Code Documentation: Image and Text Processing with Batch Processing and Progress Tracking

# Introduction

This document provides an overview of the code designed to train and test a machine learning model using image and text data. The code employs batch processing and progress tracking for handling large datasets efficiently.   
Key components include:  
 - Feature extraction using ResNet50 (a deep learning model)  
 - Text extraction using Tesseract OCR  
 - Prediction of entity values using XGBoost  
Progress tracking is implemented to ensure transparency and user feedback during processing of the data.

# Dataset

The datasets used in this code include:  
 - Training Data: 20,000 entries (image URLs, entity values)  
 - Test Data: 10,000 entries (image URLs, entity values)  
Both datasets are processed in batches to ensure efficient use of memory and computational resources.

# Model Description

The code uses a combination of image feature extraction and text extraction to make predictions for specific entity values. The model pipeline includes:  
 1. \*\*ResNet50\*\*: A pre-trained deep learning model is used to extract image features. The images are resized to 224x224 pixels to be compatible with the model.  
 2. \*\*Tesseract OCR\*\*: Text is extracted from the images, which contains relevant information (such as values and units) for the entity.  
 3. \*\*XGBoost\*\*: The extracted image features and text length are fed into the XGBoost model for prediction. The model is trained on the training dataset and then used to predict entity values for the test dataset.

## Image Preprocessing and Feature Extraction

Images are downloaded and resized to 224x224 pixels to meet the input size requirements of ResNet50. The pre-trained ResNet50 model is used to extract image features, which are essential for entity value prediction.

## Text Extraction and Processing

Text is extracted from images using the Tesseract OCR engine. This text often contains important information related to the entity's value and units (e.g., weight, height). A regex-based approach is used to extract numeric values and units, which are then combined to form a valid prediction.

# Batch Processing

To handle large datasets efficiently, the code processes data in batches. Batch sizes are:  
 - Training Data: 2,000 entries per batch  
 - Test Data: 1,000 entries per batch  
Processing data in smaller batches prevents memory overload and ensures faster execution. Each batch is processed independently, and progress is tracked using a progress bar.

# Progress Tracking

The `tqdm` library is used to display progress bars for both the training and test data processing. For every batch processed, the progress bar updates, displaying the percentage of completed batches.  
This feature provides transparency and real-time feedback to the user on the status of the data processing and model training.

# Model Training

The extracted image features and processed text are combined and used as input to train an XGBoost model. The model is trained to predict entity values (e.g., weight, dimensions) based on these combined features.  
The training data is split into training and validation sets, and label encoding is used for categorical values. After training, the model is used for predictions on the test dataset.

# Test Data Prediction

The trained XGBoost model is applied to the test dataset. Predictions are made in batches of 1,000 entries, and each prediction consists of a combined value and unit (e.g., "20 kg").  
The predictions are written to a CSV file (`test\_out.csv`) in batches, ensuring that partial results are saved incrementally in case of interruptions.

# Conclusion

This code is designed to handle large datasets using batch processing, feature extraction, and progress tracking. It leverages the power of deep learning for image processing, OCR for text extraction, and XGBoost for entity value prediction. The addition of progress tracking provides valuable real-time feedback to the user.