**Food Classification Using Deep Learning**

**Project Overview**

This project focuses on developing a deep learning-based food classification model capable of categorizing images into 34 different food categories. The solution involves dataset preprocessing, training three models (Custom CNN, VGG16, and ResNet), model evaluation, and deploying the application via a Flask web interface.

**1. Dataset Information**

* **Total Classes:** 34
* **Dataset Source:** Food Image Classification Dataset
* **Data Preprocessing:**
  + Balanced the dataset by selecting 200 images per class.
  + Uploaded selected images to Google Drive for training.

**2. Model Development**

Three different models were trained and evaluated:

**A. Custom CNN Model**

* **Architecture:**
  + Flatten layer
  + Fully connected layers with 512 neurons
  + Dropout layer (0.5)
  + Softmax activation for classification
* **Training Parameters:**
  + **Epochs:** 30
  + **Loss Function:** Categorical Crossentropy
  + **Optimizer:** Adam (learning rate = 0.0001)
* **Outputs:**
  + Trained model saved as custom\_model.h5
  + Validation Report stored in Custom\_Model.txt

**B. VGG16 (Transfer Learning)**

* **Base Model:** VGG16 (pre-trained on ImageNet, feature extraction enabled)
* **Training Parameters:**
  + **Epochs:** 30
  + **Optimizer:** Adam
* **Outputs:**
  + Trained model saved as vgg16\_model.h5
  + Validation Report stored in VGG16\_Model.txt

**C. ResNet50 (Transfer Learning)**

* **Base Model:** ResNet50 (pre-trained on ImageNet, feature extraction enabled)
* **Training Parameters:**
  + **Epochs:** 30
  + **Optimizer:** Adam
* **Outputs:**
  + Trained model saved as resnet\_model.h5
  + Validation Report stored in ResNet\_Model.txt

**3. Evaluation Metrics**

Each model was evaluated using the following metrics:

* **Accuracy**
* **True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN)**
* **Precision, Recall, F1-Score**

**4. Exception Handling & OOP Implementation**

* **Object-Oriented Programming:** Implemented a FoodClassifier class for model training and evaluation.
* **Exception Handling:** Used try-except blocks to handle errors gracefully during model training.

**5. Nutritional Data Storage**

* Extracted nutritional data for all 34 classes.
* Stored information in nutrition\_data.json (Protein, Fiber, Calories, Carbohydrates, Fat per class).

**6. Flask Web Application (Deployment)**

* **Frontend:** HTML interface using Bootstrap.
* **Backend:** Flask API to:
  + Accept image input.
  + Load the selected model (Custom CNN, VGG16, or ResNet50).
  + Display the predicted food class and validation metrics.

**7. Final Deliverables**

* **Trained Models:** Custom CNN, VGG16, ResNet.
* **Validation Reports:** Custom\_Model.txt, VGG16\_Model.txt, ResNet\_Model.txt.
* **JSON File:** nutrition\_data.json
* **Flask Web Application:** Hosted locally for predictions.
* **GitHub Repository:** Includes source code, README, and documentation.
* **Documentation & PPT:** Describes project methodology and results.