FoodCNN

Zero-shot nutrition estimation from a single photo
Deep Learning that reads your plate

École Polytechnique (X), June 2025

Georgii Kuznetsov, Georgy Salakhutdinov, Ayoub Agouzoul

Introduction

Relevance: With growing concerns about health and nutrition, there is a clear need for easy meal analysis and this project explores a potential solution through Machine Learning.

Goal:

- Predict calories and nutrition facts in a meal from a single photo

Problem:

- Input: a .png image of a dish on a plate
- Output: the amount of calories in that dish
- Data: *Nutrition5K* (3493 images and 5006 side angle videos)

Evaluation Metrics:

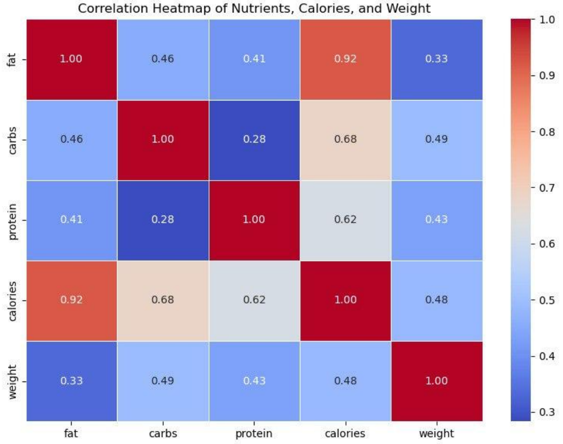
- MAE and MAE %
- RMSE
- R^2

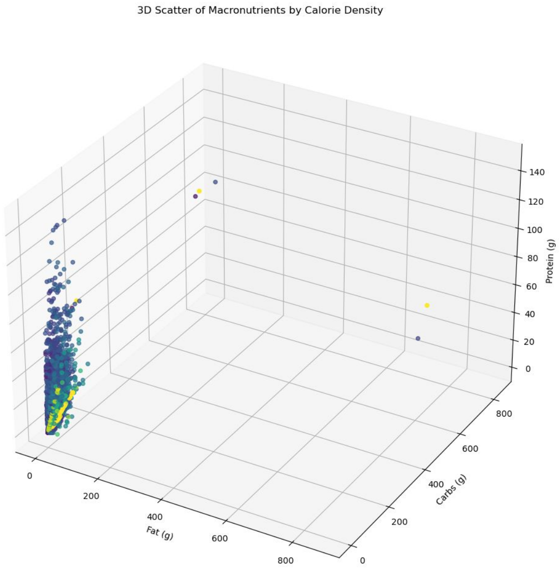
Calories:	290.6 kcal	(Err: 8.1%)
Fat:	13.9 g	(Err: 12.2%)
Carbs:	16.9 g	(Err: 10.8%)
Protein:	24.6 g	(Err: 4.8%)
Weight:	167.0 g	(Err: 1.2%)

