ÖZYEĞİN UNIVERSITY

FACULTY OF ENGINEERING

CS300

SUMMER PRACTICE REPORT

KARIM LAHYANI
Student ID: S039353

INTERNSHIP COMPANY & DEPARTMENT:
ASM - Machine Learning, Web Development

28 August 2025

Daily Work Summary

Day	Date	Work Description
1	21/07/2025	Introduction to the company and project overview. Setup development environment and Django installation.
2	22/07/2025	Started Django Essential Training course on LinkedIn Learning. Learned about Django architecture and MVT architecture.
3	23/07/2025	Continued Django training. Created first Django project and understood URL routing and views.
4	24/07/2025	Completed Django Essential Training and received LinkedIn certificate. Started building the food calorie prediction web application.
5	26/07/2025	Created Django models for FoodImage storage. Implemented basic file upload functionality and worked on the webapp design
6	28/07/2025	Searched for a relevant model in Clarifai.Then, Integrated Clarifai API for initial food recognition. Tested API responses and workflow using collected test images.
7	29/07/2025	Downloaded and explored Food101 dataset from TensorFlow Datasets. Analyzed dataset structure and class distribution.
8	30/07/2025	Started training first neural network model with the help of a kaggle script and ChatGPT using Food101 dataset. Implemented basic CNN architecture.
9	31/07/2025	Completed first model training but encountered severe overfitting issues. Training accuracy reached 90% while validation accuracy remained around 15%.
10	01/08/2025	Analyzed overfitting problem with supervisor and researched transfer learning approaches. Started implementing MobileNetV2-based model.

Student's Name: Karim Lahyani **Supervisor's Name:** Bassem Bouaziz

Student's Signature: **Supervisor's Signature:**

Daily Work Summary

Day	Date	Work Description
11	02/08/2025	Implemented MobileNetV2 transfer learning model. Fine-tuned pre-trained weights on Food101 dataset.
12	04/08/2025	Faced similar overfitting; Validation accuracy stays constant at around 45%.
13	05/08/2025	Researched advanced training techniques including data augmentation, dropout, and learning rate scheduling.
14	06/08/2025	Implemented comprehensive data augmentation pipeline with random flips, rotations, zoom, and brightness adjustments.
15	07/08/2025	Trained final model with improved techniques. Achieved better generalization with 72% validation accuracy.
16	08/08/2025	Integrated final trained model into Django application. Created the predict_calories function to integrate the model to the app.
17	09/08/2025	Added prediction history functionality and fixed some design issues.
18	11/08/2025	Tested complete application workflow. Fixed bugs and optimized model loading performance and added loading place holder for a better UX .
19	12/08/2025	Met with the supervisor to discuss the results and the importance of such experience to my career and a potential future internship to work on a specific model for the Mediet4All project
20	13/08/2025	Started working on project documentation and internship report.

Student's Name: Karim Lahyani **Supervisor's Name:** Bassem Bouaziz

Student's Signature: **Supervisor's Signature:**

Abstract

This internship was carried out at ASM in Sfax, Tunisia, focusing on developing a prototype web application for predicting food calories from images. The project aimed to serve as a starting point for the Mediet4All PRIMA vision to build a comprehensive application in the future. I began by learning Django, completing the LinkedIn Django Essential Training. Initially, the application integrated the Clarifai API to provide preliminary image recognition capabilities. Later, I tried to find a ready to use model online but I did not find one that satisfies the task properly. So, I trained three models using the Food101 dataset: a CNN from scratch (which overfitted), a MobileNetV2 model (which also overfitted), and an improved MobileNetV2 with advanced data augmentation and dropout (yielding the best results). The final solution achieved 72% validation accuracy and successfully integrated into a user-friendly web interface with a Django backend. Key skills acquired include Django web development, machine learning model training, transfer learning.

I. Introduction

The global increase in nutrition-related diseases and obesity has driven the need for nutritional assessment tools. Since manual calculation of calories intake is time-consuming and often inaccurate, the automation of the process using artificial intelligence can transform the way individuals manage their diets. During this internship, I focused on developing a food calorie prediction system that combines modern web development with artificial intelligence capability. The main objective was to create a web application that allows users to upload food images and receive automatic predictions of the type of food and the estimated relative calories.

