Capstone Project

Project: Machine Learning-Based Counterfeit Drug Detection Using Image Recognition and OCR

Data Preparation Future Engineering

Group Members's Name

1.Efrata Abebe

2.Hermela Hailegiogris

3.Mihret Tamene

April 28,2025

Data Preparation/ Feature Engineering

1. Overview

In machine learning projects, data preparation and feature engineering form the foundation for model performance. In particularly, for image-based tasks like this fake drug detection, the quality and organization of the input data directly influences the model's success. High quality, well preprocessed images of pharmaceutical packaging ensure the object detection model can accurately differentiate between genuine and counterfeit medicines.

Data preparation and feature engineering are crucial as they directly impact model's ability to learn meaningful patterns and make accurate predictions. Data preparation ensures that the dataset is clean, consistent, and suitable for training. High-quality and well-engineered features enable the model to identify patterns indicative of fake drugs, ultimately enhancing detection reliability and minimize false positives. This phase lays the foundation for successful machine learning outcomes in the project.

Data Preparation: involves cleaning and organizing the collected data to ensure they are suitable for analysis. Key steps include:

- **Data Cleaning**: Removing duplicates, correcting errors, filtering out poor-quality images, and handling missing values to improve dataset quality.
- **Data Splitting**: Dividing the dataset into training, validation, and test sets to properly evaluate model performance effectively.
- **Normalization**: Scaling numerical features to a standard range, which helps improve model performance and convergence.

Feature Engineering: focuses on creating meaningful features that enhance the model's ability to learn from the data. Key activities include:

- **Feature Extraction**: Identifying and extracting relevant characteristics from images and spectroscopic data, such as color histograms or spectral peaks.
- **Dimensionality Reduction**: Using techniques like PCA to reduce the number of features while retaining essential information, which can improve model efficiency.
- **Creating New Features**: Combining existing features or applying transformations to generate new, informative features that better capture the underlying patterns.

2. Data Collection

After evaluating multiple publicly available datasets, we have selected the ***"Counterfeit_Med_Detection"*** dataset created by Harshini T G R from roboflow Universe for the image-based counterfeit detection component of our project. This dataset contains approximately 1,550 images of pharmaceutical packaging,

captured under varied real-world conditions such as different lighting, angles, and backgrounds. Each image is annotated with bounding boxes labeling the packages as either Authentic or Counterfeit, making the dataset highly suitable for object detection model training with YOLOv8. The images include a mix of full medicine boxes, tablet packs, and blister sheets, providing diverse visual examples.

This dataset was downloaded in YOLOv8-compatible format (images and labels), ensuring a straightforward integration into the model training pipeline. Its size, annotation quality, and variability make it an excellent foundation for building a robust counterfeit drug detection system.

In addition to image data, Optical Character Recognition (OCR) will be utilized to extract printed information such as drug names, batch numbers, expiry dates and manufacturer information from packaging images. To validate the OCR output is cross-referenced against public databases including ***"FDA Drug Label Data"*** and ***"FDA Full-Text Drug Product Labeling Database"***. These official datasets provide verified drug names, active ingredients, manufacturers, and label formats, enabling the detection of inconsistencies such as spelling errors, missing fields, and invalid formats. Such anomalies in the packaging text are strong indicators of counterfeit pharmaceutical products.

Given the increasing use of barcodes and QR codes on pharmaceutical packaging for traceability, the project also integrates barcode verification components. Future extensions of the project aim to verify chemical composition information extracted from packaging or drug labels. Supporting resources include:

3. Data Cleaning

After downloading the "Counterfeit Med Detection" dataset created by Harshini T.G.R. from Roboflow Universe, an initial review of the dataset structure and quality was conducted. The dataset was already organized into separate train, validation, and test folders, with corresponding images and YOLOv8-format labels.

A preliminary quality inspection was performed, checking for missing, corrupted, or improperly formatted images. No major corruptions or missing annotations were detected. Since the dataset was exported in YOLOv8-compatible format, it included consistent image dimensions and corresponding bounding box annotations. A manual verification of several sample label files confirmed that only two class IDs were present (Authentic and Counterfeit), ensuring label consistency across the dataset. As a result, no additional cleaning or resizing operations were required, and the dataset was ready for immediate use in training the counterfeit drug detection model.

4. Exploratory Data Analysis (EDA)

A preliminary exploratory data analysis was conducted to better understand the distribution and characteristics of the counterfeit drug detection dataset.