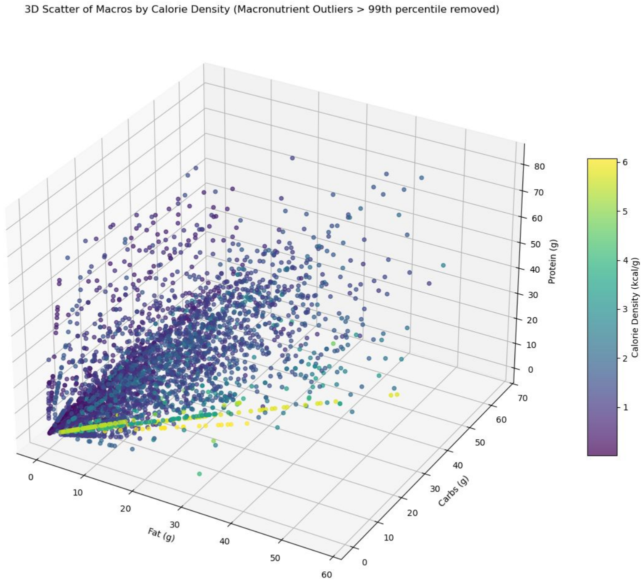
Example of prediction result

Table of contents

1. Data Exploration & Cleaning







Chosen dataset : Nutrition5k

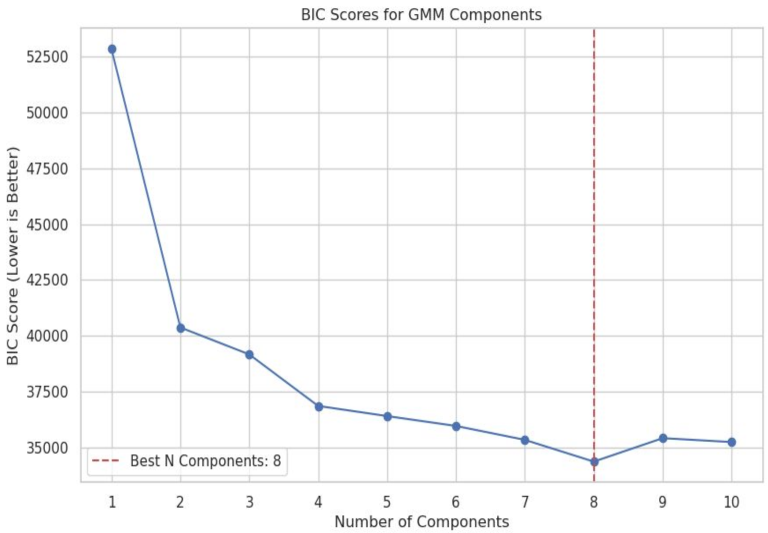
- 5,000+ unique dishes, 250+ ingredients
- Real-world cafeteria setting
- High-res RGB images, short videos, depth images (RGB-D)
- Ingredient weights and full nutritional breakdown (calories, fat, carbs, protein, weight)

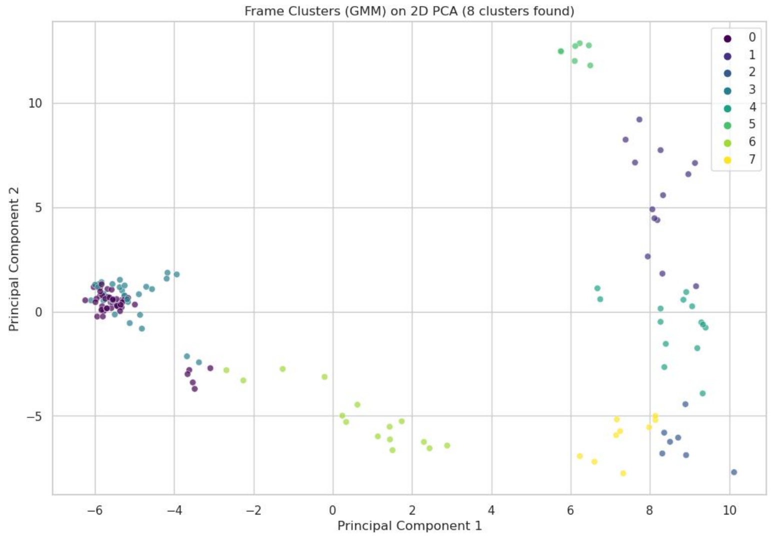
- Nutrient correlation heatmap for sanity checks and dataset exploration.
- Outlier removal (discard data points with macronutrient values exceeding 99th percentiles.
  ⇒ 4766 remaining data points

- Models perform worse on extremely small dishes ⇒ Idea: segmentation.
- Feature selection (4) based on the context and goals of this project.
- Torch random transformations
  (RandomCrop, RandomHorizontalFlip, ColorJitter)

2.1. Data: Augmentation

PCA showed that optimal amount of frames to take is 8 per video. However, this would imply ~166 GB of data, which is not feasible with the available resources. Hence, by elbow rule we chose 2. Image on the right should serve as a sanity check, that 8 frames is a clear overfit





Graph plot of BIC Scores relative to the amount of components (frames) to extract from an example video. Number of components corresponds to the amount of clusters of the plot on the right.