This project was a prototype as part of the larger vision for MEDIET4ALL PRIMA, a research initiative coordinated by my supervisor, Bassem Bouaziz. The application serves as a proof-of-concept for future diet tracking and nutrition monitoring applications intended to help health-conscious users and medical professionals.

I worked with several key technologies, including Python and TensorFlow for training the AI model, Django for web development, and other libraries. The learning process involved mastering Django's architecture, understanding transfer learning, and implementing effective machine learning strategies. The project required several iterations of model development to address challenges such as overfitting and

poor accuracy, ultimately resulting in a robust solution.

II. Company Description

ASM (All Soft Multimédia) was founded in 2007 by Mr. Bassem Bouaziz and Mr. Salah Werda, who decided to create a company that would provide practical business solutions through modern technology. What started as a vision has grown into a thriving IT services company that now serves over 4,500 clients across various industries with 18 years of experience. Based in Sfax, Tunisia, ASM has expanded its reach, opening offices in Sousse and Tunis, and even crossing borders with subsidiaries in Algiers, Algeria and Dakar, Senegal, making it a leading company in the IT field.

The company specializes in making technology accessible and useful for businesses of all sizes. Their main focus areas include building custom software, creating websites and developing mobile apps. All of these products come with a modern design and well-structured UI/UX. In addition they set up point-of-sale systems that make retail operations smoother. Over the years, ASM has earned trust as a reliable partner that understands both the technical challenges and business needs of their clients.

What impressed me about ASM is their commitment to quality, they're ISO 9001:2015 certified, which proves how serious and professional they are with their work. Their flagship product, DUX-ERP, is their enterprise solution that helps businesses manage their operations more efficiently. Besides, they also create specialized software for specific industries such as aluminum carpentry and concrete businesses, plus systems that help Tunisian companies handle their CNAM and CNSS declarations(social security in Tunisia).

What I found particularly interesting during my time there was the workflow and how clients are treated in a welcoming way. They follow a simple but effective process: they start by listening to the client's needs to figure out what suits their situation the most, then they propose a solution and wait for customer review. After agreeing on all terms , they start building the software with simultaneous testing and most importantly, they remain in contact with the customers after making everything up and running which makes their after-sales service stand out. The company genuinely believes in innovation, keeping their promises, treating everyone with respect, with a strong emphasis on finding new technological solutions that align with current market trends.

ASM conducts ongoing research on emerging technologies, particularly in machine learning and artificial intelligence, with a focus on business applications. Their research aligns with current industry trends toward intelligent automation and data-

driven decision making, projects that could help people make better choices and have a closer view of the future.

My internship topic was actually part of a bigger vision they have for the MEDIET4ALL PRIMA project, which aims to promote Mediterranean diet awareness. This showed me how ASM thinks beyond just coding, they consider how their technology can benefit the society.

The Machine Learning team focuses on making AI actually useful in real applications rather than just theoretical exercises.As an intern in this department, my primary role involved developing a comprehensive web application that combined machine learning capabilities with user-friendly interface design. Thus, I needed help from the web development team. Thankfully, both departments work hand in hand to create solutions that are both smart and user-friendly. This collaboration was crucial for my project, since I needed the experience of both teams.

III. Food Calorie Predictor

3.1 Problem Statement

During my internship at ASM, my main role was developing a web application capable of predicting the calorie content of food from images. The objective was to integrate an AI model into a Django-based web application. During the journey of project realization, I faced several constraints such as the framework requirement, as I had to implement the app using the Django framework, as requested by the supervisor. Also, time limitation was a challenge to overcome since the project was restricted to 20 days, the internship duration, with many new technologies to learn. Add to that, computational resources like limited GPU availability, restricted the choice of training strategies as training time became very long, around 8 to 12 hours. In addition, no publicly available dataset combined food types with portion sizes, which made me hard code the calorie values based on moderate portion sizes for each food type, reducing prediction accuracy. And the core challenge was model performance, as I needed to balance performance and prevent overfitting to ensure reliable results in diverse food categories of the dataset.

There was no previous similar approach in the company. But on an international level, I found that MyFitnessPal and Lose It!, leaders in the field, used sophisticated computer vision systems combined with huge databases to train their models. However, these solutions are typically proprietary and not available for public access.

