Predicting Nutritional Features via Image Deep Learning

Group 1-5:
Tony DiRubbo
Arsen Khanin
Jonah Komosinski
Rohan Ramesh Yadav

Syracuse University
IST 691: Deep Learning in Practice
Professor Christopher Dunham
12 December 2025

Project Overview

Accurate nutrition tracking remains a significant challenge in both personal health management and clinical dietary assessment. Many existing tools depend on manual text entry or barcode scanning, which slows down the logging process and often discourages consistent use. Recent progress in computer vision has produced highly accurate food classification models, but these systems primarily identify dish types rather than estimating the nutritional composition of a meal. As a result, they do not provide the quantitative information needed for calorie and macronutrient assessment.

This project investigates a deep learning approach that predicts nutritional values directly from food images. Our goal is to explore whether visual signals captured in photographs can serve as reliable indicators of calories, protein, fat, and carbohydrates. To support this objective, we utilize a dataset that connects food images with standardized nutritional information. We develop a transfer learning pipeline that improves upon a pre-trained model to classify food images into one of the selected dish categories. The predicted class is then mapped to its corresponding nutrient values, producing an estimate of the meal's caloric and macronutrient content. This method does not require depth sensors, weight measurements, or ingredient annotations. Instead, it relies on visual patterns learned from large scale image data.

The overall objective of this work is to build a foundation for accessible and automated nutrition analysis that operates on a single photograph. This contributes to ongoing research that connects computer vision with diet monitoring and offers insight into the feasibility and limitations of image based nutrient prediction in real world applications.

Goals of the Project/Pipeline

The primary goal of this project is to develop a model that can estimate nutritional information from a single food image. The model is designed to classify an image into a specific dish category and then pair to its associated calorie and macronutrient values. The prediction task focuses on four key continuous variables, which are estimated indirectly by mapping the predicted dish class to its average nutritional profile. Although the model does not compute nutrient values directly from raw pixels, it uses the dish classification to generate consistent estimates based on class level nutritional profiles. This approach evaluates whether visual recognition can support practical nutrient prediction at scale.

A second goal is to evaluate the feasibility of deploying this type of system in real world environments. This includes assessing the computational efficiency of the model, the robustness of its predictions across diverse food types, and its ability to generalize beyond the images seen during training. The project also investigates how well the model handles scenarios where portion size and ingredient variability are not explicitly represented in the data. These evaluations help determine whether the system can support practical applications in health tracking, dietary monitoring, and consumer nutrition tools.

A final goal is to establish a reproducible workflow for linking image datasets with external nutrient databases. This includes data selection, image preprocessing, nutrient aggregation, and model evaluation. By formalizing this workflow, the project provides a foundation that can support future extensions, including regression based nutrient prediction, portion size estimation, and more advanced multimodal architectures.

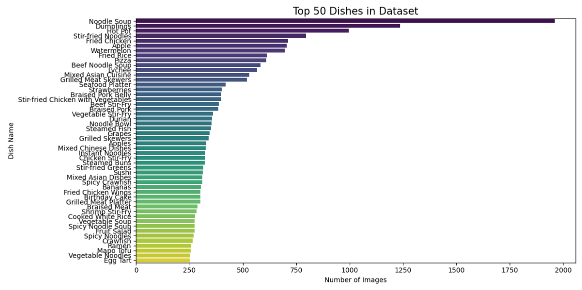
Data Exploration:

We began our exploration with the

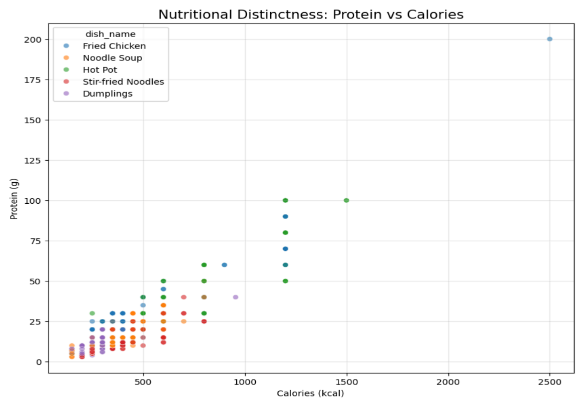
food-101 dataset for image labels and USDA dataset for nutritional value. The goal was to merge the images with associated nutritional value and use this new dataset to train the model. We were unable to get a good accuracy with this combined dataset and upon further investigation it was noticed that food-101 dataset has mislabeled entries which was perhaps responsible for this low score.

We decided to then proceed with a new dataset: MM-Food-100K (from Hugging Face), which comes with pre attributed nutritional information for the images. This dataset contains about 100k images with required metadata. To account for computational efficiency, we limited our focus to just 50 different dishes.

Our first figure displays the frequency for top 50 food classifications in our selected dataset and shows the dominance of east Asian dish in the dataset:



Our second figure is a Protein vs calorie visualization for the top 5 food classification demonstrating clustering which supports nutrition based inference.



Interesting Findings:

A notable finding from the project was how strongly the composition of the MM-Food-100K dataset appeared to influence the model's behavior. The dataset is dominated by East Asian dishes, and this heavy imbalance shaped what the model encountered during training. While we did not compute per-class accuracy scores, our qualitative observations during sample prediction suggested that the model handled East Asian dishes more reliably than Western dishes, which were either underrepresented or absent. Misclassifications were more common for foods outside the dominant categories, reinforcing how class imbalance can affect generalization even when overall training accuracy appears strong.

We also found that some nutritional traits in the dataset showed visible structure. The nutrient lookup table and exploratory scatter plots indicated that dishes with similar calorie or protein levels shared consistent visual patterns. This helped the class based nutrient estimation approach perform reasonably well despite its simplicity, since correct classifications mapped cleanly to meaningful average nutrient values. These observations highlighted how dataset structure and distribution shaped both the model's strengths and its limitations.

