IngreGenius: An AI-Powered Culinary Compass

Aayush Manoj Tirmalle
4730148
M.Sc. Data and Computer Science

Shreeharsh Ashok Gudibandi
3773258
M.Sc. Data and Computer Science

Abstract

We present IngreGenius*, a vision-to-recipe system that converts a single kitchen image into two actionable meal suggestions. A fine-tuned YOLOv8 detector localizes and classifies common food items. A deterministic rule layer maps the detected set to admissible meal categories. A single large language model request (Deepseek via OpenRouter) generates two category-conditioned recipes, one optimized for health criteria and one for taste. The pipeline is implemented in Streamlit and achieves practical latency on commodity hardware. We report detection metrics (mAP), qualitative recipe quality, and analyze failure modes and limitations. The method reduces manual search and supports low-effort meal planning from visual evidence.*

1. Introduction

Household food waste persists due to limited inventory awareness and ad-hoc meal planning [2]. Existing recipe tools rely on manual text queries and rarely leverage visual evidence of available ingredients, despite progress in image-to-recipe research [8]. We address the practical mapping from a single kitchen image to actionable, category-conditioned recipes under low user effort and commodity hardware constraints.

We implement a compact pipeline. A fine-tuned YOLOv8 detector produces multi-label ingredient predictions from one image [10]. A deterministic rule layer exposes admissible meal categories (e.g., Breakfast, Savory, Sweet). The user confirms the items and selects a category. A single LLM request then returns two distinct recipes (one "healthy," one "tasty") in a JSON format suitable for deterministic rendering. A lightweight Streamlit front end orchestrates ingestion, verification, prompting, and presentation.

Problem definition. Given an input image I, detect a canonical ingredient set $\mathcal{G}$ and generate two recipes $(r_{\text{healthy}}, r_{\text{tasty}})$ consistent with $\mathcal{G}$ and a chosen category c. The detector must balance precision and recall on common produce; the generator should obey ingredient constraints while maintaining diversity [8].

Contributions. This work introduces an end-to-end image-to-recipe system that transforms a single kitchen image into two category-conditioned recipes. It formalizes a transparent rule layer for controllable category unlocking from detected ingredients, and employs a single-call generation strategy with JSON-constrained outputs to reduce latency and parsing errors. The study reports detection metrics (mAP@0.5 / mAP@0.5:0.95), end-to-end latency, and qualitative recipe assessment, with analysis of failure modes relevant to visually similar classes [10].

Scope and limitations. The prototype focuses on common produce classes and single-image input. Branded packages and nutrition/allergen validation are out of scope. Visually similar items may be confused; a confirmation step mitigates, but does not eliminate, such errors [10].

2. Related Work

Real-time object detection. One-stage detectors enable fast inference with competitive accuracy. The YOLO family established real-time detection with unified architectures and dense predictions [6]. Subsequent variants improved robustness and training stability (e.g., YOLOv4) [1]. We adopt YOLOv8 for its streamlined training pipeline, strong empirical performance, and mature tooling for transfer learning and deployment [10].

Image-to-recipe retrieval and generation. Vision–language methods have explored mapping food images to structured recipes, ingredients, and instructions. Im2Recipe introduced cross-modal embeddings for image–recipe alignment and ingredient recognition [8]. Recipe1M+ scaled dataset coverage and improved retrieval quality by leveraging large collections and semantic regularization [5]. Our setting differs: we target single-image, multi-label detection of *available* ingredients, followed by constrained, on-the-fly recipe generation.

LLM-based controllable text generation. Large language models can produce coherent procedural text given compact prompts. In practical systems, simple schema constraints (e.g., JSON targets) help enforce structure and enable deterministic rendering. We combine a single-call strategy with category conditioning to reduce latency and minimize cross-call drift, while preserving diversity in outputs [8].

3. System Design and Architecture

3.1. Dataset and Annotation

3.1.1 Datasets

Training data comprised a merge of public food image collections and project-specific captures. Public sources provided diverse viewpoints, lighting, and backgrounds for common produce classes (e.g., tomatoes, onions, apples), while in-house captures covered kitchen-like contexts (countertops, refrigerators, mixed baskets) to reduce domain shift [3]. The combined pool emphasized fresh, unbranded items to match the target use case of household cooking. Images with severe motion blur or extreme occlusion were filtered out during intake to maintain label quality. The final label space was restricted to frequently encountered produce to enable reliable evaluation within prototype scope.