3.2 Tools and Techniques Used

To realize the task, I used Python as the primary language due to its extensive ecosystem for both web development and machine learning with various and easy to use libraries. To build the web application backend and host the model, I used Django because of its MTV structure and database integration. To handle the machine learning part, Tensorflow and Keras were used to train the models and TensorFlow Datasets to handle the dataset. For the visualization of the results, I used Matplotlib. In the early stages of the app development, Clarifai API was chosen to get a ready model for testing external image recognition before moving to custom models.

Model training was performed on my Laptop with an RTX 3060 NVIDIA LAPTOP GPU, using CUDA acceleration for TensorFlow. At first, I faced some problems with making the GPU ready for use. As a result, training was handled on CPU only, but after solving the problem the training was up to 10x faster.

To handle the overfitting issue I faced, I used data augmentation as it changes the images characteristics slightly to simulate real user-provided images and prevent the model from memorizing. Also, using transfer learning with MobileNetV2 made the training much more accurate and faster as the model performs well with shape recognition, that is where dropout layers come to fix freeze these layers and train the others to handle the food recognition.

3.3 Detailed Explanation

3.3.1 Web Development

I began the development process by setting up the Django structure by creating the necessary model to store food images and their data. The model will shape the structure of the output required by the ML model.

```python
class FoodImage(models.Model):
    image = models.ImageField(upload_to='food_images/')
    uploaded_at = models.DateTimeField(default=timezone.now)
    predicted_food = models.CharField(max_length=200, blank=True, null=True)
    predicted_calories = models.FloatField(blank=True, null=True)
    confidence_score = models.FloatField(blank=True, null=True)
```

After setting the structure, I developed the view part to handle the response of the AI model, shape it to the FoodImage model and then store the results in the database. While building the project, I tried to provide the best user experience by making a robust error handling structure that helps the user identify the problem when there is one, as appears in this part.

```python
def predict_calories(request):
    if request.method == 'POST':
        try:
            if 'image' in request.FILES:
                image_file = request.FILES['image']

                food_image = FoodImage(image=image_file)
                food_image.save()

                food_type, calories, confidence = predictor.predict_food(food_image.image.path)

                food_image.predicted_food = food_type
                food_image.predicted_calories = calories
                food_image.confidence_score = confidence
                food_image.save()

                return JsonResponse({
                    'success': True,
                    'food_type': food_type,
                    'calories': calories,
                    'confidence': confidence,
                    'image_url': food_image.image.url
                })
            else:
                return JsonResponse({
                    'success': False,
                    'error': 'No image provided'
                })

        except Exception as e:
            return JsonResponse({
                'success': False,
                'error': str(e)
            })

    return JsonResponse({
        'success': False,
        'error': 'Only POST method allowed'
    })
```

3.3.2 Model Training

The Food101 dataset was loaded from TensorFlow Datasets, I split into 80% training and 20% validation. Each image was resized to 224×224 pixels and normalized to [0,1]. The class names were saved to create the dictionary that maps each food type to a calorie value.

Using this dataset, I trained 3 models for 50 epochs each:

Model 1: Baseline CNN

The first model was a custom Convolutional Neural Network (CNN) implemented in Keras with the help of a kaggle script and ChatGPT. It consisted of 2 convolutional

layers, they define the model number of features and input type and shape , each one is followed by max-pooling to keep only important features, a dense layer to connect all the neurons, and a softmax output layer to convert the scores to probabilities.

```python
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, 3, activation='relu', input_shape=(224, 224, 3)),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Conv2D(64, 3, activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Conv2D(128, 3, activation='relu'),
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(len(class_names), activation='softmax')
])

model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```

Model 2: Transfer Learning with MobileNetV2

The second approach used **MobileNetV2**, pre-trained on ImageNet. The convolutional base was frozen and a custom classification head was added. This head is the part to be trained with the MobileNetV2 staying intact. In other words, the top head will learn based on the mobileNetV2 knowledge and together they form the final model. MobileNet V2 is a powerful and efficient convolutional neural network architecture developed by Google.

