

PROJECT REPORT : TASTELENS

TasteLens is an AI-driven system that identifies dishes from images, including Moroccan cuisine, and provides detailed nutritional facts.

[Rayane Lalaoui Hassani]

[Anass Amini]

[Oussama Mellakh]

[Moro Abderrahman]

15/01/2025



Table of contents

Problematic.....	2
Project objective	2
System Architecture	2
WORKFLOW AND SYSTEM OPERATION	3
TOOLS AND TECHNOLOGIES.....	6
EfficientNet-B3 Training Workflow.....	6
Key features	8
STRENGTHS AND LIMITATIONS	8
FUTURE IMPROVEMENTS	9
Conclusion	9

PROBLEMATIC

Analyzing the nutritional value of dishes is a complex task, quite prolonged, and can also hold the risk of human error while being done manually. Such challenges are even more pronounced with regards to certain specific cuisines such as Moroccan cuisine, where nutritional data is often scanty or dispersed. Besides, users have no access to any automated tool to provide them with well-laid facts from simple images of dishes.

Proposed solution: Create an interactive, automated system that identifies dishes from images and automatically extracts nutritional data from reliable bases, presenting them in a clear, intuitive format.

PROJECT OBJECTIVE

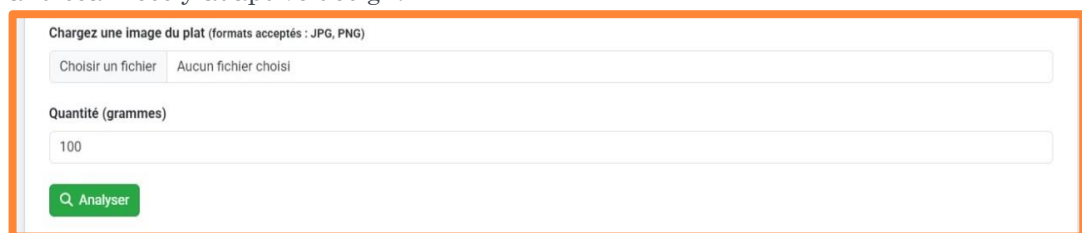
1. Dish identification: Recognize dishes using an image precisely with the help of deep learning model (EfficientNet-B3).
2. Nutritional analysis: Nutrition information will be obtained from a reliable database like MySQL Azure and World Open Food Facts.
3. Enhanced Results: Using a language model (Gemini API) to turn results into clear text responses to the user.
4. Intuitive Interface: Provide a seamless user experience through a platform that is easily accessible

SYSTEM ARCHITECTURE

The system has a modular architecture and interconnected architecture and divides into functional components:

Frontend:

- A web user interface allowing users to upload pictures and visualize results.
- Developed with HTML/CSS and JavaScript enlivened with Bootstrap for modern and seamlessly adaptive design.



The screenshot shows a web form with the following elements:

- A header text: "Chargez une image du plat (formats acceptés : JPG, PNG)"
- A file selection area with a button "Choisir un fichier" and a status "Aucun fichier choisi".
- A label "Quantité (grammes)" above a text input field containing the value "100".
- A green button with a magnifying glass icon and the text "Analyser".

Figure 1 : Frontend before inserting a food photo 1

Backend:

