Building a Food Recognition System Using Deep Learning

In this project, I set out to develop a **food recognition** system using convolutional neural networks (CNNs). Leveraging the power of deep learning, the system aims to accurately identify food items from images, which has significant applications in fields like health monitoring, food delivery, and dietary management.

Project Overview

The core objective of this project is to recognize different food categories from images. For this, I used the popular **Food-101 dataset** and pre-trained models, such as **InceptionV3**, to create a high-performing image classification system.

Key Features:

- **Data Preprocessing**: Leveraged image augmentation to increase the diversity of the training data.
- **Model Architecture**: Fine-tuned the pre-trained **InceptionV3** model.
- **Optimization**: Used Stochastic Gradient Descent (SGD) for optimizing model performance.
- **Real-Time Prediction**: The trained model can classify food items from new images with high accuracy.

Let's dive into the technical details!

1. Data Source: The Food-101 Dataset

The dataset used in this project is the **Food-101** dataset, which contains 101 food categories. However, for this project, we reduced the dataset to focus on 3 specific food classes to streamline the process and test the model's performance on a smaller scale.

Dataset Structure:

- **Train set**: 2250 images (750 per class)
- **Test set**: 750 images (250 per class)

The dataset contains images of varying sizes, so preprocessing is essential to ensure that the images are of consistent size for training.

2. Data Preprocessing

To ensure our model can generalize well, we performed **data augmentation** and **normalization**. Augmentation techniques like **zooming**, **shearing**, and **horizontal flipping** were applied to the training data, while the test data was normalized to rescale pixel values between 0 and 1.

The `ImageDataGenerator` from **Keras** to implement this.

```
# Data Augmentation for training set
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# Normalization for test set
test_datagen = ImageDataGenerator(rescale=1./255)
```

Data Flow

Generated batches of images using `flow_from_directory` to streamline loading and training on large datasets.

```
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(299, 299),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(299, 299),
    batch_size=batch_size,
    class_mode='categorical')
```

3. Model Architecture

For this project, I used the **InceptionV3** model pre-trained on **ImageNet**. The reason for choosing InceptionV3 is its depth and capacity to capture intricate features in the data. Rather than training a model from scratch, I fine-tuned the InceptionV3 model by adding custom layers for our food classification task.

Key Layers:

- **Global Average Pooling**: This replaces the fully connected layers to reduce overfitting.
- **Dense Layer**: A fully connected layer with ReLU activation.
- **Dropout Layer**: Used to prevent overfitting by randomly setting a fraction of input units to 0 during training.
- **Softmax Layer**: Used as the final classification layer, with 3 outputs representing the food classes.

```
inception = InceptionV3(weights='imagenet', include_top=False)
x = inception.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.2)(x)
predictions = Dense(3, activation='softmax')(x)
model = Model(inputs=inception.input, outputs=predictions)
```

4. Training the Model

Optimizer and Loss Function

I used the **Stochastic Gradient Descent (SGD)** optimizer with a learning rate of `0.0001` and a momentum of `0.9`. The loss function chosen is **categorical cross-entropy**, which is well-suited for multi-class classification problems.

```
model.compile(optimizer=SGD(learning_rate=0.0001, momentum=0.9),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Training Process

I trained the model for **15 epochs**, and used **callbacks** like `ModelCheckpoint` to save the best-performing model and `CSVLogger` to record the training history for future analysis.

```
history = model.fit_generator(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=30,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size,
    callbacks=[csv_logger, checkpointer])
```

5. Results and Performance

After training for 30 epochs, the model achieved excellent accuracy in classifying the food images. Below are some key results:

- **Training accuracy**: ~95%
- **Validation accuracy**: ~93%
- **Validation loss**: Gradually decreased with training

The performance could further be improved by training on the entire Food-101 dataset and applying advanced fine-tuning techniques.

6. Conclusion

This project demonstrates how we can leverage deep learning, particularly convolutional neural networks, to build an efficient food recognition system. Although the system was trained on a smaller dataset, it performed remarkably well in classifying different food categories.

Potential Future Work:

- **Expand to more food categories**: The current model handles 3 classes, but it can be expanded to 101 classes using the full dataset.
- **Real-time implementation**: The model can be deployed in real-time applications like health monitoring apps or food delivery systems.