Methodology (See Notebook for Complete Coding Process)

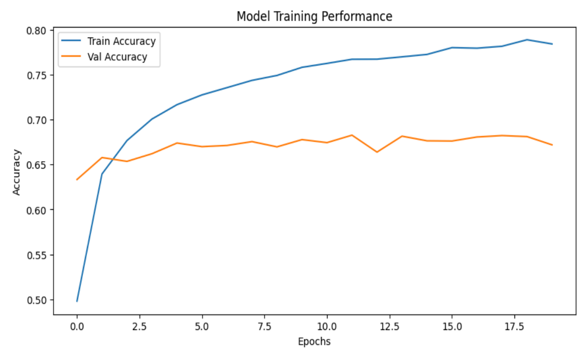
Our approach uses a two stage workflow that links food image classification with class based nutrient estimation. We prepared the MM-Food-100K dataset by parsing its nutritional fields, selecting the top 50 dish categories, and computing mean calories, protein, fat, and carbohydrates for use in a lookup table. Images were downloaded in parallel, validated, converted to RGB, and organized into class specific directories.

All images were resized to 224×224 and augmented with random flips and rotations to improve model robustness. We used a transfer learning architecture with a pretrained ResNet50 backbone, keeping its layers frozen and training a small classification head with pooling, dropout, and a softmax layer to distinguish among the selected dish categories.

The model was trained for 20 epochs using Adam and categorical cross entropy. Final nutrient predictions were generated by mapping the predicted dish label to its average nutritional profile, producing consistent calorie and macronutrient estimates derived from the lookup table.

Results

Our third figure is the training and validation accuracy across each of the 20 epochs, showing the relationship between training iterations and classification performance:



The final model produced solid performance on the food classification task given the visual complexity of the dataset. After 20 epochs, the model reached a training accuracy of 78.37 percent and a validation accuracy of 67.18 percent. The validation accuracy remained fairly stable, which suggests that the frozen ImageNet features combined with the added augmentation and classification layers were effective for distinguishing among the top 50 food categories.

Since nutrient estimates were generated by mapping each predicted class to its average nutritional profile, the accuracy of the nutrient predictions reflected the accuracy of the classification results. When the model identified a dish correctly, the estimated calories and macronutrients were close to the typical values for that dish. Misclassifications often produced estimates that were still near the correct range for visually similar items, although some errors were larger for dishes with ambiguous presentation or high variability.

The exploratory visualizations supported these findings. Dishes with higher protein levels or higher calorie density tended to form noticeable patterns in the data, which indicates that some nutritional traits have visual consistency across images. This helped the lookup table approach perform reasonably well even without direct nutrient regression.

Overall, the model delivered consistent class based nutrient estimates for many dishes and demonstrated that this approach is both practical and computationally efficient. While the system does not yet handle portion size or variation within a class, the results show clear potential for more advanced image driven nutrition analysis in future work.

Problems Encountered

Two main challenges had a significant impact on the development of our model and workflow. Both challenges involved the data that we trained our model on. The first major issue involved the need to switch datasets during the early stages of the project. Our initial attempt to merge the Food101 dataset with USDA nutritional values led to poor classification performance. Further inspection showed that Food101 contains mislabeled and inconsistent images, which introduced noise and prevented the model from learning reliable patterns. This forced us to move away from the combined dataset and adopt a new dataset that already included nutritional information. The shift required rebuilding our data pipeline, adjusting preprocessing steps, and recreating our exploratory visualizations.

The second challenge involved the composition of our newly chosen MM-Food-100K dataset. The dataset is heavily dominated by East Asian dishes, which limited the diversity of classes available for training. Many common Western dishes were underrepresented or missing, which reduced the generalizability of the final model. As a result, the system performs best on foods that reflect the distribution of the dataset and may be less accurate when applied to dishes outside that domain.

Thoughts on Assignment

The project was successful in meeting its primary goal of predicting nutritional information from a single food image. The model reached a training accuracy of 78.37 percent and a validation accuracy of 67.18 percent, which showed that the transfer learning approach was effective for many of the dish categories in the dataset. When the model correctly identified a dish, the linked calorie and macronutrient values were consistent with the expected nutritional profile for that item. This confirmed that class based nutrient estimation can provide a workable approximation without requiring portion size measurements or ingredient lists.

At the same time, the results showed clear limitations. Misclassifications directly affected the nutrient estimates because the system assigns nutrients based entirely on the predicted dish label. Foods that looked visually similar or had high internal variation were harder for the model to distinguish, which reduced the accuracy of both the classification and the nutritional output. The strong skew toward East Asian dishes in the dataset also constrained the generalizability of the system, since foods from underrepresented categories were more likely to be misclassified.

Overall, the project achieved its core prediction and inference goals and demonstrated that image based nutrient estimation is technically feasible with a class driven pipeline. The system produced reliable estimates for many images and highlighted the importance of dataset diversity and careful preprocessing. Completing this work as a full end to end deep learning project, from data collection to final prediction, showed how each stage contributes to model performance and provided practical experience with building and evaluating a complete machine learning workflow.

Works Cited

Bianco, Rachele, et al. “2D Prediction of the Nutritional Composition of Dishes from Food Images: Deep Learning Algorithm Selection and Data Curation Beyond the Nutrition5k Project.” *Nutrients*, vol. 17, no. 3, 2025, p. 548. PubMed, https://pubmed.ncbi.nlm.nih.gov/40647299/.

Bossard, Lukas, Matthieu Guillaumin, and Luc Van Gool. “Food-101: Mining Discriminative Components with Random Forests.” *European Conference on Computer Vision*, 2014, pp. 446–461. Springer.

Codatta. *MM-Food-100K*. Hugging Face, n.d., https://huggingface.co/datasets/Codatta/MM-Food-100K.