The dataset was pre-split into three subsets:

- **Training set**: 4,072 images
- **Validation set**: 122 images
- **Test set**: 65 images

A visualization of this split is shown in Figure 1 below.

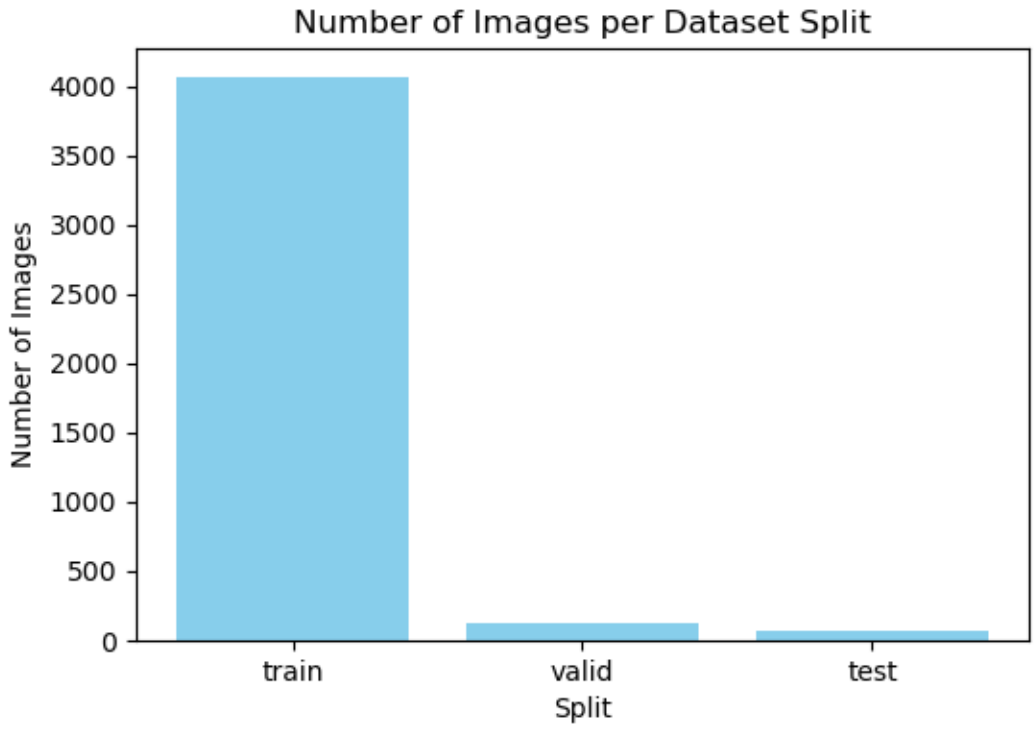

Figure 1. Number of images in the training, validation, and test sets

Upon exploratory data analysis, it was observed that the training set contained approximately 4,072 images, while the validation and test sets contained 122 and 65 images respectively. The discrepancy between the initially expected dataset size (~1,550 images) and the observed number of training samples is due to automatic data augmentation applied during dataset export from Roboflow. This augmentation introduced variations such as rotation, brightness adjustments, and flipping, effectively enlarging the dataset and

enhancing model robustness. As a result, the dataset used for training provides a more diverse and comprehensive representation of pharmaceutical packaging examples.

In addition to analyzing the dataset split, a class distribution analysis was conducted to examine the balance between the two target categories: Authentic and Counterfeit. As shown in Figure 2, the dataset contains approximately 6,089 Authentic instances and 648 Counterfeit instances across all splits.

The class distribution is highly imbalanced (Authentic vs Counterfeit ratio is about 10:1), with Authentic instances significantly outnumbering Counterfeit instances. Such class imbalance can potentially bias the model towards predicting Authentic cases more frequently. Therefore, during model training, techniques such as data augmentation for minority class, loss function weighting, or resampling strategies may be considered to address this imbalance and improve the model's ability to accurately detect counterfeit products.