```python
base_model = tf.keras.applications.MobileNetV2(
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
    include_top=False,
    weights="imagenet"
)
base_model.trainable = False

model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(len(class_names), activation='softmax')
])
```

Model 3: Fine-Tuned MobileNetV2

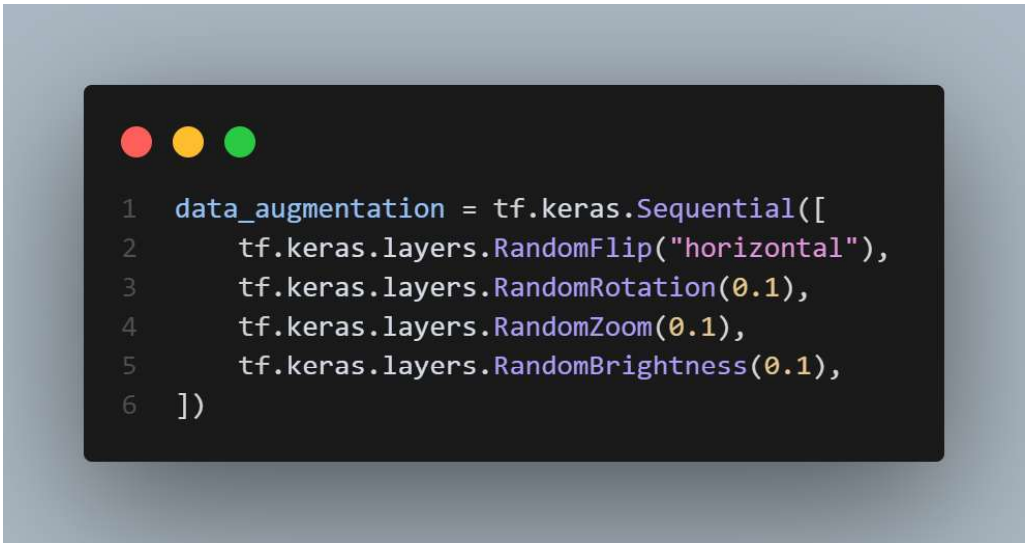
The third approach extended the previous transfer learning setup by **unfreezing part of the MobileNetV2 base**. Instead of keeping all the layers frozen, the top ones were made trainable, allowing the model to adapt more to the food dataset while still leveraging the pre-trained ImageNet features. The classification head remained the globally same as in Model 2, with dense layers followed by a softmax output for probability distribution, with a small change in the learning rate as I reduced it to guarantee the conversion of the loss function.

```python
base_model = tf.keras.applications.MobileNetV2(
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
    include_top=False,
    weights="imagenet"
)
base_model.trainable = True
for layer in base_model.layers[:100]:
    layer.trainable = False

model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(len(class_names), activation='softmax')
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(),
    metrics=[
        "accuracy",
        tf.keras.metrics.SparseTopKCategoricalAccuracy(k=5, name="top_5_acc")
    ]
)
```

In order to further improve the generalization of the model and reduce overfitting, **data augmentation** was applied during training. This technique artificially increases the size and variability of the dataset by applying transformations such as random rotations, shifts, flips, and zooming to the input images. By varying the versions of the same food images, the network learns to focus on essential features rather than memorizing specific patterns to improve its performance on unseen data.



```python
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.RandomFlip("horizontal"),
    tf.keras.layers.RandomRotation(0.1),
    tf.keras.layers.RandomZoom(0.1),
    tf.keras.layers.RandomBrightness(0.1),
])
```

3.3.3 Connecting the AI model to the webapp

Approach 1: Clarifai API Integration

In the first approach, the web application communicated directly with Clarifai's servers. When a user uploads an image, the backend sends the image after encoding it as base64 to Clarifai's food recognition model and receives the predictions in return. Then, the top prediction is mapped to its estimated calories through a local dictionary. This approach was primary step to learn how integrate external models and test the workflow of the webapp.

```python
class FoodCaloriePredictor:
    def __init__(self):
        self.pat = os.environ.get('CLARIFAI_PAT', '')
        self.user_id = os.environ.get('CLARIFAI_USER_ID', 'clarifai')
        self.app_id = os.environ.get('CLARIFAI_APP_ID', 'main')
        self.model_id = os.environ.get('CLARIFAI_MODEL_ID', 'food-item-recognition')
        self.model_version_id = os.environ.get('CLARIFAI_MODEL_VERSION_ID', '1d5fd481e0cf4826aa72ec3ff049e044')
```

