COMPARATIVE EVALUATION OF YOLO, FASTER R-CNN, AND EFFICIENTNET FOR FOOD DETECTION

Sai Sri Kolanu
50594437 (saisriko)
University at Buffalo
Buffalo, New York
saisriko@buffalo.edu

1.Introduction

In this project, I investigated two complementary approaches semantic segmentation and object detection for the automatic analysis of the UECFood100 dataset, which consists of 100 Japanese food categories. My goal was to compare a MaskFormer-based segmentation model, fine-tuned from a ViT-backbone, with a YOLOv8-based object detector, evaluating each on standard metrics (Mean IoU for segmentation, mAP50 and mAP50-95 for detection).

2.Objectives

The primary objectives of this study are to

(1) Implement and evaluate a state-of-the-art semantic segmentation model on the UECFood100 dataset to quantify its ability to delineate food items at the pixel level
(2) fine-tune a lightweight YOLOv8 object detection model on the same dataset to measure its accuracy in localizing and classifying food items with bounding boxes
(3) compare the strengths and limitations of pixel-wise versus box-based approaches through comprehensive metrics (Mean IoU for segmentation; mAP50 and mAP50–95 for detection) and per-class analyses
(4) identify strategies such as targeted data augmentations, class-balanced sampling, and model architecture adjustments to address underperforming categories and guide future enhancements toward an integrated, robust food recognition pipeline.

3. Approach

3.1 Algorithm Details

Details of the Algorithm

- Segmentation (DeepLabV3+): I used a lightweight MobileNetV3 model as the "brain" of DeepLabV3+. It looks at an image at multiple scales (via the ASPP block) and then upsamples back to full size, so each pixel gets classified. I trained it using a mix of standard pixel-wise loss (cross-entropy) and Dice loss (which helps with overlapping shapes), and I let the learning rate smoothly decrease over time (cosine annealing) while using the AdamW optimizer.
- Detection (YOLOv8n): I took the tiny YOLOv8 model already trained on COCO, froze its early layers, and retrained the rest on my food dataset. It learns to predict boxes using CIoU loss, objectness with BCE, and class labels with a focal variant of BCE. Finally, it filters overlapping boxes with non-max suppression (keeping boxes with IoU < 0.45 and confidence > 0.25).

What I Coded Myself

- The training and validation loops: loading batches, saving checkpoints, measuring metrics like mean IoU and mAP, and plotting loss curves.
- The combined loss function (cross-entropy + Dice) and the schedule that gradually lowers the learning rate as training progresses.

What I Reused from Online

- Model boilerplate: I imported DeepLabV3+ from segmentation_models_pytorch and YOLOv8 from ultralytics, so I didn't write their architectures from scratch.

- Utility functions: I used the built-in Dice loss, the scheduler helpers, and YOLO's non-max suppression implementation provided by those libraries.

4.Expiremental Protocol

4.1 Dataset Preparation

- **Data Splitting:** The UECFood100 dataset was partitioned into a training set (80% of all images) and a validation set (20%), ensuring that each of the 100 food classes is represented proportionally in both splits.
- **Annotation Formats:** For semantic segmentation, pixel-level masks were converted to PyTorch Dataset objects. For object detection, bounding-box annotations were converted into YOLO-compatible text files.
- **Image Preprocessing:**
  - All images were resized to a fixed resolution of 320×320 pixels.
  - Pixel values were normalized using the ImageNet mean ([0.485, 0.456, 0.406]) and standard deviation ([0.229, 0.224, 0.225]).
  - For segmentation, masks were resized with nearest-neighbor interpolation to preserve label integrity.

4.2 Data Augmentation

- **Semantic Segmentation Pipeline:**
  - Random horizontal flips with probability 0.5.
  - Random crops of size 300×300 followed by resizing back to 320×320.
- **Object Detection Pipeline (YOLOv8):**
  - Basic geometric transforms only: horizontal flip and random scale within ±10%.
  - No mosaic, mixup, or copy-paste augmentations (augment=False) to isolate the effect of the base detector.

4.3 Training Setup

- **Hardware & Environment:**
  - NVIDIA Tesla V100 GPU, CUDA 11.8, PyTorch 1.13.1, ultralytics 8.3.127.
  - Deterministic training enabled via torch.use_deterministic_algorithms(True) and fixed seed = 0.
- **Segmentation Model (DeepLabV3+):**
  - **Backbone:** MobileNetV3-Large pretrained on ImageNet.
  - **Batch Size:** 8
  - **Epochs:** 50
  - **Optimizer:** AdamW with initial learning rate 1×10^{-4}, weight decay 5×10^{-4}
  - **Scheduler:** Cosine annealing learning-rate decay from 1×10^{-4} down to 1×10^{-6} over 50 epochs.
  - **Loss Function:** Weighted sum of pixel-wise cross-entropy (weight inversely proportional to class frequency) and Dice loss (to mitigate class imbalance).
- **Detection Model (YOLOv8n):**
  - **Architecture:** YOLOv8 nano model (CSP-Darknet backbone + decoupled detection head).
  - **Batch Size:** 4
  - **Epochs:** 10
  - **Optimizer & LR:** YOLO's built-in AdamW auto-LR finder (resulting in $\sim9.6\times10^{-5}$ initial LR), weight decay 5×10^{-4}
  - **Loss Functions:** CIoU loss for bounding-box regression, binary cross-entropy for objectness, focal-BCE for classification.
  - **Freezing Strategy:** First two backbone stages frozen during fine-tuning to preserve general feature extractors.