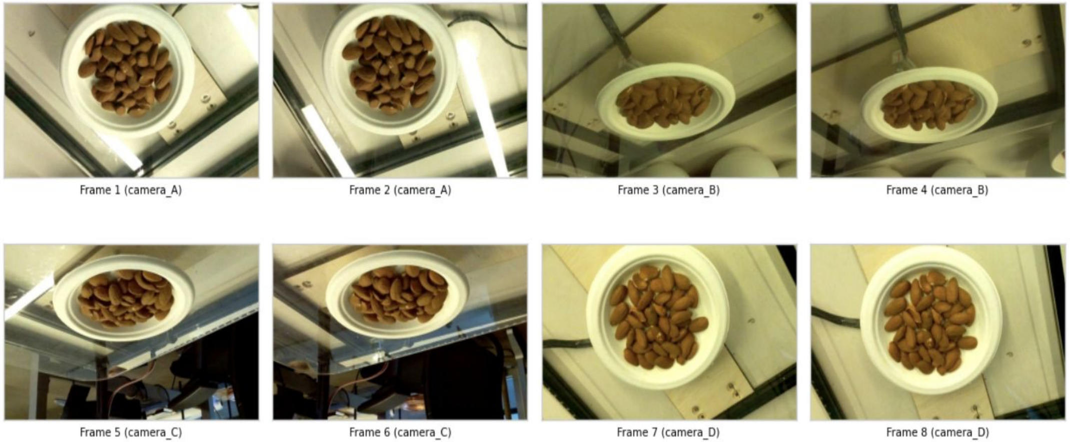
Scatter plot of frames of the example video of the dish in the PCA Space. GMM clustering is used to show the optimal amount of frames to extract

2.1. Data: Frames Extraction

Idea: extract equally splitted 8 frames from each video, i.e:

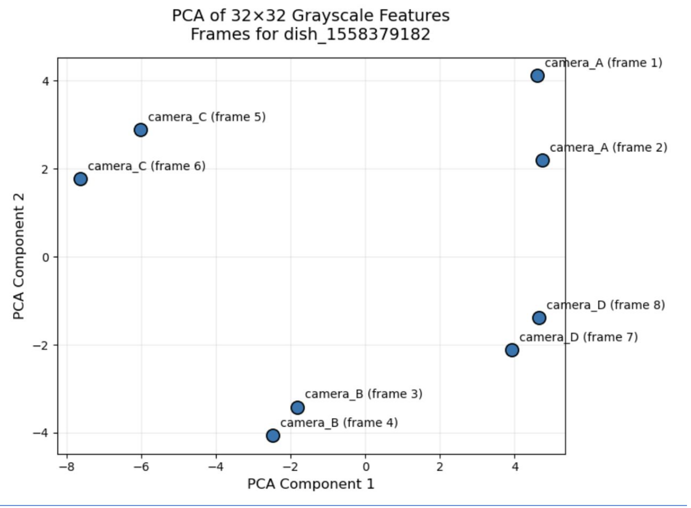
- 2 frames per camera angle extracted
- The dataset is increased by 8 times

Extracted Frames for dish_1558379182



- Captures Viewpoint and Lighting Variability
- Distinct Clusters
- Boosts Training Data

PCA of the extracted frames



2.2. Segmentation

⇒ Need to decide on a testing protocol to assess performance impact of segmentation vs. no segmentation.

⇒ Lots of trial and error to get something that works decently.

⇒ Final segmentation “pipeline” : **U^2-Net** *(Qin et al. 2020)* ⇒ cropping to include segmentation masks ⇒ blacking out outer pixels

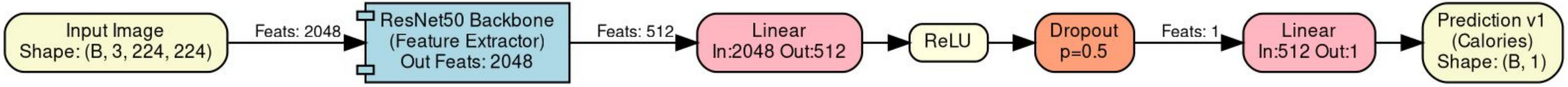
X. Qin et al., "U^2-Net: Going Deeper with Nested U-Structure for Salient Object Detection," arXiv:2005.09007v3 [cs.CV], 2022. Available at: https://arxiv.org/abs/2005.09007v3