- A flask application to address frontend communication with the deep learning model, databases, and LLM.
- Ensured orchestration of requests for smooth analysis.

```
# =====
# 1) Single-headed (Moroccan) model setup
# =====
MODEL_PATH = "model.pth" # Path to your checkpoint
NUM_OUTPUT_CLASSES = 26 # e.g., 26 Moroccan classes

# Define Transformations
transform = transforms.Compose([
    transforms.Resize((384, 384)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406],
                          [0.229, 0.224, 0.225])
])

def create_model(num_classes):
    """Create an EfficientNet-B3 with a single classification head."""
    weights = EfficientNet_B3_Weights.IMAGENET1K_V1
    model = EfficientNet_B3.from_pretrained(weights)
    in_features = model.classifier[1].in_features
    model.classifier = nn.Sequential(
        nn.Dropout(0.4),
        nn.Linear(in_features, 256),
        nn.ReLU(),
        nn.Linear(256, num_classes)
    )
    return model

# Load the raw checkpoint
raw_checkpoint = torch.load(MODEL_PATH, map_location="cpu")
# Sometimes the checkpoint includes extra keys like "classes" or "model_state_dict"
if isinstance(raw_checkpoint, dict):
    if "model_state_dict" in raw_checkpoint:
        state_dict = raw_checkpoint["model_state_dict"]
        # Different in some cases names like "classes"
        classes = raw_checkpoint.get("classes", None)
    else:
        # Possibly a raw state dict with optional "classes" key
        state_dict = {k: v for k, v in raw_checkpoint.items() if k != "classes"}
        classes = raw_checkpoint.get("classes", None)
    else:
        # raw_checkpoint is purely the state dict
        state_dict = raw_checkpoint
        classes = None

# If no class names found in checkpoint, provide defaults
if not classes:
    classes = [f"Class_{i}" for i in range(NUM_OUTPUT_CLASSES)]

model = create_model(num_classes=len(classes))
model.load_state_dict(state_dict, strict=False)
model.class = classes
model.eval()

# =====
# 2) Nutritional Info Helpers
# =====

def query_local_database(food_name):
```

Figure 2 : preview of the Backend code

Image Recognition Model (EfficientNet-B3):

- Trained on Moroccan Food and World Open Food Facts datasets to identify specific dishes using the images.

Databases:

- MySQL Azure: Holds nutrition pertaining to Moroccan dishes.
- World Open Food Facts: Some complementing facts from international dishes are available.

LLM (Gemini API):

- Functions as a raw Data Filter by producing duly enriched but meaningfully readable responses for the users.

WORKFLOW AND SYSTEM OPERATION

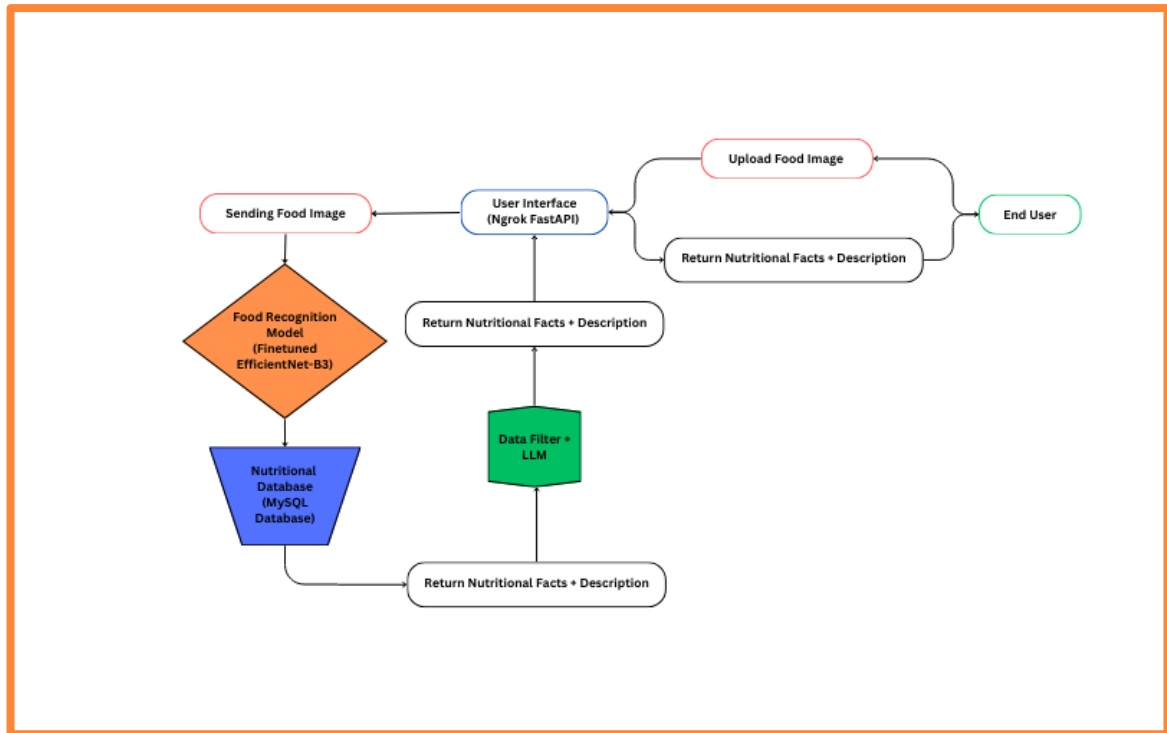


Figure 3 : system architecture

Key Features Depicted in the Diagram :

