Convolutional Neural Network-Powered Food Recognition: Smarter Sorting for Fruits and Vegetables

By Azizkhuja Asomiddinov

Overview

This project implements a complete pipeline for image classification using convolutional neural networks. It leverages both a custom model trained from scratch and several transfer learning approaches with popular pretrained architectures (MobileNetV2, ResNet50, and EfficientNetB0). This project also includes a final verification phase that downloads sample images from Bing to evaluate the best-performing model. The original pretrained models and datasets are credited to their respective sources.

Dataset

The Food and Vegetables dataset, curated by Sunny Agarwal, is a collection of high-quality images depicting various fruits and vegetables. It is designed to aid in the development and evaluation of image classification models, particularly in agricultural technology and dietary applications.

Link: SunnyAgarwal4274/Food_and_Vegetables · Datasets at Hugging Face

Dataset Details

- **Content:** The dataset comprises images of diverse fruits and vegetables, organized into 36 distinct classes. Each class corresponds to a specific type of fruit or vegetable, facilitating effective training of machine learning models
- **Structure:** Images are organized in subfolders, each named after the corresponding class (e.g., "Apples," "Carrots"). This structure simplifies data management and access.
- **Size:** The dataset contains 3114 training images and 351 validation images, providing a substantial amount of data for training and testing purposes.

Potential

The goal of this project is to develop an accurate and efficient image classification model capable of distinguishing between different types of food items, including fruits and vegetables. By leveraging deep learning and pretrained models, this project aims to automate the identification of food categories, which can have significant real-world applications in food banks, grocery stores, and storage facilities. In food banks, such a system can help streamline inventory management by automatically classifying and sorting donated items, reducing manual effort and improving distribution efficiency. Similarly, in food storage and warehouse

environments, accurate food classification can assist in proper categorization for storage conditions. Additionally, grocery stores and self-checkout systems can benefit from such a model by enabling automated product recognition, reducing checkout time, and minimizing errors. Ultimately, this project seeks to enhance efficiency and improve logistics in food-related industries through the power of machine learning.

Data preprocessing

Data preprocessing is a crucial step in preparing datasets for training convolutional neural networks (CNNs). CNNs are highly sensitive to the structure and quality of input data, and improper preprocessing can lead to suboptimal model performance, inefficient learning, or even complete failure of the training process. Preprocessing ensures that images are properly formatted, normalized, and structured for a CNN model. Addressing dimension mismatches, scaling issues, and data inconsistencies improves the stability, accuracy, and efficiency of the training process. A well-preprocessed dataset allows CNNs to focus on learning patterns rather than handling inconsistencies, leading to better overall performance in real-world applications.

Approach

In this project, the approach to model development was structured as an iterative improvement process. The goal was to start with a simple custom convolutional neural network (CNN) and gradually refine it by increasing the number of epochs, layers, and filters to enhance accuracy and learning capability. This stepwise approach allowed for controlled experimentation, ensuring that each change contributed to model performance without unnecessary complexity.

Initial Two-Epoch Model (Baseline)

The first model was designed with minimal complexity to serve as a baseline for comparison. It contained:

- One Convolutional Layer with 4 filters of size (3,3) and ReLU activation.
- One MaxPooling Layer with a (2,2) kernel size to reduce spatial dimensions.
- Flattening Layer to convert feature maps into a one-dimensional vector.
- Output Softmax Layer with 36 neurons to classify images into different categories.

Key Observations:

- This model, trained for only 2 epochs, provided a quick overview of the dataset and model behavior.
- With only one convolutional layer and a small number of filters, feature extraction was very limited.

- The model suffered from low accuracy and underfitting, as it lacked sufficient depth to extract meaningful features from images.

Five-Epoch Model – Increasing Depth and Capacity

To improve performance, the next iteration introduced:

- Two Convolutional Layers with 8 and 16 filters, respectively, to extract more detailed features.
- Two MaxPooling Layers to progressively reduce spatial dimensions and retain important features.
- Training Extended to 5 Epochs, allowing the model to learn more from the dataset.