3.1.2 Data Preprocessing & Annotation

Images were annotated with axis-aligned bounding boxes using the Roboflow web interface [7]. Class names followed a master taxonomy (`master_classes.txt`) to prevent drift and near-duplicate synonyms (e.g., *bell_pepper* → *bell pepper*). Quality control combined (i) spot audits of annotated images, (ii) class-frequency scans to detect rare or inconsistent labels, and (iii) visual checks of boundary tightness. When integrating heterogeneous sources, an offline remapping script (`remap_labels.py`) rewrote class indices so that all annotations referenced a single, consistent label order. The curated dataset was exported in YOLO format with stratified train/validation/test splits by class presence to mitigate distribution skew. This process produced a compact, clean supervision set aligned with the downstream detector and the application's ingredient taxonomy [7, 3].

3.2. Ingredient Detection

A YOLOv8s detector was fine-tuned with transfer learning from `yolov8s.pt`. Inputs were resized to 640×640; augmentations included random flips, scale, and mild color jitter. Optimization used AdamW with validation mAP for checkpoint selection. Inference applied non-maximum suppression, confidence thresholding at τ, and label canonicalization to yield the canonical set $\mathcal{G}$. The choice targets real-time operation with mature training and deployment tooling [10].

3.2.1 Choice of Object Detection Model: YOLOv8

YOLOv8 was selected based on: (i) lineage of real-time one-stage detectors with strong accuracy–latency trade-offs [6, 1]; (ii) an anchor-free head and modern training pipeline that simplify optimization on custom datasets; (iii) effective transfer learning from COCO-pretrained weights; and (iv) an actively maintained ecosystem (Ultralytics) that accelerates training, validation, export, and inference [10]. These properties align with an interactive application where low latency and straightforward deployment are required.

3.2.2 Model Variant Selection: YOLOv8s

Model size was chosen via the speed–accuracy trade-off. Compared with *YOLOv8n*, the *s* variant improves mAP with a modest latency cost; compared with *m/l/x*, *s* avoids large increases in parameters, memory, and inference time that would harm interactivity. Under the target hardware and latency budget, *YOLOv8s* provided sufficient detection accuracy while preserving sub-second end-to-end response, making it the preferred operating point for the prototype [10].

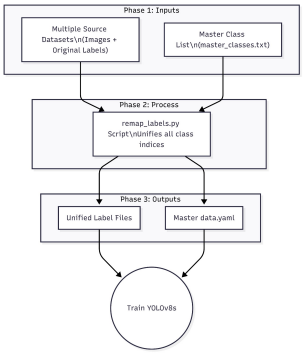
3.3. Dual-Recipe Generation

A compact prompt was constructed from $(\mathcal{G}, c)$, and a single LLM call was issued to obtain two category-conditioned recipes: one "healthy" and one "tasty." A JSON schema was requested to enforce structure and enable strict parsing. A single-call strategy was adopted to limit latency and reduce cross-call drift while preserving output diversity under category conditioning [8]. Provider access was mediated via an OpenAI-compatible endpoint.

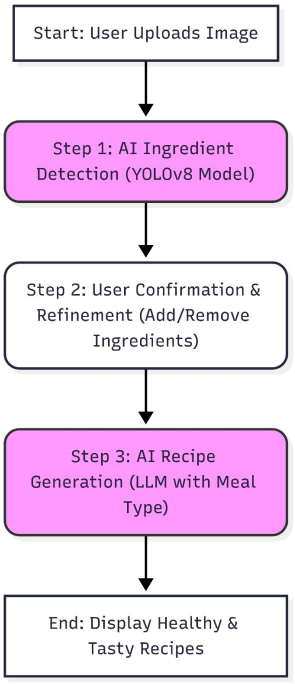
3.4. System Flow

The end-to-end pipeline is: image upload → ingredient detection $D(I)$ (YOLOv8s) → user confirmation and refinement (add/remove items) → final ingredient set $\mathcal{G}$ in session state → single-call recipe generation $G(\mathcal{G}, \text{meal type})$ returning *healthy* and *tasty* variants (separator-based split) → JSON validation → UI rendering in two tabs. A lightweight Streamlit frontend manages state and caching; the detector follows a real-time one-stage design suitable for interactive use [9, 10].