2.2. Segmentation : Testing Protocol

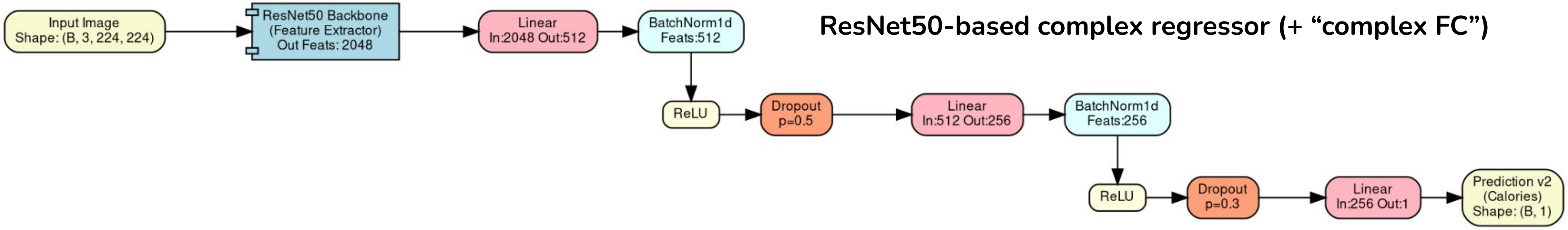
To assess the impact of the final segmentation strategy impact on performance and decide if we keep it for the subsequent parts, we tried it with multiple testing configurations (different architectures, estimating different target macronutrients).

- ResNet50 + simple FC | “calories” | overhead pictures only (example results below)
- ResNet50 + simple FC | “calories” | overhead + extracted pictures method 1
- ResNet50 + complex FC | “calories” | overhead + extracted pictures using method 1
- ResNet50 + simple FC | “all” | RAW | overhead pictures only

ResNet50-based simple regressor (+ “simple FC”)



ResNet50-based complex regressor (+ “complex FC”)

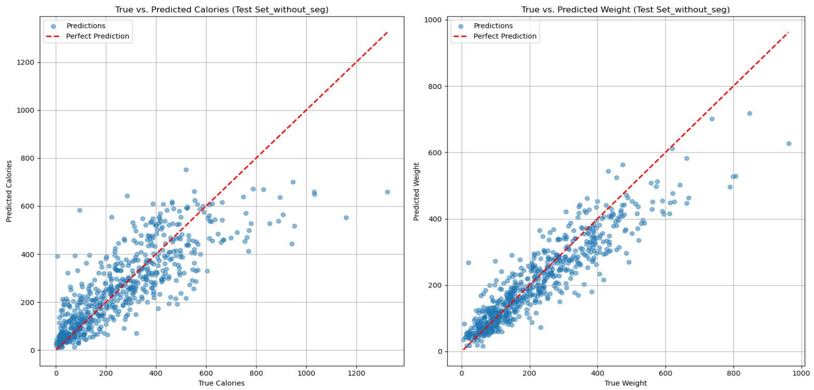


2.2. Segmentation : Testing Protocol

We noticed a performance drop across all configurations when adding segmentation.

Example results for the configuration:

ResNet50 + simple FC | “calories” | overhead pictures only



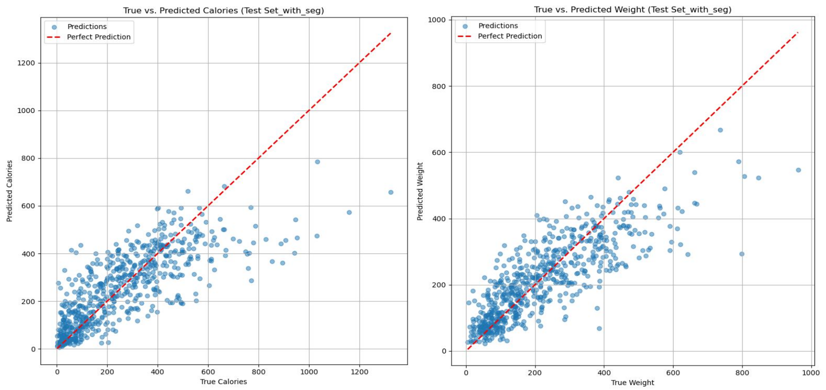


Table 1: Prediction Performance - without segmentation

Nutrient	MAE	MAE (%)	RMSE	R^2	Mean True	Mean Pred
Calories	**81.27**	**31.91**	**116.98**	**0.68**	254.71	250.89
Weight	**46.09**	**20.79**	**65.05**	**0.82**	221.65	208.10
Fat	7.25	58.20	10.27	0.43	12.45	15.11
Carbs	**9.54**	**49.40**	**12.98**	**0.34**	19.31	17.99
Protein	**9.59**	**50.77**	**14.22**	**0.55**	18.90	20.37