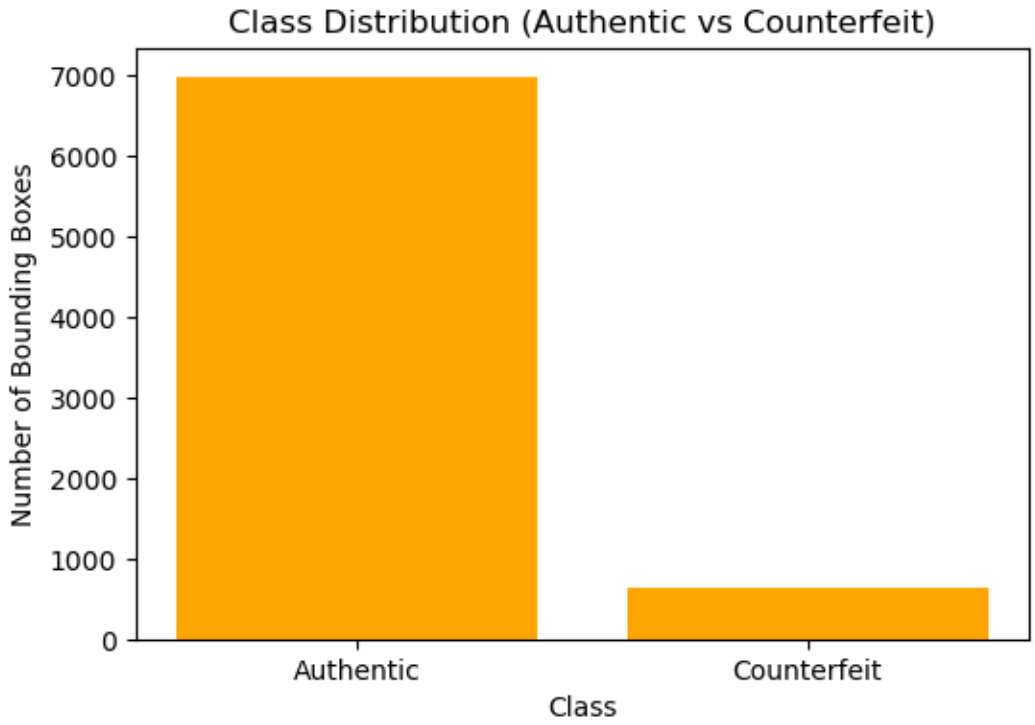

Figure 2. Class Distribution (Authentic vs Counterfeit Instances)

To further understand the visual diversity within the dataset, a random selection of training images was displayed, as shown in Figure 3. The sample images demonstrate a variety of pharmaceutical packaging types, including full medicine boxes, tablet blister packs, and strip packaging, captured under different

lighting conditions, angles, and backgrounds. This visual inspection confirmed the dataset's richness and variability, which is beneficial for training a robust object detection model capable of handling real-world scenarios.

Figure 3. Random sample images from the training set

5. Feature Engineering

Feature engineering involves preparing and transforming input images and annotations to maximize model learning and generalization. For object detection models like YOLOv8, feature engineering focuses on ensuring data consistency, improving dataset variability through augmentation, and refining annotation quality. Several steps were applied to engineer better features for training:

- **Bounding Box Validation**: All annotation files were verified to ensure bounding boxes accurately captured the pharmaceutical packages. Only two class labels Authentic (0) and Counterfeit (1) were preserved to maintain label consistency.
- **Image Resizing**: Images were standardized to 640x640 pixels to match YOLOv8 input requirements, ensuring a uniform aspect ratio and optimal performance during model training.
- **Label Cleaning**: Any redundant or misaligned annotations were removed, and missing label files were corrected where necessary.

To further enhance the model's ability to generalize across different real-world conditions, a data augmentation pipeline will be applied dynamically during training. Augmentations include:

- **Random Horizontal Flips** (to simulate mirror-like camera captures)
- **Random Brightness/Contrast Adjustments** (to simulate lighting variations)
- **Random Rotation** (up to ±10 degrees, simulating tilted mobile captures)

- **Random Scaling and Translation** (slight zooms and shifts)

These augmentations increase the effective size and diversity of the dataset without needing to manually duplicate images.

The feature engineering techniques applied were designed to make the model robust against different camera angles and positions, varying lighting conditions in real-world environments and variations in packaging appearance (authentic vs counterfeit printing quality). By enhancing variability in training samples while preserving label quality, the model is better equipped to detect counterfeit pharmaceutical packaging under diverse conditions.

6. Data Transformation

All input images were resized to 640x640 pixels to match the input dimension requirements of YOLOv8. This ensures that the model receives consistently sized images, improving convergence speed and detection performance.

Normalization is crucial for stabilizing the training process and preventing numerical instability during optimization.During training, pixel values of the input images are automatically normalized from the original range [0, 255] to a scale between [0, 1].

The bounding box annotations were structured in the YOLO format, where: x_center, y_center, width, and height are all normalized relative to image dimensions (values between 0 and 1) and Class IDs were assigned as 0: Authentic and 1: Counterfeit. This normalized label structure ensures that the detection model accurately interprets object locations during training.