```python
def predict_food(self, image_path):
        try:
            channel = ClarifaiChannel.get_grpc_channel()
            stub = service_pb2_grpc.V2Stub(channel)
            metadata = (('authorization', 'Key ' + self.pat),)
            userDataObject = resources_pb2.UserAppIDSet(user_id=self.user_id, app_id=self.app_id)
            with open(image_path, "rb") as f:
                file_bytes = f.read()
            post_model_outputs_response = stub.PostModelOutputs(
                service_pb2.PostModelOutputsRequest(
                    user_app_id=userDataObject,
                    model_id=self.model_id,
                    version_id=self.model_version_id,
                    inputs=[
                        resources_pb2.Input(
                            data=resources_pb2.Data(
                                image=resources_pb2.Image(
                                    base64=file_bytes
                                )
                            )
                        )
                    ]
                ),
                metadata=metadata
            )
            if post_model_outputs_response.status.code != status_code_pb2.SUCCESS:
                print(post_model_outputs_response.status)
                return None, None, None
            output = post_model_outputs_response.outputs[0]
            concepts = output.data.concepts
            if not concepts:
                return None, None, None
            top_concept = concepts[0]
            food_type = top_concept.name.lower()
            confidence = top_concept.value
            calories = self.food_calories.get(food_type, None)
            return food_type, calories, confidence
        except Exception as e:
            print(f"Error predicting food with Clarifai gRPC: {e}")
            return None, None, None
```

Approach 2: TensorFlow .keras Model Integration

In the second approach, I integrated the TensorFlow models trained during the internship. These models were saved locally in my laptop in .keras format and loaded directly in the Django backend. When a user uploaded an image, it is preprocessed using the same pipeline as during training, passed through the model, and the output is the index of food type later mapped from the class names file and its probability. It is then decoded into a predicted class with its corresponding calorie value.

```python
class FoodCaloriePredictor:
    def __init__(self):
        self.model = load_model("food_calorie_predictor/food101_model.keras", compile=False)
        with open("food_calorie_predictor/class_names.txt", "r") as f:
            self.class_names = [line.strip() for line in f.readlines()]
```

```python
def predict_food(self, image_path):
    try:
        img = load_img(image_path, target_size=(224, 224))
        img_array = img_to_array(img) / 255.0
        img_array = np.expand_dims(img_array, axis=0)

        predictions = self.model.predict(img_array)
        food_index = np.argmax(predictions, axis=-1)[0]
        confidence = float(np.max(predictions))
        if confidence < 0.7:
            return None, None, None

        food_type = self.class_names[food_index]
        calories = self.food_calories.get(food_type, None)