ETH Zürich, Computer Vision Lab. *Food-101 Dataset*. 2014, https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/.

Philip, Adebayo Rotimi. “Optimizing Food101 Classification with Transfer Learning: A Fine-Tuning Approach Using EfficientNetB0.” *International Journal of Intelligent Information Systems*, vol. 13, no. 4, 2024, pp. 65–72. Science Publishing Group, https://www.sciencepublishinggroup.com/article/10.11648/j.ijiis.20241304.11.

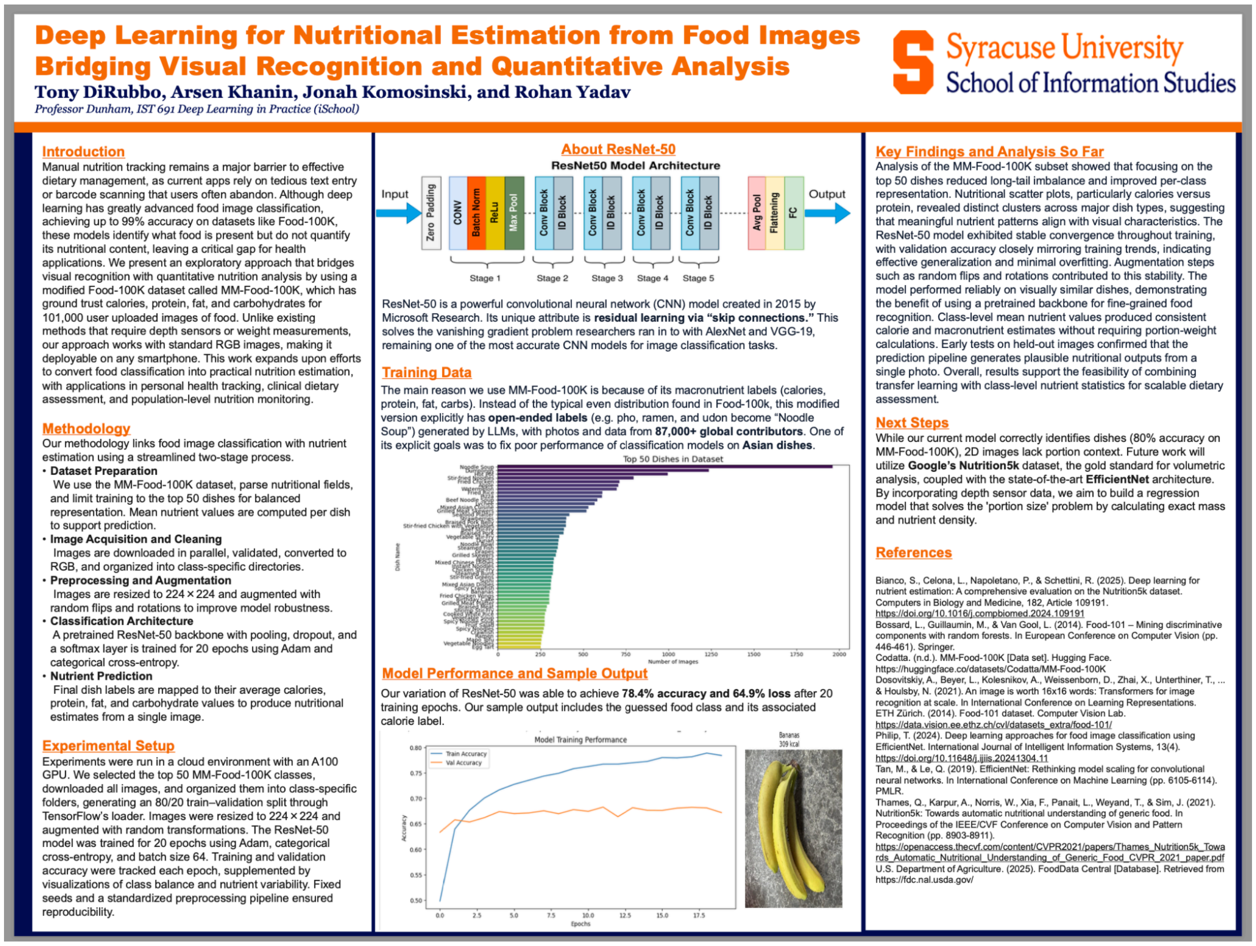
Tan, Mingxing, and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” *Proceedings of the International Conference on Machine Learning*, 2019, pp. 6105–6114. PMLR.

Thames, Quinn, et al. “Nutrition5k: Towards Automatic Nutritional Understanding of Generic Food.” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8903–8911. https://openaccess.thecvf.com/content/CVPR2021/papers/Thames_Nutrition5k_Towards_Automatic_Nutritional_Understanding_of_Generic_Food_CVPR_2021_paper.pdf.

United States Department of Agriculture. *FoodData Central*. 2025, https://fdc.nal.usda.gov/.

Appendix:

Poster presented on 5th of December 2025, one week before submissions



Deep Learning for Nutritional Estimation from Food Images Bridging Visual Recognition and Quantitative Analysis

Tony DiRubbo, Arsen Khanin, Jonah Komosinski, and Rohan Yadav

Professor Dunham, IST 691 Deep Learning in Practice (iSchool)

Syracuse University School of Information Studies

Introduction

Manual nutrition tracking remains a major barrier to effective dietary management, as current apps rely on tedious text entry or barcode scanning that users often abandon. Although deep learning has greatly advanced food image classification, achieving up to 99% accuracy on datasets like Food-100K, these models identify what food is present but do not quantify its nutritional content, leaving a critical gap for health applications. We present an exploratory approach that bridges visual recognition with quantitative nutrition analysis by using a modified Food-100K dataset called MM-Food-100K, which has ground trust calories, protein, fat, and carbohydrates for 101,000 user uploaded images of food. Unlike existing methods that require depth sensors or weight measurements, our approach works with standard RGB images, making it deployable on any smartphone. This work expands upon efforts to convert food classification into practical nutrition estimation, with applications in personal health tracking, clinical dietary assessment, and population-level nutrition monitoring.