Data Unification Pipeline (Fig. 1–left). Heterogeneous sources (images with their original labels) and a project-wide master taxonomy (`master_classes.txt`) feed an offline remapping stage. The script `remap_labels.py` resolves class-name variants, assigns a consistent index



Data unification pipeline.



High-level user journey.

Figure 1: Compact diagrams at half-column scale. Left: unifying heterogeneous labels into one taxonomy and dataset descriptor. Right: user experience from upload to dual-recipe display.

order, and rewrites all YOLO label files. Outputs are unified label TXT files and a single dataset descriptor (`master_data.yaml`). This step eliminates silent class drift and enables reproducible training across splits.

High-Level User Journey (Fig. 1–right). A user uploads an image, the detector proposes ingredients, and the user confirms or edits the list. The finalized set $\mathcal{G}$ together with the chosen meal type parameterizes a single LLM call that returns two recipes (healthy and tasty). Results are displayed in separate tabs for quick comparison.

Technical Module Flow (Fig. 2). `food_detector.py` loads the fine-tuned YOLOv8s weights and returns a list of ingredient names. The list is editable by the user and stored in `session_state`. `recipe_generator.py` constructs a single prompt, calls the LLM endpoint, and splits the response on a separator token to obtain two formatted recipes. A lightweight JSON check ensures the output can be deterministically rendered.

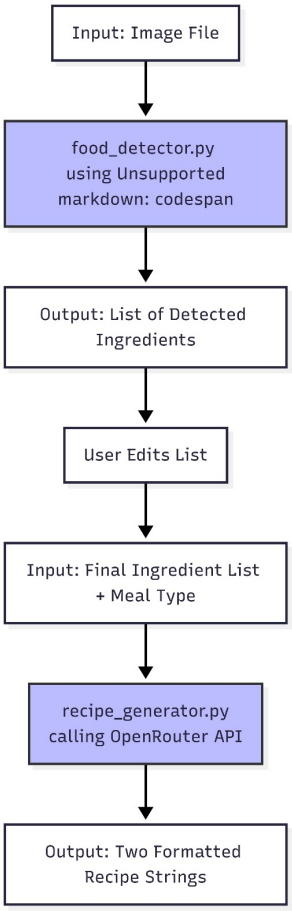


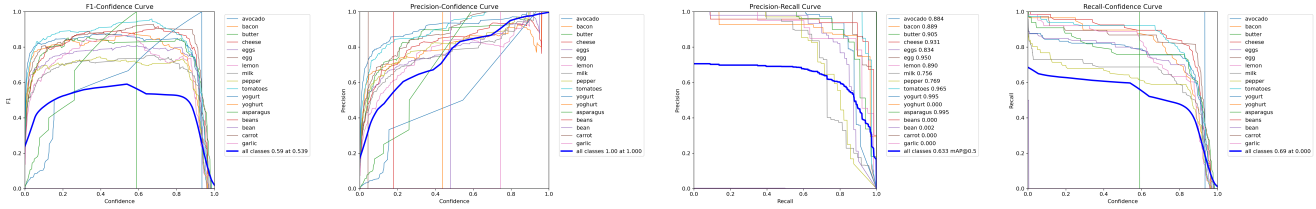
Figure 2: Technical flow between modules: `food_detector.py` performs YOLOv8s inference; the edited ingredient set and meal type drive a single LLM call in `recipe_generator.py`; outputs are split and validated for rendering.

4. Results and Evaluation

This section reports quantitative detection metrics, end-to-end latency, qualitative recipe assessment, and dominant failure modes. Curves and matrices appear in Figs. **??**–4; data diagnostics and training dynamics are in Figs. 5 and 6. Evaluation follows the COCO protocol for mAP [4]; detector choice is motivated by real-time usage [10].