- **Food Recognition Model:** This employs EfficientNet-B3 in the recognitional features of the dishes
- **Nutritional Database:** Queries MySQL Azure and World Open Food Facts for retrieving nutritional attributes.
- **Data Filter + LLM:** Used the Gemini API to transform raw nutritional information into human-understandable well-structured text.
- **User Interface:** Developed in Ngrok and FastAPI to enable interaction between the user and backend systems.

The system operates in the following workflow :

1. Image upload by the user :

- User Image Upload through Ngrok hosted UI built with FastAPI and allowing users to upload an image of a dish.
- Optionally, users can optionally indicate the weight of the dish-in grams-for purposes of converting nutritional values accordingly.

2. Image processing :

- Once an image is uploaded, the image is sent to Flask backend for processing.
- It then processes this image through several pre-processes like resizing and normalizing it to be ready for the EfficientNet-B3 model.

3. Plate Recognition :

- This is input a processed photo to the Effectiveness Net B3 pre-trained and fine-tuned models for predicting dish type and labeling that prediction with a name and confidence score : The name of the dish
- And it returns a label with the name (of the dish) plus a confidence percentage.


4. Querying the databases :

- The controller queries the MySQL Azure DB for nutritional information for Moroccan dishes if not, he would send a query to World Open Food Facts to retrieve more details.

5. Filtering and enrichment of data (LLM Gemini API) :

- The base nutritional data (calories, proteins, fats, carbohydrates) go to the Gemini API.
- The LLM creates a structure informative text for the users such as :

" This food item provides a substantial 466 calories, making it a good source of energy. It contains 4.5 grams of protein, which is important for muscle repair and growth. With 20 grams of fats, it contributes to satiety and the absorption of fat-soluble vitamins. The 66 grams of carbohydrates serve as the primary fuel source for the body, supporting daily activities and brain function. This combination of macronutrients makes it a well-rounded option for those needing a boost of energy and essential nutrients. "



Plat identifié : **Couscous** 100.0% de confiance

Informations Nutritionnelles

Nutrition	Valeur
Calories	359.0
Proteins	13.0
Fats	2.9
Carbs	71.0

Description du Plat

{ "description": "Couscous, a staple in North African cuisine, is a delightful dish made from tiny granules of durum wheat semolina. Its light and fluffy texture makes it a versatile base for a variety of flavors, from savory stews and tagines to sweet fruit salads. Nutritionally, couscous offers a good source of carbohydrates, providing energy for the body, with approximately 71 grams per 100-gram serving. It also contains a moderate amount of protein, about 13 grams, contributing to muscle repair and growth. While relatively low in fat, around 2.9 grams, couscous is a good source of dietary fiber, aiding in digestion and promoting satiety. A 100

Figure 4 : Frontend result

6. Return results to frontend:

- The enriched results are sent to JSON form from frontend.
- These results include the name of the dish, detailed nutrition information, the analyzed image, and the explanatory text generated by Gemini API.

7. Results display:

- The frontend presents the data in a clear and visual way :
- For examplee :
 - Dish name.
 - Nutrition data adjusted according to the specified amount.
 - The image uploaded by the user.
 - The text generated and explained by API Gemini.

TOOLS AND TECHNOLOGIES

Languages and Frameworks:

- Python - Programming language utilized for backend.
- Flask - Lightweight framework for orchestration of components.
- HTML/CSS/JavaScript - Frontend technology for user interface.
- Bootstrap: For modern and responsive design.

Libraries and Tools:

- PyTorch: For loading and running the EfficientNet-B3 model.
- Pillow: Used for image processing.
- MySQL Connector: Connects and run queries against the database provided by Azure.
- Gemini API: This will convert unstructured nutrition data to structured text.
- Ngrok: For exposing the app with public link.

EFFICIENTNET-B3 TRAINING WORKFLOW

1. Datasets Preparation

Two datasets are used for the training pipeline: the Moroccan Food Dataset, concentrating on dishes that are native to Morocco, and the World Open Food Facts, which serves to complement the data with generalization of an international model for dishes.