Methodology

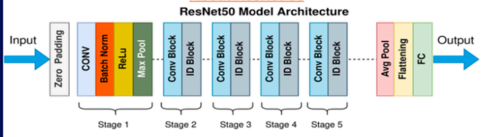
Our methodology links food image classification with nutrient estimation using a streamlined two-stage process.

- **Dataset Preparation**
  We use the MM-Food-100K dataset, parse nutritional fields, and limit training to the top 50 dishes for balanced representation. Mean nutrient values are computed per dish to support prediction.
- **Image Acquisition and Cleaning**
  Images are downloaded in parallel, validated, converted to RGB, and organized into class-specific directories.
- **Preprocessing and Augmentation**
  Images are resized to 224 × 224 and augmented with random flips and rotations to improve model robustness.
- **Classification Architecture**
  A pretrained ResNet-50 backbone with pooling, dropout, and a softmax layer is trained for 20 epochs using Adam and categorical cross-entropy.
- **Nutrient Prediction**
  Final dish labels are mapped to their average calories, protein, fat, and carbohydrate values to produce nutritional estimates from a single image.

Experimental Setup

Experiments were run in a cloud environment with an A100 GPU. We selected the top 50 MM-Food-100K classes, downloaded all images, and organized them into class-specific folders, generating an 80/20 train–validation split through TensorFlow's loader. Images were resized to 224 × 224 and augmented with random transformations. The ResNet-50 model was trained for 20 epochs using Adam, categorical cross-entropy, and batch size 64. Training and validation accuracy were tracked each epoch, supplemented by visualizations of class balance and nutrient variability. Fixed seeds and a standardized preprocessing pipeline ensured reproducibility.

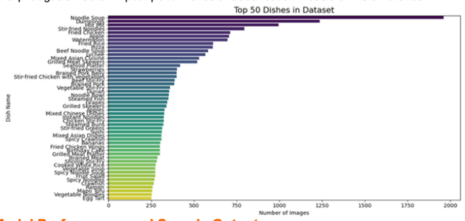
About ResNet-50



ResNet-50 is a powerful convolutional neural network (CNN) model created in 2015 by Microsoft Research. Its unique attribute is **residual learning via "skip connections."** This solves the vanishing gradient problem researchers ran in to with AlexNet and VGG-19, remaining one of the most accurate CNN models for image classification tasks.

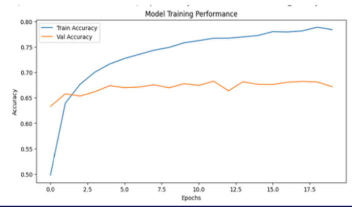
Training Data

The main reason we use MM-Food-100K is because of its macronutrient labels (calories, protein, fat, carbs). Instead of the typical even distribution found in Food-100k, this modified version explicitly has **open-ended labels** (e.g. pho, ramen, and udon become "Noodle Soup") generated by LLMs, with photos and data from **87,000+ global contributors**. One of its explicit goals was to fix poor performance of classification models on **Asian dishes**.



Model Performance and Sample Output

Our variation of ResNet-50 was able to achieve **78.4% accuracy and 64.9% loss** after 20 training epochs. Our sample output includes the guessed food class and its associated calorie label.



Key Findings and Analysis So Far

Analysis of the MM-Food-100K subset showed that focusing on the top 50 dishes reduced long-tail imbalance and improved per-class representation. Nutritional scatter plots, particularly calories versus protein, revealed distinct clusters across major dish types, suggesting that meaningful nutrient patterns align with visual characteristics. The ResNet-50 model exhibited stable convergence throughout training, with validation accuracy closely mirroring training trends, indicating effective generalization and minimal overfitting. Augmentation steps such as random flips and rotations contributed to this stability. The model performed reliably on visually similar dishes, demonstrating the benefit of using a pretrained backbone for fine-grained food recognition. Class-level mean nutrient values produced consistent calorie and macronutrient estimates without requiring portion-weight calculations. Early tests on held-out images confirmed that the prediction pipeline generates plausible nutritional outputs from a single photo. Overall, results support the feasibility of combining transfer learning with class-level nutrient statistics for scalable dietary assessment.

Next Steps

While our current model correctly identifies dishes (80% accuracy on MM-Food-100K), 2D images lack portion context. Future work will utilize **Google's Nutrition5k** dataset, the gold standard for volumetric analysis, coupled with the state-of-the-art **EfficientNet** architecture. By incorporating depth sensor data, we aim to build a regression model that solves the 'portion size' problem by calculating exact mass and nutrient density.

References

Bianco, S., Celona, L., Napoletano, P., & Schettini, R. (2025). Deep learning for nutrient estimation: A comprehensive evaluation on the Nutrition5k dataset. Computers in Biology and Medicine, 182, Article 109191. https://doi.org/10.1016/j.compbiomed.2024.109191
Bossard, L., Guillaumin, M., & Van Gool, L. (2014). Food-101 – Mining discriminative components with random forests. In European Conference on Computer Vision (pp. 446-461). Springer.
Codatta. (n.d.). MM-Food-100K [Data set]. Hugging Face. https://huggingface.co/datasets/Codatta/MM-Food-100K
Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations.
ETH Zürich. (2014). Food-101 dataset. Computer Vision Lab. https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/
Philip, T. (2024). Deep learning approaches for food image classification using EfficientNet. International Journal of Intelligent Information Systems, 13(4). https://doi.org/10.11648/j.ijiis.20241304.11
Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). PMLR.
Thames, Q., Karpur, A., Norris, W., Xia, F., Panait, L., Weyand, T., & Sim, J. (2021). Nutrition5k: Towards automatic nutritional understanding of generic food. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8903-8911). https://openaccess.thecvf.com/content/CVPR2021/papers/Thames_Nutrition5k_Towards_Automatic_Nutritional_Understanding_of_Generic_Food_CVPR_2021_paper.pdf
U.S. Department of Agriculture. (2025). FoodData Central [Database]. Retrieved from https://fdc.nal.usda.gov/

Introduction

This notebook contains code for **"Predicting Nutritional Features via Image Deep Learning"** project completed by Tony DiRubbo, Arsen Khanin, Jonah Komosinski, and Rohan Ramesh Yadav for "Deep Learning in Practice" class. Notebook is organized into the following sections: Setup, Loading the Dataset, Downloading Images, Defining Datasets and Model, Training, and Sample Prediction.

