**CNN-BASED FOOD IMAGE RECOGNITION AND NUTRIENT ANALYZER FOR DIABETIC PATIENTS**

**Mentor:** prof. Anandi Giridharan

**Author :** Ashwini Kannan K

**Institution :** Bengaluru

**Project :** BTech Internship Project

**Date :** July 2025

**Abstract**

This project introduces a compact deep learning system that identifies food products from pictures and automatically provides nutrition-related details beneficial for dietary monitoring and diabetes education. A transfer-learning framework based on MobileNetV2 is developed using the Food-101 dataset (101 types of food), and a nutritional lookup layer after prediction connects identified foods to recorded attributes like Glycemic Load (GL), Glycemic Index (GI), Calories, Carbohydrates, Fat, Protein, and a diabetes advisory label (Low / Moderate / High). Model development utilized a two-phase approach: (1) train the newly added classification layers while keeping the base CNN frozen; (2) unfreeze and fine-tune specific deeper layers using a lower learning rate. An example inference shows effective categorization of an input image, along with related nutritional data obtained from an external datasheet.   
The resulting pipeline can act as a basis for mobile health applications that facilitate meal tracking, diabetes education, and automated nutritional feedback. Future efforts might expand to estimating portion sizes, handling multi-label meals, categorizing by age, and creating region-specific nutrition tables.

**1. Introduction**

Supervision of food intake is becoming crucial for controlling conditions like diabetes and obesity, which can be life-threatening. Nevertheless, manual logging is tedious and susceptible to mistakes. Progress in computer vision and convolutional neural networks (CNNs) now allows for the identification of everyday foods directly from images taken with mobile devices. When connected to curated nutritional information, such systems can promptly offer users practical dietary recommendations.

This project implements an end-to-end pipeline that:

1. Accepts an image of food captured by the user
2. Then a transfer-learned CNN (MobileNetV2 backbone) classifies food category
3. Nutrition is looked up from the processed datasheet.
4. Displays per-serving(100g) metrics and diabetes-oriented advice.

The goal is not just recognition accuracy but *practical usability*—a fast, lightweight model suitable for day-to-day use.

**2. Problem Statement & Objectives**

**Problem:** Automatically identify the food item in an image and provide quick nutrition guidance that can assist diet tracking and diabetic meal planning.

**Objectives:**

* Use transfer learning to build an image classifier for 101 food categories(based on food101 dataset.
* To support user-friendly inference .
* The GL , macronutrients and diabetes flag are to be denoted.
* It demonstrates end to end prediction with the data provided by the user

**3. Dataset Description**

**3.1 Food-101 Dataset**

* **Classes:** 101 food categories (e.g., pizza, apple pie, ramen, sushi).
* **Images per class:** 800 (food images with variable background, lighting, presentation).
* **Split :** 80:20
* **Variability:** Realistic noise— different plating styles, toppings variation—makes it a good benchmark for robustness

**3.2 Nutrition Mapping Sheet**

A separate spreadsheet (e.g., GL\_food101.xlsx) provides per-food nutrition entries aligned (as closely as possible) to Food-101 class labels. Each row includes:

* Predicted Food (label string aligned to model class name)
* Diabetes Advice (Low / Moderate / High)
* Glycemic Load (numeric)
* Glycemic Index (numeric)
* Carbohydrates (g)
* Fat (g)
* Protein (g)
* Calories (kcal)

**3.3 Actual Data Used in This Project**

The nutrition data used here is got from reliable sources like USDA ,University of Sydney, Foodstruct and Harvard GI Table. The GL value is calculated and the final datasheet is got from nutrients.ipynb.

|  |  |  |
| --- | --- | --- |
| **Split** | **No.of.Images** | **Notes** |
| **Training** | 80,800 | 80% of the dataset. Used ImageDataGenerator with augmentation during training |
| **Validation** | 20,200 | 20% of the dataset. Used to monitor model performance and avoid overfitting |