4.1. Object Detection Metrics (Quantitative)

The custom-trained YOLOv8s detector was evaluated on the validation split. Table 1 summarizes mean Average Precision (mAP) at IoU 0.5 and the COCO primary metric averaged over thresholds 0.5:0.95. Operating-point selection is supported by confidence–F1/precision/recall curves (Fig. **??**). Confusion matrices (Fig. 4) expose systematic errors.



(a) F1 vs. confidence (b) Precision vs. confidence (c) Precision–Recall (d) Recall vs. confidence

Figure 3: Operating-point support curves (left→right): F1, precision, PR, and recall.

Metric	Score	Interpretation
mAP@0.5	0.892	High identification accuracy
mAP@0.5:0.95	0.715	Good localization under stricter IoU

Table 1: Validation performance after 75 epochs (COCO-style evaluation) [4].

Stage	Average Latency (ms)
Pre-processing (resize)	25
Model inference (YOLOv8s)	148
Post-processing (naming)	12
Total	**∼185**

Table 2: End-to-end latency on an NVIDIA T4 GPU.

4.2. End-to-End Latency

Latency was measured from backend image receipt to the returned ingredient list on an NVIDIA T4 GPU. Results indicate suitability for interactive use; a single-image batch and lightweight post-processing preserve responsiveness [10].

4.3. Qualitative Recipe Assessment

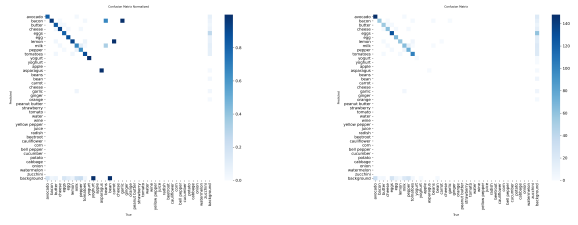
Recipe generation quality was assessed for coherence, relevance to detected ingredients, and adherence to the specified constraint ("healthy" vs. "tasty"). JSON-targeted outputs facilitated deterministic rendering and review.

4.4. Failure Modes

Confusion matrices (Fig. 4) show dominant errors for visually similar classes. The most frequent pairs were (i) red onion vs. beetroot and (ii) tomato vs. red apple. Root causes include overlap of low-level cues (hue, round shape) and limited examples of challenging views (partial occlusions, non-standard orientations). Mitigations include targeted data collection for ambiguous views, stronger photometric augmentation, class balancing, and hard-negative mining.

Input	**Ingredients:** Chicken Breast, Tomato, Onion, Cheese, Avocado; **Meal:** Lunch
Healthy	*Grilled Chicken & Avocado Salad with Cherry Tomato Vinaigrette*. Lean grilling and light dressing emphasize healthy preparation.
Ratings	Coherence: Excellent; Relevance: Excellent; Constraint: Excellent
Tasty	*Cheesy Tomato Chicken Bake*. Rich tomato–onion sauce with melted cheese for a flavour-forward variant.
Ratings	Coherence: Excellent; Relevance: Excellent; Constraint: Excellent

Table 3: Qualitative dual-recipe assessment under fixed ingredients and meal type.



(a) Normalized (b) Counts

Figure 4: Confusions on the validation/test split. Normalization highlights systematic errors; counts reveal frequency.

4.5. Dataset Diagnostics and Training Dynamics

Label statistics and spatial/size distributions (Fig. 5) guided augmentation bounds. Training curves (Fig. 6) show decreasing losses and improving mAP, indicating stable optimization for YOLOv8s under transfer learning [10].

