Food Image Classification System

This presentation provides a detailed overview of a comprehensive Food Image Classification System. This system integrates image classification techniques with nutritional information retrieval to provide users with valuable insights about the food they consume.

The system leverages various technologies, including deep learning models, web scraping, and APIs, to achieve its functionality. The following slides will delve into the key components and functionalities of the system, offering a clear understanding of its architecture and capabilities.

by Harsh Raj Gupta

Setup and Dependencies

Key Libraries

- requests: HTTP requests to APIs
- beautifulsoup4: Web scraping
- serpapi: Fetching image URLs from Google Images
- tensorflow and keras: Building and training the deep learning model
- cv2 (OpenCV): Image processing
- PIL (Pillow): Image manipulation
- streamlit: Creating a web-based user interface

The system relies on a set of Python libraries to perform its functions. These libraries facilitate tasks such as making API calls, web scraping, image processing, and building the user interface. The deep learning model is built using TensorFlow and Keras, while OpenCV and Pillow are used for image manipulation.

Image Data Collection

1 SerpAPI

Fetches image URLs from Google Images for a list of food items, downloads, and saves the images into a structured directory.

2 DuckDuckGo Search

Uses the duckduckgo_search library to fetch additional images, randomly selects food categories, and downloads images to a test directory.

The system employs two primary methods for collecting food images: SerpAPI and DuckDuckGo Search. These methods ensure a diverse dataset for training and testing the model. Images are organized into subfolders based on food item, and named sequentially for easy management.

Data Cleaning

Delete Invalid Images

Checks for corrupt or unsupported image files using OpenCV and PIL.

Delete Corrupt Images

Removes files that fail the checks to ensure the dataset contains only valid images.

Check Images

Scans the dataset for invalid files and removes them.

Data cleaning is a crucial step in maintaining the quality of the training data. The system includes functions to identify and remove corrupt or unsupported image files. By using both OpenCV and PIL, the system ensures that only valid images are used for training, improving the model's accuracy and reliability.

Dataset Preparation

1. **Load Images**

   Loads images from the directory and splits them into training and test sets.

2. **Resize Images**

   Resizes images to a consistent size (128x128 pixels).

3. **Normalize Pixel Values**

   Normalizes pixel values to the range [0, 1].

4. **Convert Labels**

   Converts labels to categorical format for multi-class classification.

The dataset preparation process involves several key steps to ensure the data is ready for training. Images are loaded from the directory, resized to a consistent size, and normalized to improve training efficiency. Labels are converted to a categorical format to support multi-class classification, and the dataset is cached and shuffled to further enhance training performance.

Model Architecture/Pipeline

1. **Convolutional Layers**

   Three convolutional layers with ReLU activation.

2. **Max-Pooling Layers**

   Followed by max-pooling layers.

3. **Batch Normalization**

   Applied after each convolutional layer to stabilize training.

4. **Dense Layers**

   Flattened output passed through two dense layers with dropout for regularization.

5. **Softmax Activation**

   Final layer uses softmax activation to output probabilities for 50 food classes.

The deep learning model is built using a Convolutional Neural Network (CNN) architecture. This architecture is well-suited for image classification tasks, consisting of convolutional layers, max-pooling layers, batch normalization, dense layers, and softmax activation. The model is designed to output probabilities for 50 food classes, providing accurate and reliable predictions.

Model Training

Optimizer

Adam with a learning rate of 0.001.

Loss Function

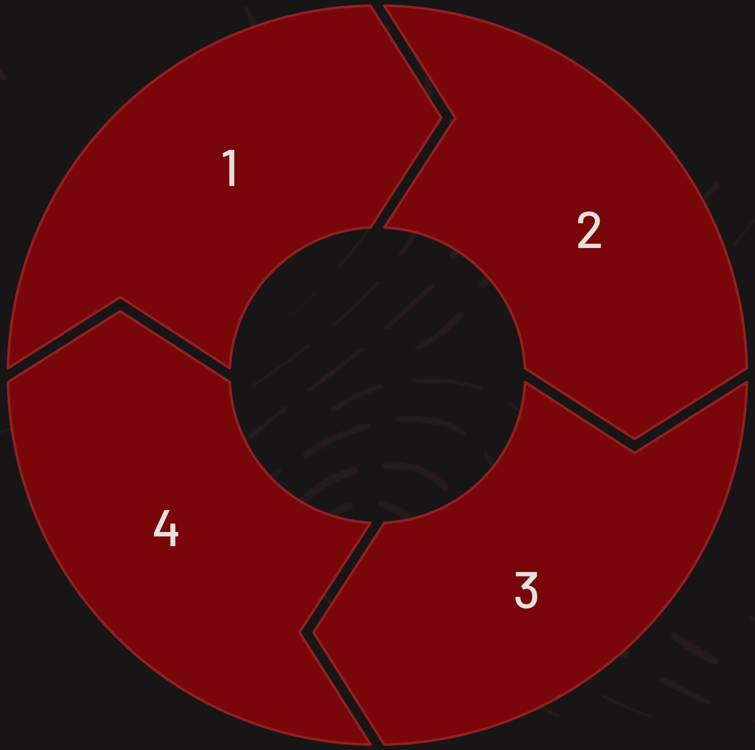
Categorical cross-entropy.

Metrics

Accuracy.

Training

10 epochs with data augmentation.



The model training process involves several key parameters and techniques to optimize performance. The Adam optimizer is used with a learning rate of 0.001, and the categorical cross-entropy loss function is employed for multi-class classification. Data augmentation is applied to improve generalization, and the training process is monitored using validation data. The trained model is saved for future use.

Model Inference

1

Preprocess Image

Resizes and normalizes the input image.

2

Predict Image

Uses the model to predict the food class.

3

Return Results

Returns the predicted class name and confidence score.

The model inference process involves preprocessing the input image, predicting the food class using the trained model, and returning the predicted class name along with a confidence score. This allows the system to accurately classify food images and provide users with reliable results. The preprocessing step ensures that the input image matches the model's input requirements.

Nutritional Information Retrieval

USDA API

Integrates with the USDA FoodData Central API.

Fetch Info

Sends a query to the USDA API with the predicted food name.

Extract Details

Extracts and returns key nutritional details.

The system integrates with the USDA FoodData Central API to fetch nutritional information for the predicted food item. This adds a valuable feature to the system, allowing users to get both the food classification and its nutritional content. The system sends a query to the USDA API with the predicted food name and extracts key nutritional details such as calories, protein, fats, and carbs.

Streamlit Web Application

Upload Image

Users can upload an image of a food item.

Prediction

The model predicts the food class and displays the result along with confidence.

Nutritional Information

The app fetches and displays nutritional data for the predicted food item.

The system uses Streamlit to create a user-friendly web interface. Users can upload an image of a food item, and the model predicts the food class and displays the result along with confidence. The app also fetches and displays nutritional data for the predicted food item, making the system accessible to non-technical users.

Potential Improvements

- **Data Augmentation**:
  - Increase the diversity of the training data using more advanced augmentation techniques.
- **Model Optimization**:
  - Experiment with different architectures (e.g., transfer learning with pre-trained models like ResNet or EfficientNet).
- **Error Handling**:
  - Add more robust error handling for API requests and image processing.
- **User Feedback**:
  - Allow users to provide feedback on predictions to improve the model over time.