Table 2: Prediction Performance - with segmentation

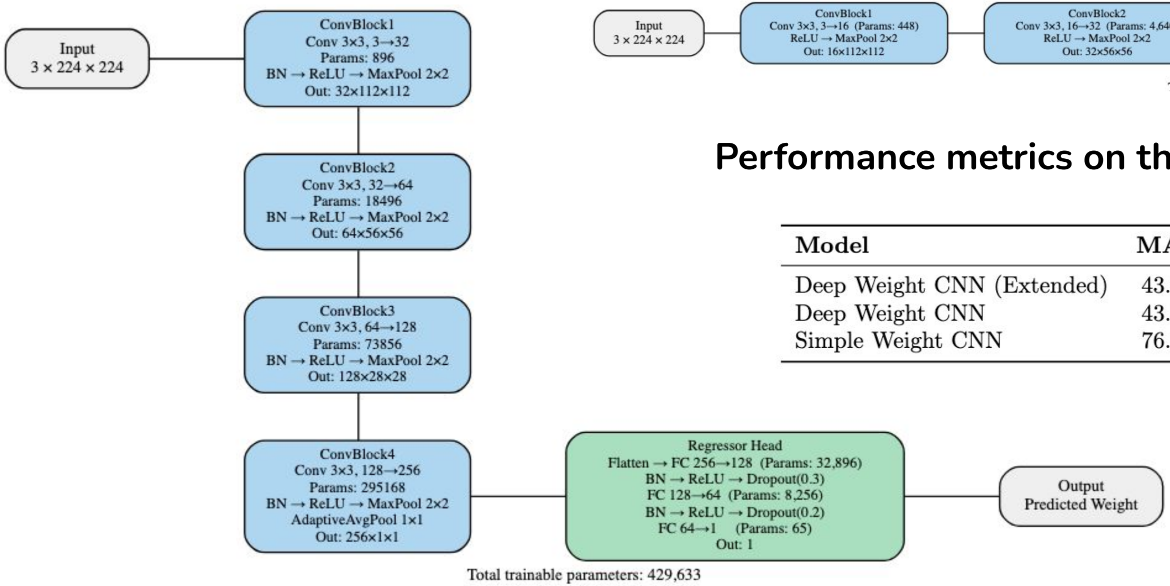
Nutrient	MAE	MAE (%)	RMSE	R^2	Mean True	Mean Pred
Calories	90.36	35.48	130.89	0.60	254.71	247.79
Weight	64.87	29.27	89.59	0.66	221.65	213.92
Fat	**6.45**	**51.81**	**9.95**	**0.46**	12.45	12.29
Carbs	10.44	54.04	14.05	0.23	19.31	18.49
Protein	10.12	53.54	15.29	0.48	18.90	18.63

Bold values indicate the best performance for each metric.

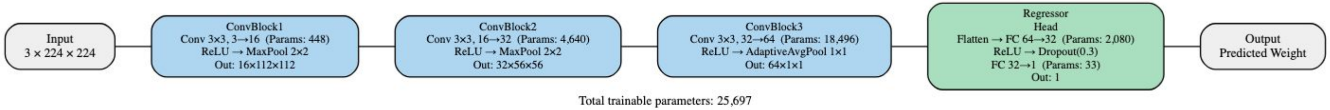
3. Weight Estimation

We build a model to estimate dish weights. We first validate the task with a 3-block Simple CNN with a 1-hidden-layer MLP head (~25 k params). Once validated, we upscale to a 4-block Deep CNN with a 2-hidden-layer MLP head (~430 k params), which cuts MAE by ~43% and lifts R^2 to ~0.84. Training on the frame-augmented dataset lowers the relative MAE to 18.7%, underscoring that more capacity + more data ⇒ better weight prediction.