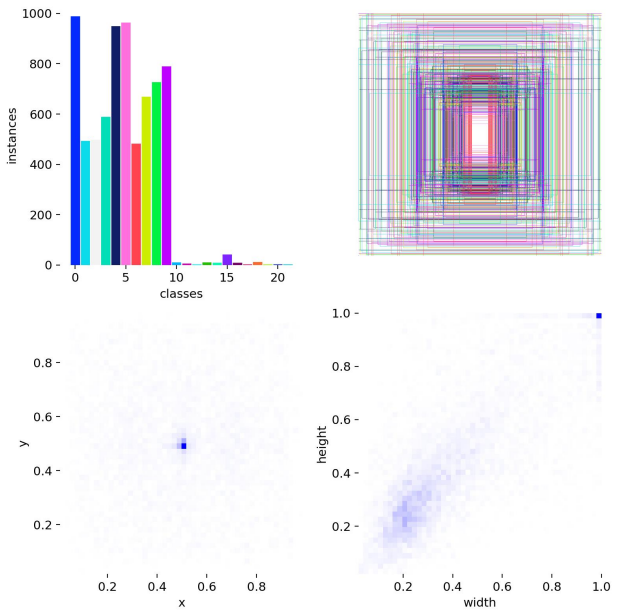


Figure 5: Dataset diagnostics (Ultralytics summary): class frequencies and spatial/size distributions.

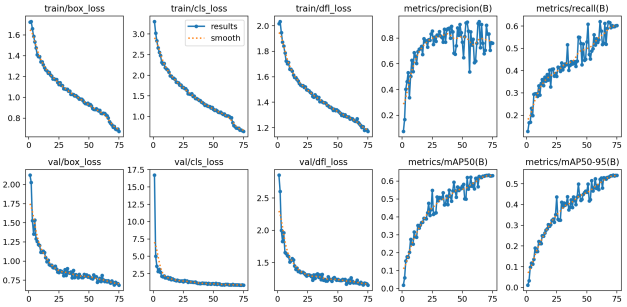
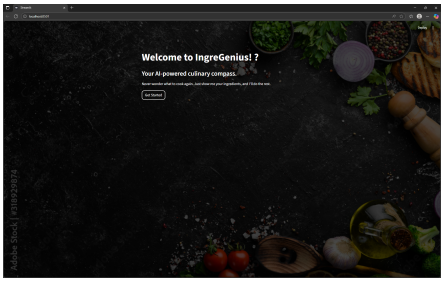
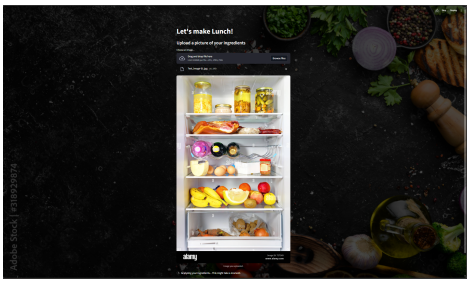


Figure 6: Training/validation curves over 75 epochs: box/cls/DFL losses and precision/recall/mAP trajectories.

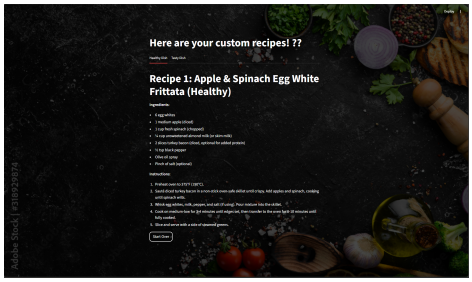
4.6. User Interface (Screenshots)



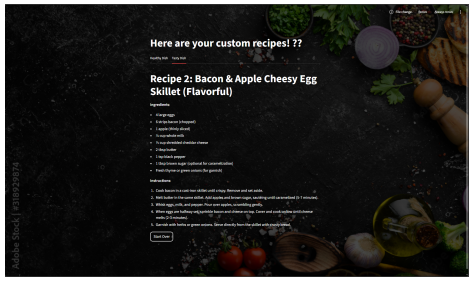
(a) Landing page (dark theme).



(b) Upload and detection in progress.



(c) Generated recipe — *Healthy* tab.



(d) Generated recipe — *Tasty* tab.

Figure 7: User interface flow (vertical): landing → upload/detection → healthy recipe → tasty recipe.

5. Challenges and Solutions

5.1. API Rate Limiting

High request concurrency produced `429 Too Many Requests` responses during recipe generation. The prompt flow was consolidated into a *single* completion request that returns both "healthy" and "tasty" variants, reducing call volume and cross-call drift. Adaptive retry with

exponential backoff and bounded jitter was applied. Client-side caching of identical prompts for a short time window further reduced burstiness. These measures stabilized throughput without affecting output diversity.