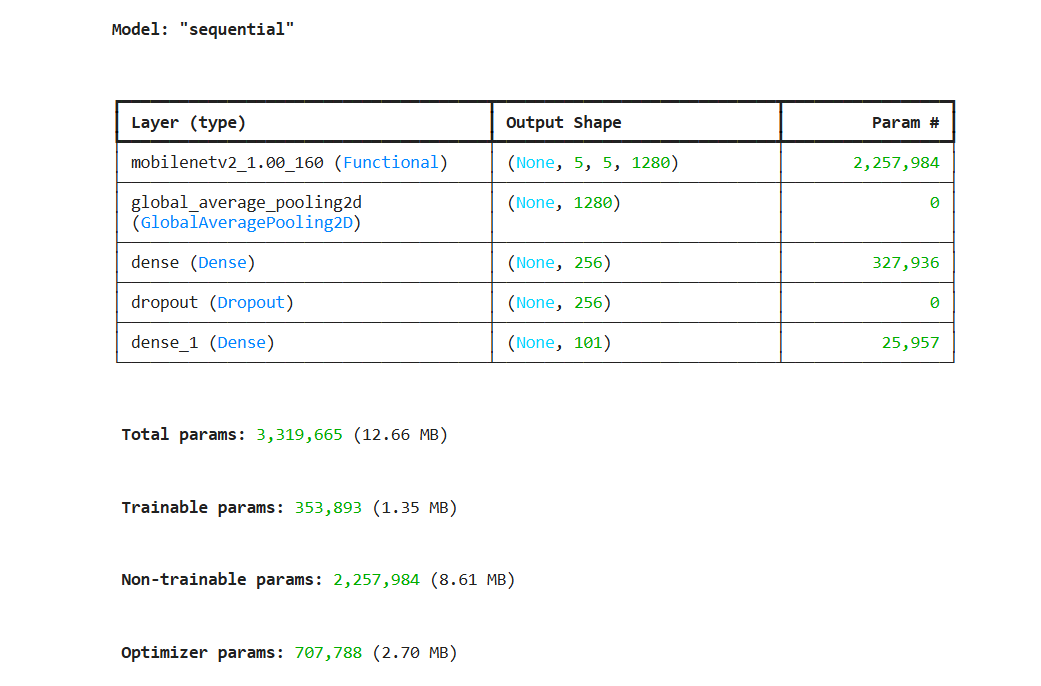
**4. System Overview**The five stages of the pipeline are:  
1. **Data Loading & Preprocessing** – Images loaded from class-labelled directories; resized (e.g., 160×160) and scaled to [0,1] or pre-processed per MobileNetV2 requirements.  
2. **Data Augmentation** – It is used by ImageDataGenerator for tasks like flips, rotations , zoom.  
3. **A Transfer Learning Model** – MobileNetV2 backbone (a model pretrained on ImageNet) + Global Average Pooling + Dense + Dropout + Output layer.  
4**. A Two-Phase Training** – (i) Top dense layers are trained with base frozen; (ii) Deeper layers are unfreezed and fine-tuned.  
5. **The Prediction + Nutrition Lookup** – Input image from user is taken for inference and then it's mapped to the predicted label in nutrition and provides the advisory message.

**5. Methodology**

**5.1 Model Architecture**

**MobileNetV2** is used as the feature extractor. In Keras this variant appears as mobilenetv2\_1.00\_160 . The base network outputs a 5×5×1280 feature map (for its configured input size), followed by:

* GlobalAveragePooling2D
* Dense(256, activation='relu')
* Dropout(0.3)
* Dense(101, activation= softmax )



**Figure 1:** Model Summary.

**5.2 Input Size & Preprocessing**

* tensorflow.keras.preprocessing.imagegenerator is used to load input image.
* The images are resized to (160,160) with 3 colour channels to ensure compatibility with the model.
* For the normalization the pixel values scaled by dividing by 255.0 which helps with performance.

**5.3 Freezing & Fine-Tuning Strategy**

**Phase 1 (Frozen Training):**

* Pre trained ImageNet weights are used to initialize base model (trainable=False).
* Dense(512), Dropout(0.3), and final Dense(101, softmax layer) are the only ones trained.
* This is trained for 100 epochs.