        return food_type, calories, confidence
    except Exception as e:
        print(f"Error predicting food with TensorFlow: {e}")
        return None, None, None
```

3.4 Results

The target project was completed successfully, achieving both technical and functional objectives. The iterative approach to model development showed significant performance improvements, resulting in a fully functional web application that predicts food calories from images. In the future, to further improve the app and its real world application, I recommend developing another model capable of identifying portion sizes. To ensure credibility, only results with accuracy above 70% were taken into account, as shown in the previous code snippet.

3.4.1 Model Performance Analysis

During the project, three different models were developed and evaluated during the internship. Each model addresses specific limitations identified in the previous iteration.

Model 1: Baseline CNN

The first model, a custom CNN architecture, demonstrated severe overfitting. As shown in Figure 3.1, the training accuracy rapidly reached approximately 90% within the first 10 epochs, while the validation accuracy remained low around 15% throughout the training process. This significant gap between training and validation perfor-

mance indicated that the model was memorizing training data rather than learning useful features for food recognition.

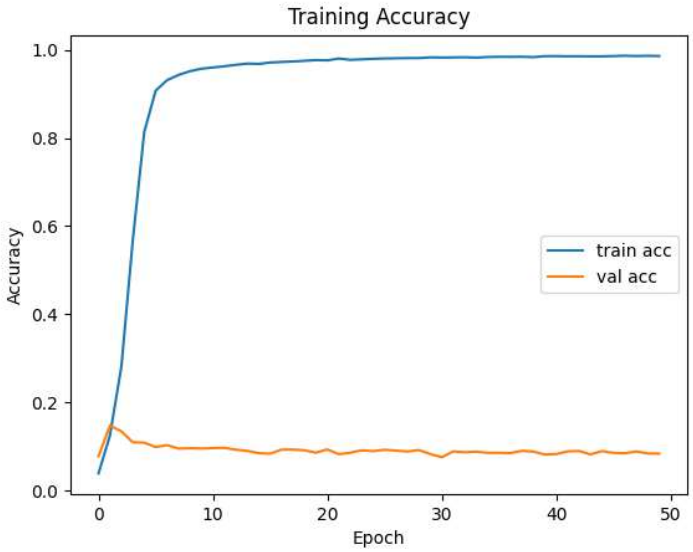


Figure 3.1: Training and validation accuracy curves for Model 1

Model 2: Transfer Learning with MobileNetV2

The second approach utilizing MobileNetV2 transfer learning showed a remarkable improvement over the first model. Figure 3.2 presents a better training behavior, with validation accuracy stabilizing around 45%. Although overfitting was still present and increased with every epoch, the gap between training and validation performance was significantly reduced compared to the first model showing that transfer learning approach successfully leveraged pre-trained ImageNet features.

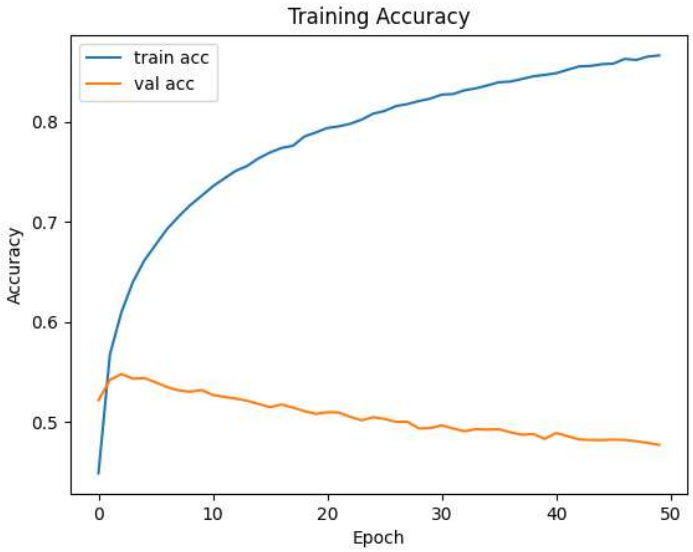


Figure 3.2: Training and validation accuracy curves for Model 2

Model 3: Fine-Tuned MobileNetV2

The final model incorporating advanced training techniques achieved the best performance. As demonstrated in Figure 3.3, both training and validation curves follow similar trajectories, indicating a successful generalization. The model achieved 84% training accuracy and 72% validation accuracy, representing a significant improvement in the model's ability to perform on unseen data. The implementation of data augmentation, dropout regularization, and fine-tuned learning rates effectively addressed the overfitting issues encountered in previous iterations.

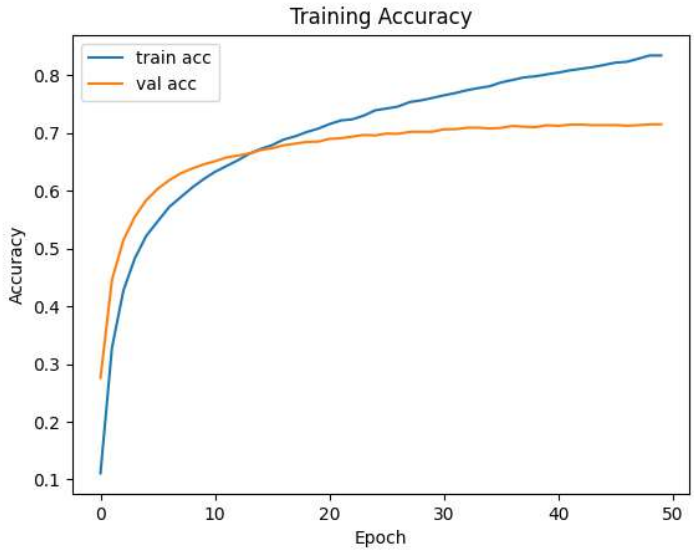


Figure 3.3: Training and validation accuracy curves for Model 3

3.4.2 Performance Comparison

Table 3.1 summarizes the performance metrics of all three models, clearly demonstrating the progressive improvement achieved through iterative development.

Model	Training Accuracy	Validation Accuracy
Baseline CNN	90%	15%
MobileNetV2 Transfer Learning	85%	45%