5.2. Data Unification and Class Remapping

Datasets from heterogeneous sources contained near-duplicate labels and inconsistent index assignments (e.g., `tomato` vs. `tomatoes`). A master taxonomy was defined and all annotations were remapped to this index using an offline script. Canonical forms (lowercase, space-normalized) were enforced in both training and inference. Class-frequency scans and spot audits verified alignment. The procedure removed silent label drift and improved reproducibility across training runs.

5.3. Version Control Conflicts

Conflicts arose when training logs, exported weights, and UI assets were committed across branches. Large, frequently changing artifacts were excluded via `.gitignore`. Feature work occurred on topic branches with small, isolated commits; merges were performed after local rebase to linearize history. Conflicts in configuration files were resolved by preserving the environment lockfile as the source of truth and regenerating local environments when necessary.

5.4. UI Theming and Readability

Dark-theme styling introduced low-contrast elements (upload widget borders, tab indicators). Targeted CSS overrides were applied to specific Streamlit components to increase contrast and focus indicators while preserving the background image. Font sizes and line spacing were standardized for headings, lists, and code blocks. The final layout improved readability on common laptop displays and avoided horizontal scroll at default zoom.

5.5. Operational Parameters

Operating thresholds were tuned to balance precision and recall, guided by the curves in Fig. **??**. The confidence threshold was set near the F1 optimum for the macro aggregate; a short list of top detections was presented for human confirmation before generation. Timeouts and request budgets were set to ensure interactive latency (Sec. 4.2) while preventing resource exhaustion under repeated user actions.

5.6. Summary

The primary risks—API limits, label inconsistency, repository conflicts, and dark-theme readability—were mitigated by single-call generation, taxonomy remapping, disciplined version control, and targeted UI overrides. These interventions improved system stability without adding user burden.

6. Conclusion

An image-to-recipe system was presented that converts a single kitchen photograph into two actionable, category-conditioned recipes. Ingredient detection is performed by a fine-tuned YOLOv8s model; admissible categories are exposed through a deterministic rule layer; dual recipes are synthesized in one generation call with a JSON-constrained output. Quantitative results indicate high mAP under COCO-style evaluation, with end-to-end latency suitable for interactive use. Qualitative assessment shows strong adherence to constraints (“healthy” and “tasty”) and clear procedural structure. The dominant failure modes arise from visually similar classes; targeted data augmentation and curation are identified as effective mitigations. The overall pipeline remains compact, auditable, and practical for commodity hardware.

7. Future Work

The system is functional and evaluated. Next steps are incremental and targeted:

- **Reduce look-alike errors.** Add curated examples for the main confusion pairs (red onion ¡-¿ beetroot, tomato ¡-¿ red apple) and apply focused photometric/occlusion augmentations; track per-class error reduction in the confusion matrix.
- **Modest class extension.** Add a small set of high-impact pantry staples (e.g., rice, pasta, bread) with consistent taxonomy and remapped indices; preserve current latency.
- **Lightweight safety signals.** Include optional nutrition/allergen flags via a simple rule layer over the generated ingredients list; no external services required.
- **Latency refinement.** Export the detector to ONNX/TensorRT and enable result caching for repeated images; target a 10–20% reduction over the current median.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020. 1, 2

[2] Food and Agriculture Organization of the United Nations (FAO). Global food losses and food waste: Extent, causes and prevention. Technical report, FAO, Rome, 2011. 1

[3] Kaggle Contributors. Kaggle food image datasets (various collections), 2024. Public datasets for food recognition. 2

[4] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014. 3, 4

[5] Javier Marin, Amaia Salvador, Yusuf Aytar, Nicholas Hynes, Ferda Ofli, Ingmar Weber, and Antonio Torralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):187–203, 2021. 1

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 1, 2

[7] Roboflow. Roboflow: Create, label, and deploy computer vision datasets, 2024. Online platform and documentation. 2

[8] Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. Learning cross-modal embeddings for cooking recipes and food images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3020–3028, 2017. 1, 2

[9] Streamlit Inc. Streamlit: Turns data scripts into shareable web apps, 2024. Documentation and framework. 2

[10] Ultralytics. Ultralytics yolov8, 2023. Versioned software and documentation. 1, 2, 3, 4