2. Preprocessing

The first step involves resizing the images to 300 x 300 pixels. Then, normalize the pixel values using mean [0.485, 0.456, 0.406] and std [0.229, 0.224, 0.225] from the ImageNet dataset. Use other data augmentations such as random rotations, flips, and crops to make the model robust.

3. Pre-training Pre-Trained Weights

The model uses as a base EfficientNet-B3 pre-trained weights over ImageNet. Transfer Learning This consists of freezing the first part of the convolutional

layers to maintain the features learned on ImageNet, and train just the last layers of the classifier head.

4. Fine Tuning

Fine-tuning of the model is done on the Moroccan Food Dataset as a way to finely specialize the model in Moroccan dishes. Unfreeze some (or all) layers of the network with smaller learning rates (e.g., $1e-4$) for the pre-trained layers and larger learning rates (e.g., $1e-3$) for the classification layers.

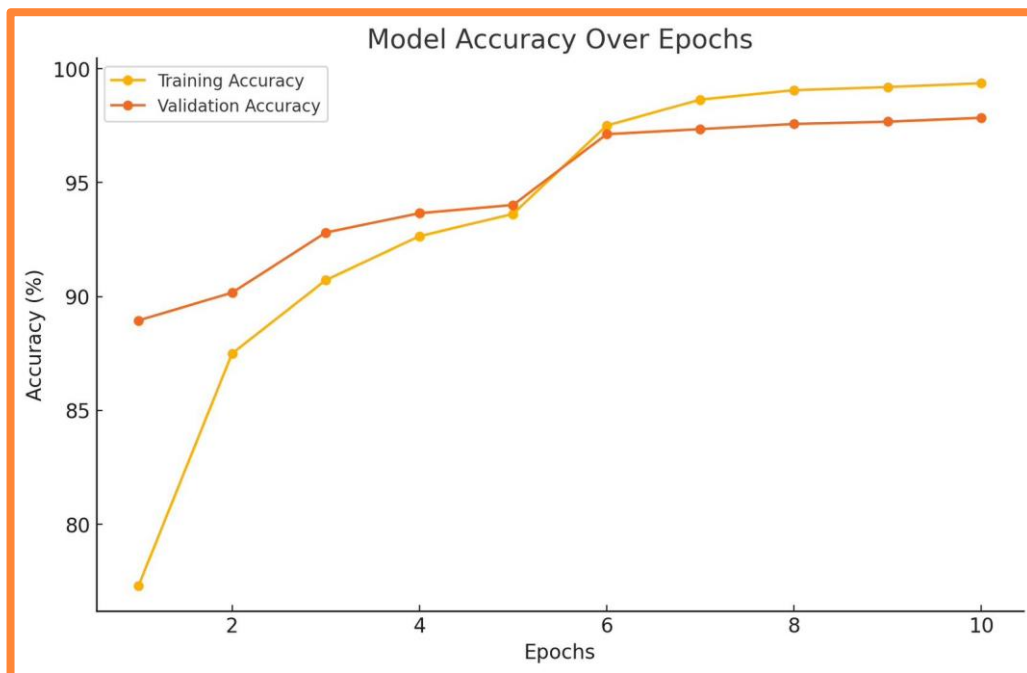
```
Epoch 1/10
-----
Train Loss: 0.7826 | Train Acc: 0.7729
Val Loss: 0.3988 | Val Acc: 0.8895
=> Nouveau meilleur modèle sauvegardé: /content/drive/MyDrive/best_model_colab.pth

Epoch 2/10
-----
Train Loss: 0.4127 | Train Acc: 0.8750
Val Loss: 0.3378 | Val Acc: 0.9017
=> Nouveau meilleur modèle sauvegardé: /content/drive/MyDrive/best_model_colab.pth

Epoch 3/10
-----
Train Loss: 0.3051 | Train Acc: 0.9072
Val Loss: 0.2420 | Val Acc: 0.9281
=> Nouveau meilleur modèle sauvegardé: /content/drive/MyDrive/best_model_colab.pth

Epoch 4/10
-----
Train Loss: 0.2474 | Train Acc: 0.9265
Val Loss: 0.2322 | Val Acc: 0.9366
=> Nouveau meilleur modèle sauvegardé: /content/drive/MyDrive/best_model_colab.pth
```