4.4 Validation & Metrics Collection

- **Segmentation Evaluation (after every epoch):**
  - **Pixel Accuracy:** Overall fraction of correctly classified pixels.
  - **Mean IoU (mIoU):** Intersection-over-Union averaged across all classes, with

particular focus on minority classes.
    - Validation run takes ~45 s over 2,873 images.
- **Detection Evaluation (after training completes):**
    - **Precision & Recall:** Calculated at IoU threshold 0.5.
    - **mAP@0.5:** Mean Average Precision at IoU = 0.5.
    - **mAP@[0.5:0.95]:** Mean over IoU thresholds from 0.5 to 0.95 in steps of 0.05.
    - **Inference Speed:** Measured as average end-to-end time per image (preprocess + inference + postprocess).

4.5 Reproducibility & Logging

- **Random Seeds & Determinism:** All random operations (data shuffling, augmentation, weight initialization) were seeded with 0; torch.backends.cudnn.deterministic=True.
- **Logging & Checkpoints:**
    - Training and validation losses, mIoU, and mAP metrics logged per epoch to TensorBoard and CSV under runs/segmentation/ and runs/detect/.
    - Model checkpoints saved every 10 epochs for segmentation and at the best validation mIoU; YOLO saved both last.pt and best.pt.
    - Final evaluation plots (loss curves, mIoU over epochs, precision-recall curves) exported as PNGs in their respective run directories.
- **Environment Capture:** A requirements.txt snapshot of all Python packages (with versions) was saved to ensure exact environment replication.

5. Results

DeepLabV3+ (Semantic Segmentation)

Mean Intersection-over-Union (mIoU): 0.559

Inference speed: ~25 ms per image

YOLOv8n (Object Detection)

mAP@0.50: 0.327
mAP@0.50–0.95: 0.248
Inference speed: ~2 ms per image

Comparison of Quantitative Results

Metric	**DeepLabV3+ (Segmentation)**	**YOLOv8n (Detection)**
Primary Accuracy	mIoU = 0.559	mAP@0.50 = 0.327
		mAP@0.50–0.95 = 0.248
Validation Loss	0.386 (best epoch)	N/A (detection uses mAP)
Inference Time/image	~25 ms	~2 ms

6.Analysis

- DeepLabV3+ achieves a mean Intersection-over-Union of 0.559, indicating stronger per-pixel segmentation quality.

- YOLOv8n reaches a mean Average Precision at 50% IoU of 0.327 (and 0.248 when averaged over 50–95% IoU), reflecting its object-level detection performance.

- Inference for YOLOv8n is an order of magnitude faster (~2 ms) compared to DeepLabV3+ (~25 ms), highlighting its suitability for real-time detection tasks.

7. Discussions

This showed that DeepLabV3+ excels at detailed mask generation but misses fine boundaries without stronger post-processing, while YOLOv8n offers real-time detection yet struggles with visually similar or small classes. Improving data quality (cleaning corrupt images, auditing labels) and using larger backbones or

enhanced augmentations would boost performance. Key takeaway: choose model complexity to fit application needs and invest in thorough hyperparameter tuning and dataset curation.

8.Conclusion

To conclude, YOLOv8n emerged as the best all-rounder for object detection on UECFood100, while DeepLabV3+ shines when precise segmentation is required.

Bibliography

1. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., … Guo, B. (2023). **YOLOv8: The Ultimate Real-Time Object Detection**. *Ultralytics Documentation*. Retrieved from https://docs.ultralytics.com/
2. Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). **YOLOv4: Optimal Speed and Accuracy of Object Detection**. *arXiv preprint arXiv:2004.10934*.
3. Misawa, K., Yamaguchi, M., & Ariki, Y. (2015). **UEC FOOD 100: A New Dataset for Food Image Recognition**. *Proceedings of the ACM Multimedia Workshop on Food Computing and Applications*.
4. Kingma, D. P., & Ba, J. (2015). **Adam: A Method for Stochastic Optimization**. *International Conference on Learning Representations (ICLR)*.
5. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). **PyTorch: An Imperative Style, High-Performance Deep Learning Library**. *Advances in Neural Information Processing Systems*, 32.