Machine Learning Approach for Feature Extraction from Images

Objective:

The primary objective of this project is to automatically extract critical product details, such as weight, volume, voltage, wattage, and dimensions, from images. This capability is highly valuable for digital marketplaces, where product descriptions are often incomplete or unavailable. Accurate feature extraction directly from images enhances product metadata, supporting better decision-making in e-commerce and content moderation.

Approach:

Data Preparation

Dataset: The project utilizes a dataset containing images with corresponding labels for training purposes. These labels annotate product features (e.g., weight, volume, etc.). A separate test set is reserved for evaluating model performance, where images are provided without labels to simulate real-world conditions.

Image Downloading: A script is employed to download images from provided URLs using the download_images function. In the case of download failures, placeholder images are automatically used to ensure continuity in the dataset.

Preprocessing: Each image is resized to 224x224 pixels and normalized to standardize inputs to the model. Preprocessing is essential to ensure that all images are in a consistent format suitable for both training and inference stages.

Machine Learning Models

Feature Extraction Model

Convolutional Neural Network (CNN): A CNN model serves as the backbone for feature extraction from images. The architecture comprises:

Convolutional layers: for automatic feature detection.

Activation functions: such as ReLU to introduce non-linearity.

Pooling layers: to reduce spatial dimensions and computational complexity.

Fully connected layers: to synthesize features into meaningful classifications or regressions.

This model is responsible for learning and extracting relevant visual features, which are then used to predict key product characteristics.

Training Process

Data Loaders: Custom data loaders are employed to handle image loading and preprocessing. The get_data_loaders function ensures proper batching and shuffling of both the training and validation datasets, thus maintaining data integrity and enabling efficient training.

Model Training: The CNN model is trained using the labeled images from the dataset. Key elements of the training process include:

A **loss function**, such as CrossEntropyLoss for classification tasks.

An **optimizer**, such as Adam, for updating model weights based on the loss gradients.

Multiple **epochs** of training, where the model iteratively minimizes the loss on the training set while monitoring performance on the validation set to prevent overfitting.

Experiments

Model Evaluation

Accuracy Metrics: The performance of the model is evaluated using several metrics, including:

Accuracy: to measure the proportion of correct predictions.

Loss: to quantify the model's error during training and validation.

Precision, Recall, and F1 score: for a more comprehensive assessment of the model's ability to classify product features accurately, especially in cases of imbalanced data.

Predictions

Test Dataset: The trained model is applied to the test dataset to generate predictions for each image. The predict.py script automates this process, where each image is passed through the model to output predictions in a structured format (e.g., "2 grams" for weight).

Conclusion

The project demonstrates the successful application of convolutional neural networks (CNNs) to extract product features directly from images. Key conclusions drawn from the experiments include:

Model Performance: The CNN model exhibited strong performance in feature extraction, achieving high accuracy and low loss on both training and validation datasets.

Feature Extraction: The model reliably identified and classified product attributes, such as weight and dimensions, based solely on image data.

Challenges: Some challenges included handling diverse image qualities and ensuring consistent preprocessing across the dataset. These were addressed by applying robust data handling techniques and preprocessing pipelines.

Future Work: Potential future improvements include:

Enhancing model robustness to handle a wider variety of image conditions (e.g., low resolution or varying angles).

Incorporating advanced architectures such as transfer learning to further improve accuracy and efficiency.