Setup

Here we will just install and import all the dependencies. Additionally, we set `TOP_N_CLASSES` to 50 to speed up training, `IMG_SIZE` is set to 224x224 pixels, `EPOCHS` is kept low as it takes a while per epoch to train.

```
!pip install datasets tensorflow matplotlib seaborn pandas requests tqd
```

```
import os
import json
import requests
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, models, applications
from tensorflow.keras.preprocessing import image_dataset_from_directory
from datasets import load_dataset
from concurrent.futures import ThreadPoolExecutor
from tqdm import tqdm
from io import BytesIO
from PIL import Image

# CONFIGURATION
TOP_N_CLASSES = 50  # Limit to top 50 dishes for better accuracy & spee
IMG_SIZE = (224, 224)
BATCH_SIZE = 64
EPOCHS = 20
```

Load Dataset

Here we just load the dataset which contains image URLs as well as nutritional information. We keep only `TOP_N_CLASSES` food classes and create a `lookup_table` that contains average nutritional values per food class.

Additionally we plot some graphs to learn more about the data.

```
print("Loading dataset metadata...")
# Load only the text data first (instant)
ds = load_dataset("Codatta/MM-Food-100K", split="train")
df = ds.to_pandas()

# Helper to parse the nutrient string into columns
def parse_nutrients(row):
    try:
        data = eval(row['nutritional_profile']) if isinstance(row['nu
        return pd.Series([data.get('calories_kcal', 0), data.get('pro
    except:
        return pd.Series([0, 0, 0, 0])

print("Parsing nutritional info...")
df[['cal', 'prot', 'fat', 'carb']] = df.apply(parse_nutrients, axis=1

# Filter for Top N Classes
top_classes = df['dish_name'].value_counts().head(TOP_N_CLASSES).inde
filtered_df = df[df['dish_name'].isin(top_classes)].copy()

print(f"Filtered dataset from {len(df)} to {len(filtered_df)} images

# --- POSTER PLOT 1: Class Distribution ---
plt.figure(figsize=(12, 6))
sns.countplot(y=filtered_df['dish_name'], order=top_classes, palette='
plt.title(f"Top {TOP_N_CLASSES} Dishes in Dataset", fontsize=15)
plt.xlabel("Number of Images")
plt.ylabel("Dish Name")
plt.tight_layout()
plt.savefig("poster_class_dist.png", dpi=300)
plt.show()

# --- POSTER PLOT 2: Nutrient Scatter (Protein vs Calories) ---
# Pick top 5 classes for a clean scatter plot
top_5 = top_classes[:5]
subset = filtered_df[filtered_df['dish_name'].isin(top_5)]
plt.figure(figsize=(10, 8))
sns.scatterplot(data=subset, x='cal', y='prot', hue='dish_name', alph
plt.title("Nutritional Distinctness: Protein vs Calories", fontsize=1
plt.xlabel("Calories (kcal)")
plt.ylabel("Protein (g)")
plt.grid(True, alpha=0.3)
plt.savefig("poster_nutrient_scatter.png", dpi=300)
plt.show()

# --- CREATE LOOKUP TABLE ---
# We calculate the MEAN nutrients for each dish
lookup_table = filtered_df.groupby('dish_name')[['cal', 'prot', 'fat'
with open('nutrient_lookup.json', 'w') as f:
    json.dump(lookup_table, f)
print("✅ Lookup table saved to nutrient_lookup.json")
```

```
Loading dataset metadata...
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your sett
You will be able to reuse this secret in all of your notebooks
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to ac
  warnings.warn(
```

README.md: 14.6k/? [00:00<00:00, 1.54MB/s]

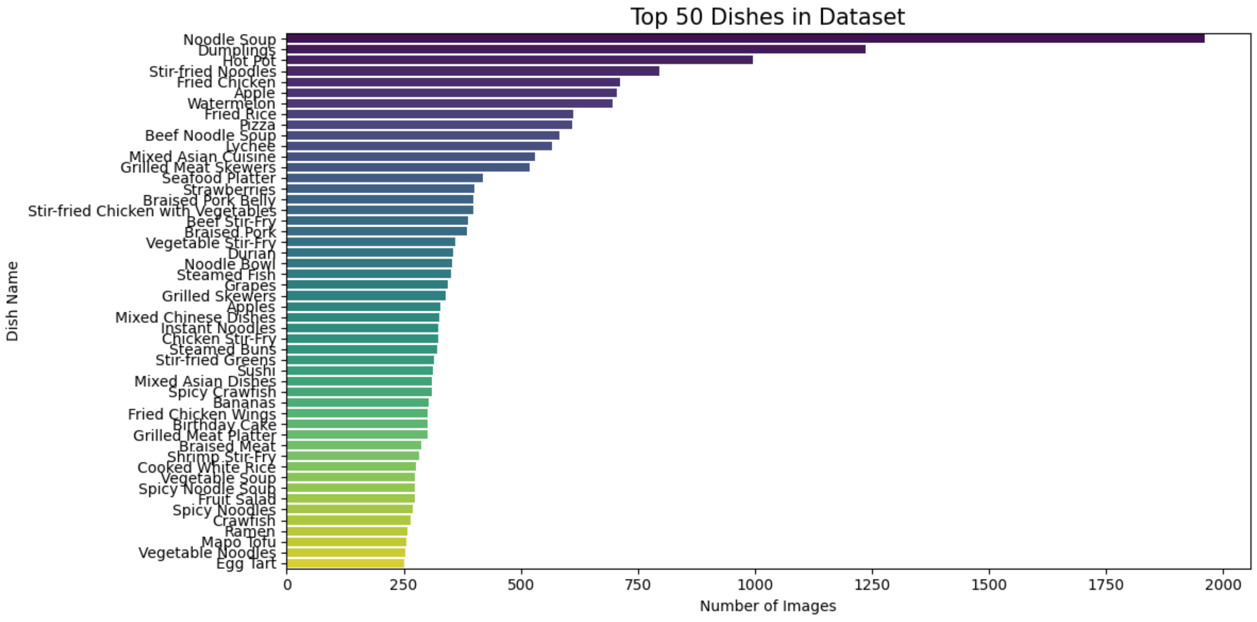
MM-Food-100K.csv: 100% 28.6M/28.6M [00:03<00:00, 7.42MB/s]