Figure 5 : Preview of the pretrained model



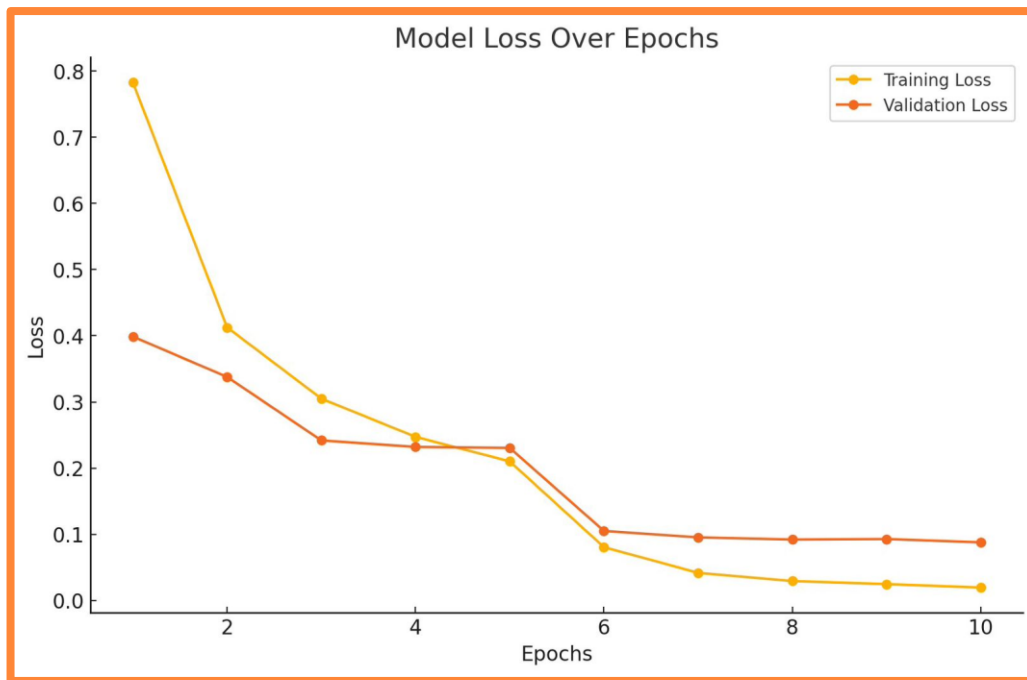


Figure 6&7 : Accuracy over epochs and loss Over epoch when training the model

5. Training parameters

Adam or SGD with momentum optimizer ($lr=1e-4$ at first). For multiclass classification, the loss function is cross-entropy loss. Train 20-50 epochs, depending on the validation accuracy and loss. Set the batch to fit 32 images (or as possible within the GPU memory). Learning rate scheduler-reduce learning rates on plateaus to enhance convergence.

KEY FEATURES

a) Accurate Recognition of Dishes :

Using EfficientNet-B3 trained on specialized datasets for reliable output..

b) Augmented Results:

Nutritional information is structured and transformed into text-based answers via Gemini API..

c) User-friendly Interface:

Simple uploading of images and clear, attractive presentation of results.

STRENGTHS AND LIMITATIONS

Strengths :

- High accuracy due to trained on specialized datasets.
- Enrichment and access of results via Gemini API.
- Intuitive interface and modern design.

Limitations :

- Data dependency : Recognition is limited to dishes included in datasets.
- Network dependence : Stable connection is needed to access database and API..

FUTURE IMPROVEMENTS

Extension of the databases :

- Add international foreign dishes to increase coverage.

Performance optimization:

- Reduce response time for image analysis.

User customization:

- Offering personalized nutrition advice based on dietary preferences and needs.

CONCLUSION

The project is the fusion of advanced technologies into an image recognition model (EfficientNet-B3) and a language model (Gemini API) to come to a very friendly interface for the nutrients analyses of the dishes. It will automatically recognize dishes and enrich them by clear text replies to the end user's needs for fast comprehension and understanding of the nutritional information while still being scalable to cover international cuisines