**Phase 2 (Fine-Tuning):**

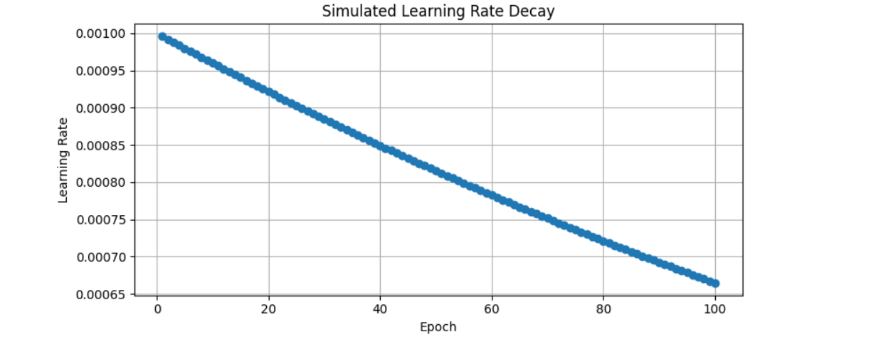
* After the initial training the other layers are unfrozen (trainable=True). This is to prevent overfitting.
* To update pre trained weights the lower learning rate used.
* This is trained with 30 epochs.

**5.4 Optimizer & Learning Rate**

* Optimizer: [ Adam]
* Initial LR: [0.001]
* During fine-tuning LR reduced ( 1e-4 ).
* Learning Rate behaviour is observed between various checkpoints and visualized in LR decay graph.



**Figure 2:** Learning Rate extracted from saved models.



**Figure 3:** Simulated learning rate decay curve.

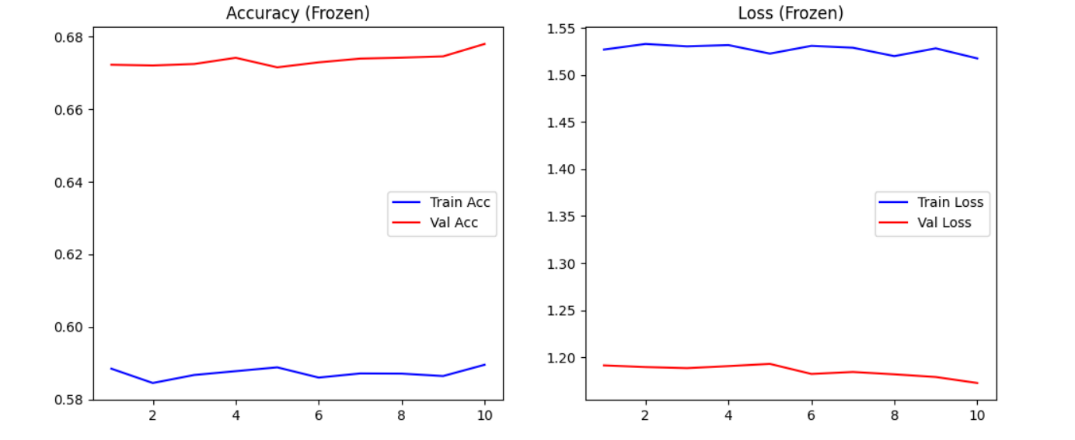
**5.5 Regularization & Overfitting Control**

* To generalize the images data augmentation is used.
* Dropout layer in classifier head to reduce the memorization of patterns.
* To stabilize the updates lower LR is used during fine-tuning.
* ReduceLROnPlateau and EarlyStopping are used to used.

**6. Model Training and Evaluation**

**6.1 Frozen Base Results**

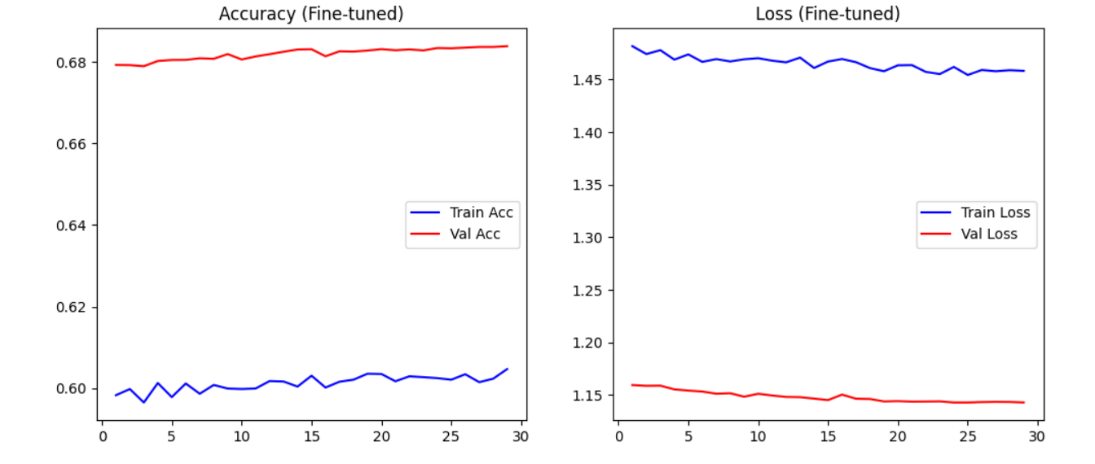
In the initial frozen training stage 10 epochs are taken in account and here the validation accuracy is stabilised around 67% while the training accuracy was around 59%(100 epochs are used)The validation loss is lower than training loss due to data augmentation and batch normalization.