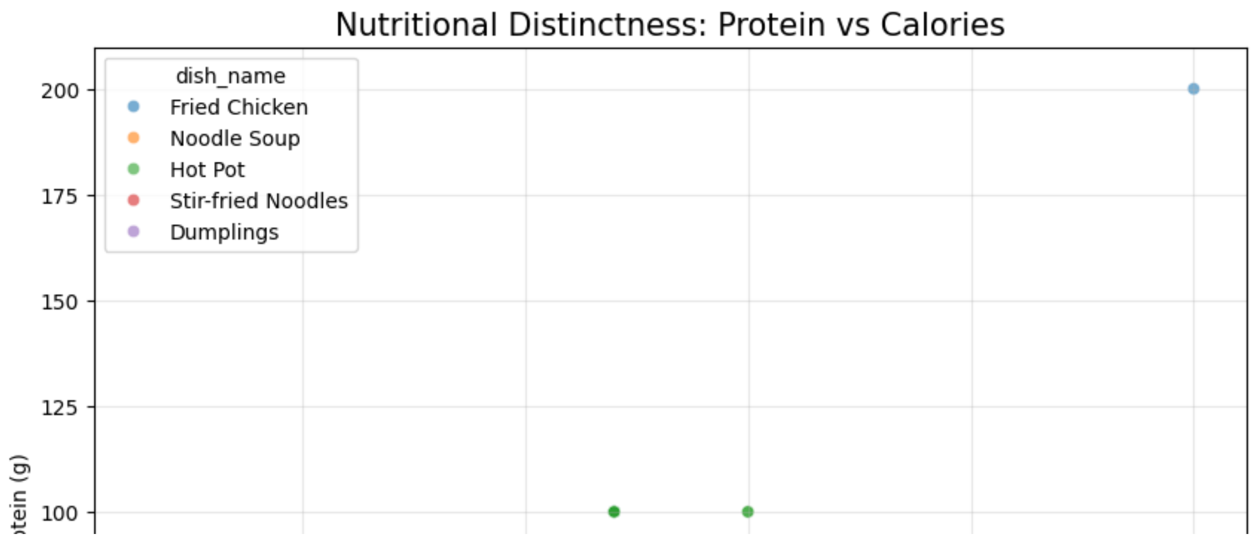
Generating train split: 100% 100000/100000 [00:00<00:00, 191021.22 examples/s]

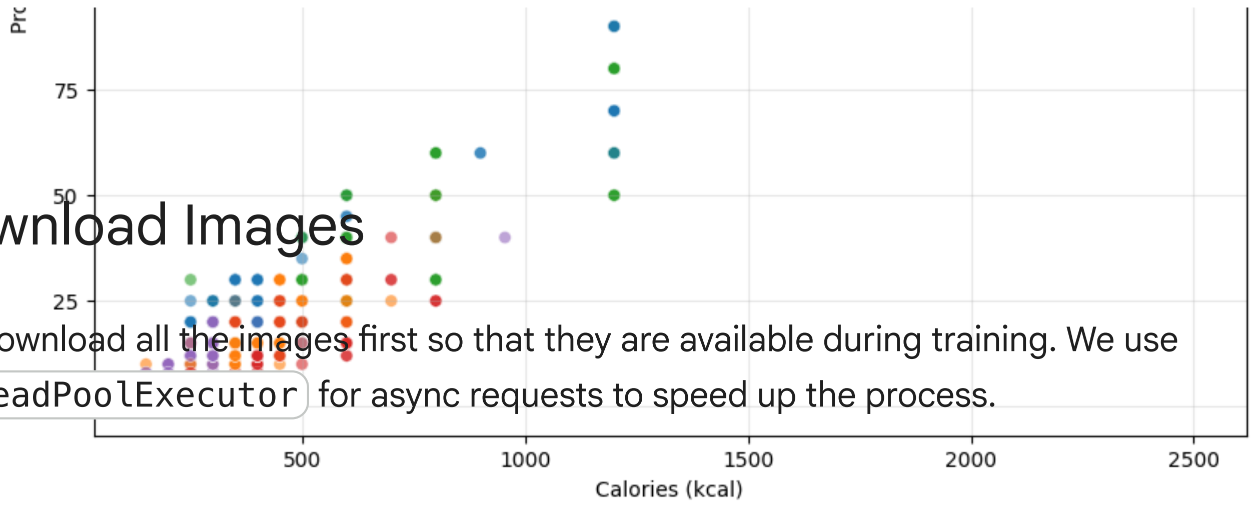
```
Parsing nutritional info...
Filtered dataset from 100000 to 22312 images (50 classes).
/tmp/ipython-input-2380338336.py:25: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be rem

  sns.countplot(y=filtered_df['dish_name'], order=top_classes, palette=
```







```
✅ Lookup table saved to nutrient_lookup.json
```