Improvements and Benefits:

- Adding a second convolutional layer allowed the model to detect more complex patterns and textures, improving feature extraction.
- Increasing the number of filters from 4 to 8 → 16 enabled the model to capture finer details.
- More training epochs from 2 to 5 improved generalization but still left room for optimization.
- While performance improved, the model still lacked deep feature hierarchies, which are crucial for handling complex image classification tasks.

The final custom model iteration included:

- Three Convolutional Layers with 32, 64, and 128 filters, increasing feature extraction power.
- Three MaxPooling Layers to systematically reduce feature map size while preserving key spatial information.
- A Fully Connected Dense Layer (128 neurons) before the output layer, introducing non-linear transformations that allow for better classification.
- Training Extended to 10 Epochs, further refining learned features.

Key Performance Enhancements:

- Higher filter counts (32 → 64 → 128) significantly improved the ability to detect edges, textures, and shapes.
- More convolutional layers helped the model learn hierarchical patterns, improving accuracy in distinguishing different food types.

- Adding a Dense Layer (128 neurons) before classification helped increase model capacity, allowing it to form more complex decision boundaries.
- Longer training (10 epochs) allowed the network to fine-tune its parameters and generalize better, reducing both bias and variance.

Conclusion: Progressive Model Refinement

This progressive approach to model development provided valuable insights:

1. Starting Simple – The two-epoch model acted as a lightweight baseline, helping identify dataset characteristics.
2. Gradual Expansion – Incrementally increasing depth and filters allowed for controlled performance gains without over-complicating the model.
3. Final Optimized Model – The ten-epoch model provided the best balance of complexity and performance, leveraging multiple convolutional layers, higher filters, and additional dense layers to extract rich feature representations.

By following this structured improvement process, the final model achieved significantly better classification performance, making it well-suited for real-world applications such as food recognition in food banks, grocery stores, and inventory management systems. Increasing the number of epochs and layers further would not provide a significant performance boost but could lead to overfitting, higher computational costs, and diminishing returns. Instead, refining training strategies, optimizing existing layers, and incorporating transfer learning are more practical ways to enhance the model's effectiveness.

Implementing Pretrained Models: MobileNetV2, ResNet50, and EfficientNetB0

After training a custom CNN model from scratch, the next step in the project was to leverage pretrained models such as MobileNetV2, ResNet50, and EfficientNetB0. These models have been trained on large-scale datasets like ImageNet and have already learned powerful feature representations, making them highly effective for transfer learning.

For each pretrained model, two different architectures were implemented:

1. Using GlobalAveragePooling2D for feature extraction
2. Using Flatten with Dropout for feature extraction

1. Differences Between MobileNetV2, ResNet50, and EfficientNetB0

Each pretrained model has unique architectural characteristics that impact performance:

MobileNetV2

- Designed for efficiency and optimized for mobile applications.
- Uses depthwise separable convolutions, reducing computational cost while maintaining good accuracy.
- Best for lightweight models, but performance can be slightly lower compared to larger architectures.

ResNet50

- A deep model with 50 layers, featuring residual connections that help train deep networks without vanishing gradient issues.
- Well-suited for complex image classification tasks because it learns hierarchical features at multiple levels.
- Best generalization and accuracy when combined with GlobalAveragePooling2D.

EfficientNetB0

- Uses compound scaling, balancing width, depth, and resolution for optimal efficiency.
- Outperforms older architectures (like ResNet50) in some cases but is computationally lighter than deeper ResNet variants.
- Great balance between accuracy and efficiency, though ResNet50 performed better in this project.

Comparing GlobalAveragePooling2D vs. Flatten with Dropout

For feature extraction, two different approaches were tested:

GlobalAveragePooling2D

- Takes the average of each feature map across spatial dimensions.
- Reduces dimensionality significantly, minimizing overfitting by keeping only the most important information.
- Works best with pretrained models as it retains the most meaningful features while reducing unnecessary complexity.

Flatten with Dropout

- Flatten converts the feature map into a 1D vector, keeping all spatial features.
- A Dense(256) layer is added to refine features before the output layer.
- Dropout (rate = 0.5) randomly deactivates neurons, which helps reduce overfitting.
- Introduces more trainable parameters, requiring more data and computation to generalize well.