Enhanced MobileNetV2	84%	72%

Table 3.1: Performance comparison of all three models.

The final model's 72% validation accuracy represents a significant achievement for food recognition tasks, particularly considering the diversity of the Food101 dataset containing 101 different food categories. The final overfitting gap of only 12% indicates that the model has successfully learned to generalize beyond the training data instead of just memorizing it.

3.4.3 Web Application Implementation

The final model was successfully integrated into the Django web application. Figure 3.4 shows the complete user interface, demonstrating the practical implementation of the machine learning model in a real-world application. Also, my app presented useful UI/UX features: Drag-and-drop image upload, real time predictions, confidence scoring for credibility and reliability, prediction history and a responsive design including loading indicators.

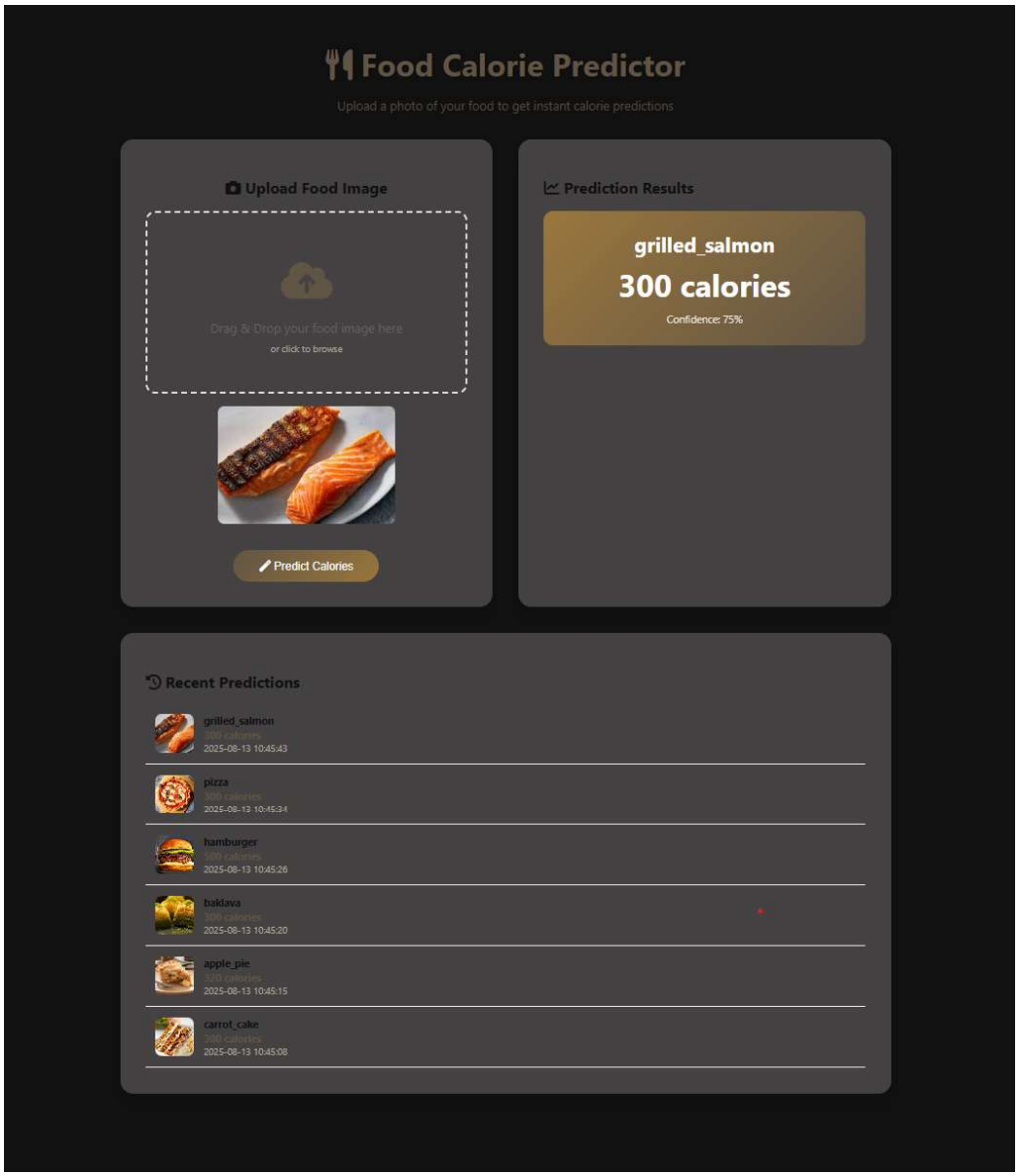


Figure 3.4: Complete web application interface.

3.4.4 Practical Performance Evaluation

I tested the app with different food images, it demonstrated the model's capability to accurately identify food items from the Food101 dataset for clear images with

good lighting. The application successfully processed images of diverse foods from various international kitchens existing in the dataset including main dishes such as grilled salmon, hamburger and pizza, also desserts including apple pie and baklava.

The calorie estimation system, while simplified to moderate portion sizes, provides users with close approximations. This approach serves the project in its proof-of-concept nature and objectives, and establishes a foundation for more sophisticated portion-size predicting model in future attempts.

IV. Conclusions

During my 2 years at Özyeğin university, I have completed 132 credits. Throughout the journey, I have learned a lot from various courses, especially as a CS student, everything seems always to be connected. Hence, this internship represented a valuable opportunity to connect my academic background with a real-world project.

In **CS201**, I have learned the skill of problem solving, this quality gives a boost to ones way of thinking which made me overcome the problems I faced. Completing **CS102**, strengthened my understanding of object-oriented programming through my projects involving management systems with Java Swing, both of which translated naturally into Django's MVT architecture: the separation of models, views, and templates. Furthermore, in **CS202** the Bank Management System and Food Ordering Website projects and coursework familiarized me with database management. The food ordering system web app was my best project at the time, it taught me user interface design and full-stack integration. Also, my previous experience with Flask in the Food Ordering Website project also made the transition to Django smoother. Learning PC components and their connection in **CS240** helped me solve some problems with RAM and GPU while training the models.

