function and optimizing the weights, it passes the validation images through the model to obtain the predicted outputs.
- This trained model is saved and validated over sample images to create disparity images. The different regions from the images of the food generated from these disparity maps are visualized using segments of different RGB values.

2.3. Transfer Learning

Transfer learning is a machine learning technique in which we take the capabilities of a model trained for a specific task and retrain it for another task. The pre-trained model is trained on a large dataset, and we leverage the already trained features using transfer learning. Instead of training a model from scratch, transfer learning allows us to take advantage of the knowledge and representations learned by the pre-trained model. This can be beneficial, especially when the new task has limited labeled data. The idea behind transfer learning is that low-level features learned by the pre-trained model, such as edges, shapes, or textures that can be applied to other tasks, are frozen, and the classification layer is changed. By using a pre-trained model, we can save computational resources and training time. Transfer learning models achieve optimal performance faster than traditional ML models. It is because the models that leverage knowledge (features, weights, etc.) from previously trained models already understand the features. It makes it faster than training neural networks from scratch. The pipeline is as follows:

Select a pre-trained model: The first step is to choose a pre-trained model that has been trained on a large dataset. The pre-trained model should have relevant features that can be transferred for performing the new task.

- **Data Preparation**: Create a new dataset with annotated or labeled images of food to train the model for food segmentation.

- **Create a new model**: This is the most important part of transfer learning. To create a new model, the following steps are followed:

    - Freeze the layers containing the low-level information.
    - Create a trainable layer over the frozen layers.

- The training process involves feeding the input data through the network, computing predictions, calculating the loss, and updating the model's weights using optimization algorithms like gradient descent. The number of training epochs and the

choice of hyperparameters depends on the dataset.

- **Fine tuning**: Unfreeze the frozen layers and train them slowly at a very low learning rate, as it will cause overfitting of weights.

- **Evaluation**: After training, evaluate the performance of your transfer learning model on a separate validation or test dataset. Measure the accuracy and precision to assess how well the model performs on the specific dataset.

Here we are doing transfer learning on a training model based on ResNet50 architecture. The Resnet50 architecture has more than 23 million trainable parameters, thus making it a powerful model for image classification, detection, and segmentation.

The input layer is typically of size 224x224 pixels with three color channels (RGB). Figure 4 shows the architecture layout of ResNet50. The entire layers can be divided into different stages. Stage 1 consists of convolution followed by ReLu and pooling layers. Stage 2 to Stage 5 contains several residue blocks stacked on top of each other. Each residual block consists of multiple convolutional layers and skip-connections. The skip connections allow the gradient to bypass one or more convolutional layers, facilitating the flow of information through the network. After several residue blocks, a max polling operation is applied to reduce the spatial dimensions of the features. The output of the last polling layer is flattened and fed into a fully connected layer.

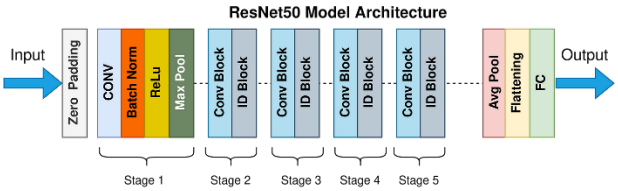

Figure 4: architecture layout of ResNet50

Here we remove the fully connected layer and add a new layer whose weights are adjusted according to the new dataset. Transfer learning is a powerful technique in deep learning that allows us to leverage knowledge learned from one task or domain and apply it to another related task or domain. Our algorithm aims to adapt a pre-trained ResNet-50 model to a new food dataset by replacing its fully connected layer and training it on new data.

- The step by the breakdown of the code is as follows:

- The necessary libraries are imported, including 'torch' and 'torchvision.'

- The data is loaded using the 'load_data' function, which creates data loaders for the source and target domains.

- A custom transfer learning model, 'transferModel' is defined, initializing a pre-trained ResNet-50 model and replacing its fully connected layer with a new one.

- An instance of the 'Transfer model' is created and moved to the GPU.

- The model is tested with a random input tensor to check if it's working correctly.

- The 'finetune; function is defined, which performs the fine-tuning process. It loops over epochs and performs training and evaluation on the source and target domains.

- The model parameters are organized into different parameter groups for optimization.

- The SGD optimizer is created with the defined parameter groups and momentum.

- The 'finetune' function is called to start the fine-tuning process.

3. Result

The initial step in the approach was to train a model using a custom model. The results for the custom model are given below.

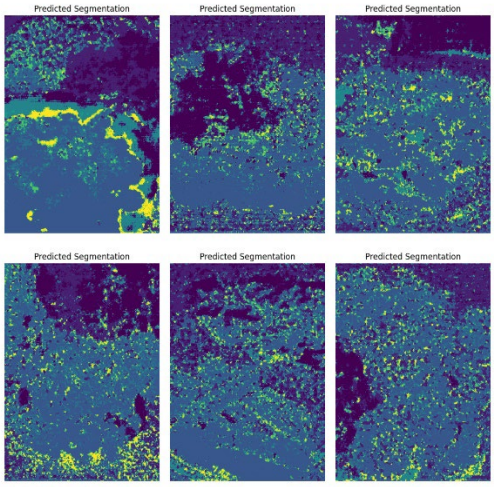

Figure 5: Output for the pointrend custom model

Following are the Transfer Learning model results.