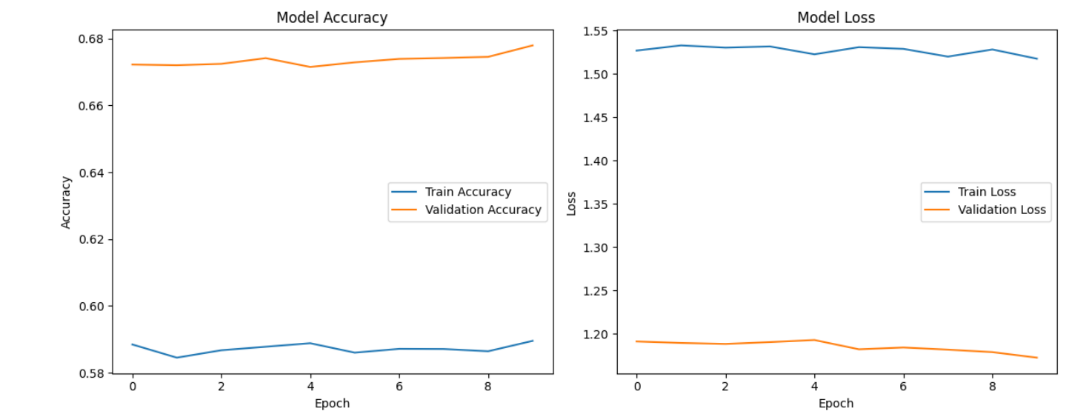


**Figure 4:** Accuracy vs Loss (Frozen Stage).

**6.2 Fine-Tuning Results**

After unfreezing and fine-tuning the validation accuracy showed a slight increase(68%). Compared to frozen training the training accuracy rose a bit which points out that the noise of the dataset limited overfitting. The validation loss trend was downward relative to frozen training.





**Figure 5:** Accuracy vs Loss (Fine-Tuned Stage), Accuracy vs Epoch

**6.3 Final Evaluation Metrics**

|  |  |  |
| --- | --- | --- |
| Metric | Train | Validation |
| Accuracy | 0.6047 | 0.6835 |
| Loss | 1.4473 | 1.1430 |

The per class accuracy and confusion matrix are printed out in the form of XLSX file.

**7. Inference Workflow & Example**

**7.1 Prediction Code**

****

**Figure 6:** Screenshot of notebook prediction cell output.

**7.2 Nutrition Lookup**

After predicting the class label, the system searches the nutrition spreadsheet (GL\_food101.xlsx) for a match. When found, the per-serving(100g)nutrition values are extracted and rendered in a table.

**Example Output (**Waffle):



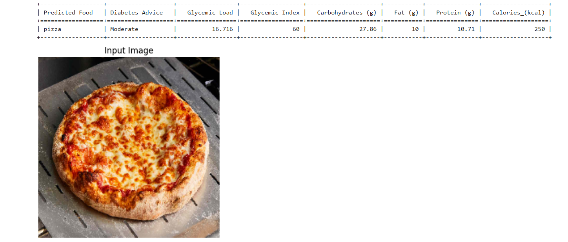
**Figure 7:** Screenshot of nutrition table + display.

**7.3 Visual Result**

Few examples of the user input image and the output.