More complex architecture: Deep CNN



Basic architecture: Simple CNN



Performance metrics on the original dataset and the extended one

Model	MAE	MAE (%)	RMSE	R^2	Mean True	Mean Pred
Deep Weight CNN (Extended)	43.89	18.68	63.12	0.82	234.93	229.17
Deep Weight CNN	43.54	21.31	57.82	0.84	204.27	207.99
Simple Weight CNN	76.11	37.26	103.91	0.49	204.27	202.27

4. Relative nutrition estimation

Table 1: Comparison of Nutrition Estimation Models (per 100g)

Model	MAE ↓	MAE (% Err) ↓	RMSE ↓	R^2 ↑
Calories (per 100g) — *True Mean: 152.8*				
ResNetPretrained	**22.899**	**14.988**	**38.056**	**0.850**
DeepConvNet	40.011	26.189	57.694	0.656
SimpleConvNet	41.068	26.881	60.153	0.626
MobileLikeNet	49.930	32.681	74.671	0.423
Fat (per 100g) — *True Mean: 8.6*				
ResNetPretrained	**2.130**	**24.854**	**3.421**	**0.838**
DeepConvNet	3.461	40.397	5.008	0.653
SimpleConvNet	3.654	42.646	5.417	0.593
MobileLikeNet	4.286	50.016	6.552	0.405
Carbohydrates (per 100g) — *True Mean: 11.0*				
ResNetPretrained	**2.683**	**24.396**	**4.254**	**0.787**
DeepConvNet	4.622	42.029	6.925	0.435
MobileLikeNet	5.616	51.067	8.645	0.119
SimpleConvNet	6.097	55.442	9.163	0.011
Protein (per 100g) — *True Mean: 9.5*				
ResNetPretrained	**2.175**	**22.997**	**3.125**	**0.747**
DeepConvNet	3.605	38.105	4.716	0.424
MobileLikeNet	4.114	43.494	5.372	0.252
SimpleConvNet	4.160	43.975	5.420	0.239

We train different models to predict the relative nutrition metrics for further combining with weight estimation model, to get the absolute values.

4

ResNet34
Pre-trained model to achieve the best performance

3

MobileNet
Lightweight MobileNet for applied user-case

2

Deep CNN
More layers, better performance

1

Simple CNN
Despite being simple, produces some nice results

5. Combined Model: Final Results

Combining the previous parts together (% nutrient x weight), we get a working pipeline for predicting absolute macronutrient content in grams.

Per 100g:
Calories: 174. 0 kcal
Fat: 8.33 g
Carbs: 10.12 g
Protein: 14.73 g

Weight: 167.0 g

Total:
Calories: 290.6 kcal
Fat: 13.9 g
Carbs: 16.9 g
Protein: 24.6 g
Weight: 167.0 g

Table 1: Performance metrics of combined model: Weight est. * Per100g est.

Nutrient	MAE	RMSE	R^2	% Err	Mean True	Mean Pred
Calories (abs)	87.495	128.939	0.607	28.552	306.448	304.311
Fat (abs)	6.045	8.975	0.519	37.583	16.084	15.858
Carbs (abs)	7.760	12.154	0.449	37.008	20.969	21.700
Protein (abs)	7.588	11.938	0.686	34.080	22.265	20.851
Weight (g)	49.084	74.165	0.758	21.339	230.023	226.192

Takeaways

- Segmentation accuracy could be improved by replacing the U^2-Net masks (generic saliency models miss food edges) by domain-specific food masks (e.g., FoodSeg103).

- Solution to the problem of not having enough samples: using the available videos, we were able to significantly increase the training set size and quality, leading to better generalization.

- We experimented and implemented powerful optimizations like batching, data caching, and distributed training to make the most out of the limited resources we had (time, GPU...).

- Insightful End-to-end ML pipeline experience: From data cleaning through architecture search and hyper-parameter tuning to rigorous MAE/R^2 evaluation, we iterated across 4 CNN families and cut weight-prediction error to 18.7 % relative MAE.