Download Images

We download all the images first so that they are available during training. We use `ThreadPoolExecutor` for async requests to speed up the process.

```
def download_and_save(row):
    url = row['image_url']
    label = row['dish_name']

    # Create folder
    folder_path = os.path.join("food_dataset", label)
    os.makedirs(folder_path, exist_ok=True)

    # Hash URL for unique filename
    filename = str(abs(hash(url))) + ".jpg"
    file_path = os.path.join(folder_path, filename)

    if os.path.exists(file_path):
        return

    try:
        response = requests.get(url, timeout=5)
        if response.status_code == 200:
            # Verify it's an image
            img = Image.open(BytesIO(response.content))
            img = img.convert('RGB')
            img.save(file_path, "JPEG", quality=85)
    except:
        pass

print("Starting download... (This will take a few minutes)")
# Convert dataframe to list of dicts for threading
rows = filtered_df.to_dict('records')

with ThreadPoolExecutor(max_workers=64) as executor:
    list(tqdm(executor.map(download_and_save, rows), total=len(rows)))

print("✅ Download complete!")
```

```
Starting download... (This will take a few minutes)
 38%|████      | 8412/22312 [03:36<04:45, 48.68it/s]/usr/local/lib/pyt
  warnings.warn(
100%|██████████| 22312/22312 [09:09<00:00, 40.63it/s]
✅ Download complete!
```

Define Datasets & Model

First we define our training and validation datasets. Next we utilize transfer learning technique with ResNet 50 as the base model. We add data augmentation (random flip + rotation) and a drop out layer to increase robustness and decrease overfitting.

```
# 1. Create Datasets
train_ds = image_dataset_from_directory(
    "food_dataset",
    validation_split=0.2,
    subset="training",
    seed=1337,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    label_mode='categorical' # Important for >2 classes
)

val_ds = image_dataset_from_directory(
    "food_dataset",
    validation_split=0.2,
    subset="validation",
    seed=1337,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    label_mode='categorical'
)

class_names = train_ds.class_names
print(f"Classes found: {class_names}")

# 2. Performance Tuning
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.prefetch(buffer_size=AUTOTUNE)

# 3. Build Model (ResNet50 Transfer Learning)
base_model = applications.ResNet50(
    include_top=False,
    weights='imagenet',
    input_shape=IMG_SIZE + (3,)
)
```

```
base_model.trainable = False  # Freeze base initially

inputs = layers.Input(shape=IMG_SIZE + (3,))
# Augmentation
x = layers.RandomFlip("horizontal")(inputs)
x = layers.RandomRotation(0.1)(x)

# Preprocessing & Base
x = applications.resnet50.preprocess_input(x)
x = base_model(x, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(len(class_names), activation='softmax')(x)

model = models.Model(inputs, outputs)
model.compile(optimizer='adam', loss='categorical_crossentropy', metr

model.summary()
```

```
Found 22309 files belonging to 50 classes.
Using 17848 files for training.
Found 22309 files belonging to 50 classes.
Using 4461 files for validation.
Classes found: ['Apple', 'Apples', 'Bananas', 'Beef Noodle Soup', 'Beef
Downloading data from https://storage.googleapis.com/tensorflow/keras-a
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
Model: "functional"
```

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0	-
random_flip (RandomFlip)	(None, 224, 224, 3)	0	input_layer_1
random_rotation (RandomRotation)	(None, 224, 224, 3)	0	random_flip[0
get_item (GetItem)	(None, 224, 224)	0	random_rotati
get_item_1 (GetItem)	(None, 224, 224)	0	random_rotati
get_item_2 (GetItem)	(None, 224, 224)	0	random_rotati
stack (Stack)	(None, 224, 224, 3)	0	get_item[0][0 get_item_1[0] get_item_2[0]

```
│                           │                            │                  │ get_item_2[0]
├───────────────────────────┼────────────────────────────┼──────────────────┼──────────────
│ add (Add)                 │ (None, 224, 224,           │                0 │ stack[0][0]
│                           │ 3)                         │                  │
├───────────────────────────┼────────────────────────────┼──────────────────┼──────────────
│ resnet50                  │ (None, 7, 7,               │       23,587,712 │ add[0][0]
│ (Functional)              │ 2048)                      │                  │
├───────────────────────────┼────────────────────────────┼──────────────────┼──────────────
│ global_average_poo…       │ (None, 2048)               │                0 │ resnet50[0][0
│ (GlobalAveragePool…       │                            │                  │
├───────────────────────────┼────────────────────────────┼──────────────────┼──────────────
│ dropout (Dropout)         │ (None, 2048)               │                0 │ global_averag
├───────────────────────────┼────────────────────────────┼──────────────────┼──────────────
│ dense (Dense)             │ (None, 50)                 │          102,450 │ dropout[0][0]
└───────────────────────────┴────────────────────────────┴──────────────────┴──────────────

 Total params: 23,690,162 (90.37 MB)
 Trainable params: 102,450 (400.20 KB)
 Non-trainable params: 23,587,712 (89.98 MB)
```