Why Models with Dropout Showed Poorer Performance

Models using **Flatten with Dropout** consistently performed worse than those using **GlobalAveragePooling2D** due to:

Overfitting and Generalization Issues

- Flattening retains too much spatial detail, making the model prone to overfitting.
- The Dense(256) layer adds more trainable parameters, which requires a large dataset to generalize well.
- Dropout removes neurons randomly, leading to loss of useful feature information from pretrained models.

Unnecessary Additional Parameters

- Pretrained models already have well-optimized feature extractors, and adding a fully connected Dense(256) layer introduces millions of additional parameters.
- This makes training less efficient, slowing down convergence and increasing computational cost.

Better Regularization with GlobalAveragePooling2D

- GlobalAveragePooling2D acts as a built-in regularization technique, reducing overfitting without needing Dropout.
- Averaging feature maps ensures the most important information is retained, improving robustness.

Why ResNet50 with GlobalAveragePooling2D Showed the Best Performance

Among all pretrained models, **ResNet50 with GlobalAveragePooling2D** performed the best due to:

Residual Connections Improve Learning

- ResNet50 uses skip (residual) connections, which prevent the vanishing gradient problem, making training deeper networks more effective.
- This allows ResNet50 to capture hierarchical features more efficiently than MobileNetV2 and EfficientNetB0.

Balanced Depth and Feature Extraction

- With 50 layers, ResNet50 is deep enough to learn detailed patterns without being excessively computationally expensive.
- The combination of depth and residual learning makes it highly effective for complex image classification tasks.

GlobalAveragePooling2D Enhances Generalization

- Since ResNet50 already extracts high-quality feature maps, GlobalAveragePooling2D enhances generalization by averaging important features while reducing overfitting.
- Unlike Flatten with Dropout, it does not introduce unnecessary trainable parameters, ensuring better accuracy with fewer computations.

Conclusion

By testing multiple pretrained architectures, it became clear that:

- ResNet50 with GlobalAveragePooling2D provided the best balance of accuracy, efficiency, and generalization.
- Flatten with Dropout models struggled due to unnecessary parameters and information loss.
- Transfer learning using GlobalAveragePooling2D is the best approach for pretrained models, as it optimally extracts high-level features while preventing overfitting.

Final Decision: ResNet50 with GlobalAveragePooling2D

Based on comparative experiments and performance evaluations, ResNet50 with GlobalAveragePooling2D was selected as the final model. It achieved the best balance of accuracy, generalization, and computational efficiency, making it the most suitable choice for the food classification task.

This final model is now ready for real-world applications, including automated food recognition in food banks, grocery stores, and inventory management systems, ensuring accurate and efficient classification of food items.

Weaknessess

Although ResNet50 with GlobalAveragePooling2D was chosen as the final model for its strong feature extraction and high classification accuracy, it has notable limitations. The model is designed to classify images containing a single type of fruit or vegetable, making it unsuitable for images featuring multiple food items. Since it is a single-label classifier, it assumes each image belongs to only one category, which does not reflect real-world scenarios where different foods often appear together. Additionally, the model cannot detect object count or position, meaning it does not provide information on the quantity of food items within an image. Its accuracy may also be affected by complex backgrounds, poor lighting conditions, and occlusions, which can obscure key features necessary for classification.

Future Improvements

Future improvements for this project could focus on enhancing the model's versatility and real-world applicability by addressing its current limitations. Expanding the dataset to include more diverse food arrangements, different lighting conditions, and complex backgrounds would help improve generalization and robustness. Introducing subclasses for fruits and vegetables—such as distinguishing between ripe and unripe bananas or different apple varieties—could refine classification accuracy further. Additionally, transitioning from a single-label classifier to multi-label classification would allow the model to process images containing multiple food items simultaneously. Implementing object detection models like YOLO or Faster R-CNN could enable the model not only to classify food items but also to detect and localize them within an image, making it more practical for food banks, grocery stores, and automated checkout systems. Another potential improvement would be incorporating quantity estimation algorithms to count the number of fruits or vegetables in an image, which could streamline inventory management. Furthermore, integrating quality detection features could help assess the freshness or ripeness of produce, making the model useful in food safety and waste reduction efforts. By implementing these advancements, the system could evolve into a more comprehensive and intelligent food classification tool, improving efficiency in food distribution and management.