While there were similarities between class projects and internship in terms of building complete systems, the differences were significant. University projects typically have are usually related to class material and with a clearer expectations of the results. In contrast, the internship project required me to discover the world of machine learning and to search for the things that I am not familiar with, such as selecting an appropriate machine learning model and debugging GPU compatibility issues. In summary, unlike class assignments, the internship required extensive self-learning and research.

As an artificial intelligence and machine learning enthusiast, the internship enriched my academic perspective by introducing new concepts and techniques that can be applied in my coursework. I became familiar with transfer learning, data augmentation, dropout regularization, and fine-tuning pretrained models, all ad-

vanced machine learning strategies not yet covered in my curriculum. Additionally, I gained hands-on experience with full-stack development which will enhance both my future academic projects and personal research initiatives.

Beyond technical knowledge, I acquired valuable skills that will have a long-term impact on my career. I strengthened my ability to work independently under time pressure, improved my debugging and research strategies, and focused on my self-learning techniques and professional guidance to quickly adapt to new technologies and projects requirements. This internship confirmed my interest in pursuing a career in **artificial intelligence**. It also highlighted the importance of bridging technical innovation with user-centered design to provide society with useful tools that help in daily life.

Finally, my experience at ASM was very positive. The collaborative culture between the Machine Learning and Web Development teams ensured a supportive learning environment, and the company's relationship with clients inspired me to think about technology in terms of social impact. The professionalism and welcoming atmosphere of the team, especially my supervisor, Mr. Bassem Bouaziz, made the internship both enriching and motivating. Overall, this experience not only reinforced my academic career but also left me with both confidence and curiosity for future challenges.

Appendix

Environment Setup

- Python 3.10
- TensorFlow 2.15
- Django 5.0

Django Certificate

References

Kanbar, A. (n.d.). Helper Functions Essential [Jupyter notebook]. Kaggle. https://www.kaggle.com/code/alikanbar/helper-functions-essential

ASM Tunisie (n.d.). About Us. https://asm-tunisie.com/a-propos/

ASM Tunisie (n.d.). DUX-ERP. https://dux-erp.com/

Clarifai. (n.d.). Food-item recognition model. https://clarifai.com/clarifai/main/models/food-item-recognition?tab=overview

Django Software Foundation. (n.d.). Django Documentation. https://docs.djangoproject.com

GeeksforGeeks (2025). MobileNetV2 definition. https://www.geeksforgeeks.org/

Lahyani, K. (2025). Food Calories Prediction [Source code]. GitHub. https://github.com/KarimLahyani/Food-Calories-Prediction

MEDIET4ALL PRIMA. (n.d.). Mediterranean Diet for All. https://www.mediet4all.eu/

TensorFlow. (n.d.).TensorFlow Datasets: https://www.tensorflow.org/datasets