Train the model

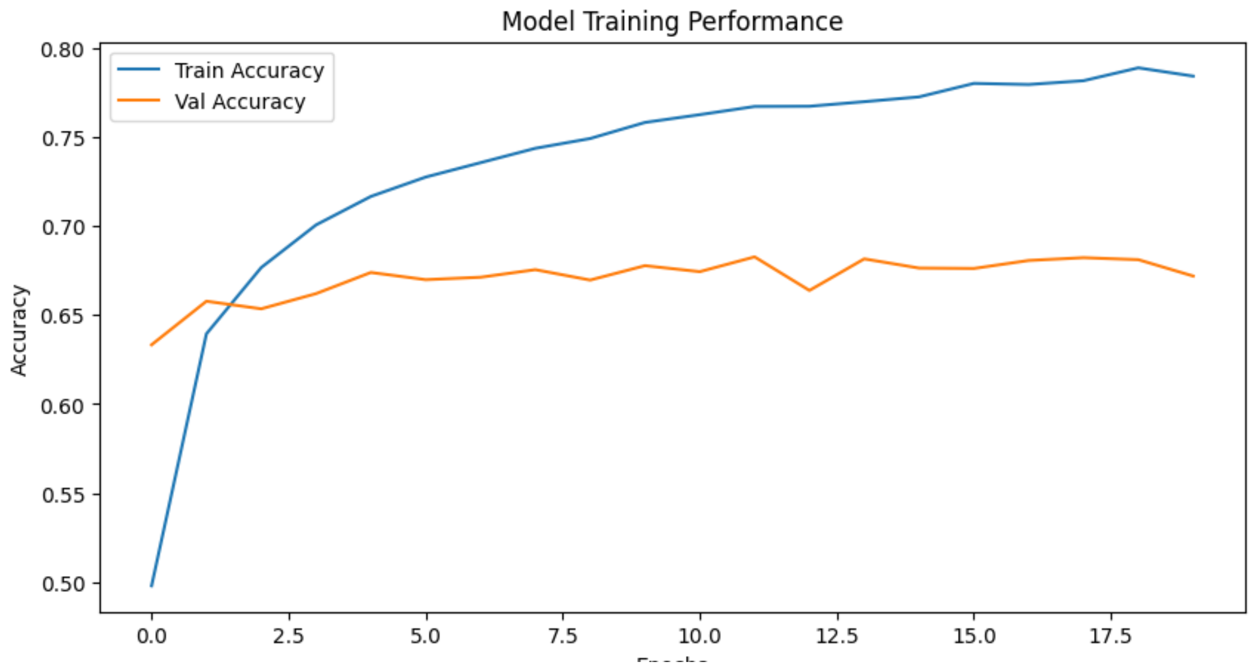
Here we just wait for the model to train. After training we plot train & validation accuracies to see how the model converges.

```python
history = model.fit(train_ds, validation_data=val_ds, epochs=EPOCHS)

# --- POSTER PLOT 3: Accuracy Curves ---
plt.figure(figsize=(10, 5))
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.title("Model Training Performance")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.savefig("poster_training_curve.png")
plt.show()
```

```
Epoch 1/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 91s 285ms/step - accuracy: 0.3814 - loss:
Epoch 2/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 77s 274ms/step - accuracy: 0.6275 - loss:
Epoch 3/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 77s 274ms/step - accuracy: 0.6752 - loss:
Epoch 4/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 77s 274ms/step - accuracy: 0.6992 - loss:
```

```
Epoch 5/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 77s 274ms/step - accuracy: 0.7097 - loss:
Epoch 6/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 76s 274ms/step - accuracy: 0.7223 - loss:
Epoch 7/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 77s 274ms/step - accuracy: 0.7367 - loss:
Epoch 8/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 76s 274ms/step - accuracy: 0.7399 - loss:
Epoch 9/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 77s 276ms/step - accuracy: 0.7484 - loss:
Epoch 10/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 76s 274ms/step - accuracy: 0.7539 - loss:
Epoch 11/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 77s 275ms/step - accuracy: 0.7608 - loss:
Epoch 12/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 76s 274ms/step - accuracy: 0.7697 - loss:
Epoch 13/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 76s 274ms/step - accuracy: 0.7701 - loss:
Epoch 14/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 76s 273ms/step - accuracy: 0.7651 - loss:
Epoch 15/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 77s 274ms/step - accuracy: 0.7696 - loss:
Epoch 16/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 76s 274ms/step - accuracy: 0.7762 - loss:
Epoch 17/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 76s 274ms/step - accuracy: 0.7747 - loss:
Epoch 18/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 76s 274ms/step - accuracy: 0.7829 - loss:
Epoch 19/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 76s 274ms/step - accuracy: 0.7830 - loss:
Epoch 20/20
279/279 ━━━━━━━━━━━━━━━━━━━━ 76s 274ms/step - accuracy: 0.7837 - loss:
```



Sample Prediction

Finally, we define a helper function `predict_food`, which takes an image path as an argument, reads the image, uses our model from previous step to predict the food class, uses `lookup_table` defined earlier to get nutritional information for the predicted food class.

```
def predict_food(image_path):
    img = tf.keras.preprocessing.image.load_img(image_path, target_si
    img_array = tf.keras.preprocessing.image.img_to_array(img)
    img_array = tf.expand_dims(img_array, 0) # Create batch axis

    predictions = model.predict(img_array)
    score = tf.nn.softmax(predictions[0])
    predicted_class = class_names[np.argmax(predictions)]
    confidence = 100 * np.max(score)

    # LOOKUP NUTRITION
    nutrients = lookup_table.get(predicted_class, {"cal":0, "prot":0,

    print(f"🍽️ Dish: {predicted_class} ({confidence:.2f}% confidence)
    print(f"🔥 Calories: {nutrients['cal']:.0f}")
    print(f"💪 Protein: {nutrients['prot']:.1f}g")

    # Display image
    plt.imshow(img)
    plt.axis("off")
    plt.title(f"{predicted_class}\n{nutrients['cal']:.0f} kcal")
    plt.show()

# Run on a sample
sample_image = "food_dataset/" + class_names[2] + "/" + os.listdir("f
predict_food(sample_image)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
🍽️ Dish: Bananas (5.26% confidence)
🔥 Calories: 309
💪 Protein: 3.7g
```