**Figure 8**: Original Input image .



**Figure 9**: Input image + prediction display

**7.3 Challenges**

* Data Collection
* Duration of the training
* Categorization of similar looking dishes



**Figure 10**: True vs prediction

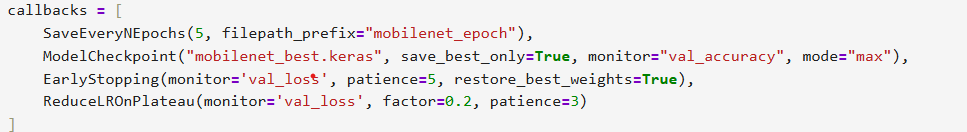
**8. Implementation Details**

**8.1 Environment**

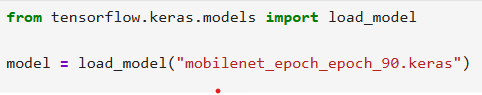
* Framework: TensorFlow / Keras (version?)
* Python
* Hardware[AMD Ryzen 5 3450U with 4 cores(CPU) ,Radeon Vega Mobile Graphics(GPU)]
* Training time: [ 52 hrs for frozen + 30 hrs fine-tune]

**8.2 Saving & Reloading Models**

When run for a long time to save progress of the model checkpoints are used.



**Figure 11**: The checkpoint for every 5 epoch (code snippet)

****

**Figure 12**: To reload the model (code snippet)

**9. Discussion**

**9.1 Observations**

* The validation accuracy is higher compared to training accuracy and the trend continues from frozen training to the fine-tuning.
* Fine-tuning produces finer results compared to frozen training.
* Some classes may be misleading (e.g. .cakes and pies)

**9.2 Limitations**

* A single dish can be classified and the combination of the dishes isn’t analyzed.
* The output is given for standard (100g) serving and it doesn’t analyze the portion of the input image.
* Inconsistencies in the nutritional data (As the average value is taken for the dishes; e.g. cupcakes, chicken curry).

**9.3 Future Work**

* Portion estimation based on the input image.
* Multi label classification of the meal.
* More inclusive data collection from various regions.
* On device deployment .

**10. Conclusion**

The model-built for CNN-based Food Image Recognition and Nutrient Analyzer for Diabetic Patients is a transfer learning model where it undergoes two-phase training(frozen, unfrozen + fine-tuned) ,which later uses the data from food101 and classifies the user image and gives an output of nutritional data and advisory message for the diabetic patients.

**References**

1. Sun, J., Radecka, K., & Zilic, Z. (2019). *FoodTracker: A Real-time Food Detection Mobile Application by Deep Convolutional Neural Networks*. arXiv.<https://doi.org/10.48550/arXiv.1909.05994>
2. Lu, Y., Stathopoulou, T., Vasiloglou, M. F., Christodoulidis, S., Stanga, Z., & Mougiakakou, S. (2020). An artificial intelligence-based system to assess nutrient intake for hospitalised patients. *IEEE Transactions on Industrial Informatics*, *16*(11), 7474–7483.<https://doi.org/10.1109/TII.2020.2987444>
3. Tahir, G. A., & Loo, C. K. (2021). *A comprehensive survey of image‑based food recognition and volume estimation methods for dietary assessment*. Healthcare, 9(12), 1676. https://doi.org/10.3390/healthcare9121676
4. Sahoo, D., Hao, W., Ke, S., Wu, X., Le, H., Achananuparp, P., Lim, E.-P., & Hoi, S. C. H. (2019). *FoodAI: Food Image Recognition via Deep Learning for Smart Food Logging*. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. Retrieved from<https://ink.library.smu.edu.sg/context/sis_research/article/5430/viewcontent/4._FOODAI_FOOD_IMAGE_RECOGNITION_VIA_DEEP_LEARNING_FOR_SMART_FOOD_LOGGING__KDD2019_.pdf>
5. Bossard, L., Guillaumin, M., & Van Gool, L. *Food-101 – Mining Discriminative Components with Random Forests.* In ECCV, 2014.
6. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. *MobileNetV2: Inverted Residuals and Linear Bottlenecks.* CVPR, 2018.
7. TensorFlow/Keras Documentation. Transfer learning & fine-tuning guides.
8. Optional: Add dietary references for GL & GI definitions (ADA, Harvard Health, etc.) if you discuss medical guidance formally.

**Appendix – Glycaemic Load & Diabetes Advisory Logic**

Assigns diabetic advice:

* **Low GL (<10)**: Safe in moderation for individuals with diabetics.
* **Moderate GL (10–19)**: Portion control recommended.
* **High GL (≥20)**: Limit frequency.