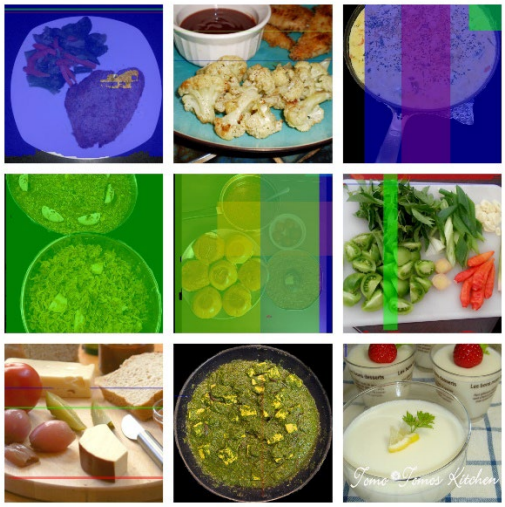

Figure 6: Output before Transfer Learning

Figure 7: Output after applying Transfer Learning

The predicted outputs are compared with the ground truth labels to calculate the loss using a specified criterion (e.g., cross-entropy loss). The number of correct predictions is updated by comparing the predicted outputs with the labels. The total loss is accumulated by multiplying the batch loss by the number of inputs in the batch. At the end of the batch iteration, the epoch loss is calculated by dividing the total loss by the total number of samples seen so far in the current phase. The epoch accuracy is calculated by dividing the number of correct predictions by the total number of samples.

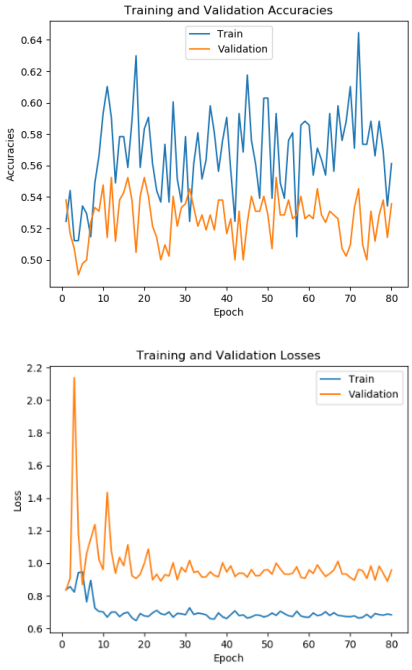

Figure 8: Loss and Accuracy results for 80 iterations and 500 images.

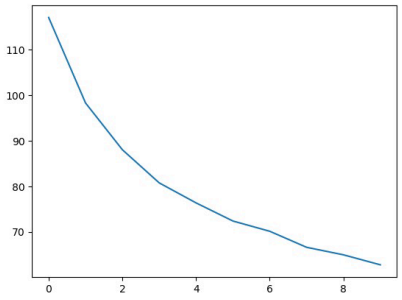

Figure 9: Plot for training losses for custom model

The accuracy and losses shown in Figure 8 are not from the final model, which was trained for 4935 images and tested for 2135 images for 200 iterations. Due to the shortage of time, we are just training and testing the model for fewer images and iterations to plot the loss and accuracy results. The final test accuracy for the transfer learning model and custom-built model, which has been trained for 4935 images and tested for 2135 images for 200 iterations, is given below. The accuracy for the transfer learning model is about 0.67 percent, and the custom-built model's accuracy is 0.43.

4. Conclusion

From the comparison between different architectures and different training techniques, we can conclude that a large-sized annotated data set is required for training models from scratch. This limitation can be overcome by the process of transfer learning, where we use limited data to modify a pre-trained model for food segmentation. The model developed through transfer learning could identify and segment different food regions.

The superior performance of the transfer learning model suggests that the pre-trained ResNet50 model captures relevant and discriminative features that are essential for the given task. It effectively learns task-specific information while retaining previously learned knowledge by fine-tuning the pre-trained model and adapting it to the new task. This enables the transfer learning model to achieve higher accuracy and better overall performance.

4. References

[1] Bossard, L., Guillaumin, M., & Van Gool, L. (2014). Food-101 – Mining Discriminative Components with Random Forests. Computer Vision Lab, ETH Zürich, Switzerland.

[2] J. Marin, N. Ponomarenko, V. Melnikov, K. Egiazarian, and J. Astola, "FoodCam: A Mobile Food Recognition System Based on Deep

[3] Convolutional Neural Network," in IEEE International Conference on Computational Photography (ICCP), 2017. [Online]. Available: http://foodcam.mobi/dataset100.html

[4] Wu, X., Fu, X., Liu, Y., Lim, E. P., Hoi, S. C. H., & Sun, Q. (2019). A Large-Scale Benchmark for Food Image Segmentation. arXiv preprint arXiv:1902.05797

[5] Paszke, A., Chaurasia, A., Kim, S., & Culurciello, E. (2016). ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. arXiv preprint arXiv:1606.02147.

Link to Code	here