Data augmentation transformations including random horizontal flips, rotations, brightness/contrast adjustments, and scaling will be applied dynamically during model training. These augmentations improve model generalization by exposing it to diverse visual variations of pharmaceutical packaging.

Model Exploration

1. Model Selection

The choice of Random Forest as the machine learning model for fake drug detection is driven by its ability to handle mixed data types, robustness against overfitting, and overall performance. While it has certain limitations regarding interpretability and computational efficiency, its strengths align well with the project's objectives, making it a suitable candidate for effectively detecting counterfeit drugs.

2. Model Training

The model was trained using a yolo with the following hyperparameters: a learning rate of 0.001, a batch size of 32, and a total of 50 epochs. Data augmentation techniques included random rotations, shifts, and flips to enhance diversity. For validation, Stratified K-Fold Cross-Validation with 5 splits was employed to ensure balanced representation of classes. The Adam optimizer and sparse categorical cross-entropy loss function were used to compile the model.

3. Model Evaluation

The primary goal of the evaluation phase is to assess the effectiveness of the object detection model (YOLOv8) in accurately distinguishing between authentic and counterfeit pharmaceutical packages. Evaluation will be based on both quantitative performance metrics and visual inspection of predictions.
The following metrics will be used to evaluate model performance on the validation and test sets:

- Precision: Measures the proportion of correctly predicted counterfeit samples among all samples predicted as counterfeit.
- Recall: Measures the proportion of correctly predicted counterfeit samples among all actual counterfeit samples.

- F1-Score: Combines precision and recall into a single metric to balance false positives and false negatives.
- mAP (mean Average Precision):
    - mAP@0.5: Measures accuracy at a single Intersection over Union (IoU) threshold of 0.5.
    - mAP@[0.5:0.95]: A more comprehensive metric that averages precision across IoU thresholds from 0.5 to 0.95.
- Confusion Matrix: Will be used to summarize classification outcomes and visualize misclassifications between the Authentic and Counterfeit classes.

In addition to numerical evaluation, visual inspection of model predictions will be conducted:

- Bounding Box Visualizations: Predicted boxes with class labels will be overlaid on validation/test images to visually assess detection accuracy.
- Misclassification Review: False positives and false negatives will be analyzed to understand common error patterns, such as packaging similarities or poor lighting conditions.

These metrics are selected due to their robustness in handling imbalanced datasets and their relevance to object detection tasks.

4. Code Implementation

```
import os
import random
import matplotlib.pyplot as plt
import cv2

# Set your dataset base path
base_path = './Counterfeit_med_detection.v4i.yolov8'
splits = ['train', 'valid', 'test']
# Initialize counters
count_dict = {'train': 0, 'valid': 0, 'test': 0}
class_count = {'Authentic': 0, 'Counterfeit': 0}
# Define class names based on your dataset
class_names = {0: 'Authentic', 1: 'Counterfeit'}
# Count images and labels
for split in splits:
    image_folder = os.path.join(base_path, split, 'images')
    label_folder = os.path.join(base_path, split, 'labels')

    images = os.listdir(image_folder)
    labels = os.listdir(label_folder)

    count_dict[split] = len(images)
```

```
    for label_file in labels:
        with open(os.path.join(label_folder, label_file), 'r') as f:
            lines = f.readlines()
            for line in lines:
                class_id = int(line.split()[0])
                class_count[class_names[class_id]] += 1

# Print Image Split Statistics
print("\n📦 Number of Images in Each Split:")
for split, count in count_dict.items():
    print(f" - {split.capitalize()}: {count} images")

# Print Class Distribution
print("\n🏷️ Class Distribution Across All Splits (bounding boxes):")
for class_name, count in class_count.items():
    print(f" - {class_name}: {count} instances")

# Plot 1: Number of Images per Split
plt.figure(figsize=(6, 4))
plt.bar(count_dict.keys(), count_dict.values(), color='skyblue')
plt.title('Number of Images per Dataset Split')
plt.ylabel('Number of Images')
plt.xlabel('Dataset Split')
plt.show()

# Plot 2: Class Distribution
plt.figure(figsize=(6, 4))
plt.bar(class_count.keys(), class_count.values(), color='orange')
plt.title('Class Distribution (Authentic vs Counterfeit)')
plt.ylabel('Number of Bounding Boxes')
plt.xlabel('Class')
plt.show()

# Show Random Sample Images
def show_random_images(folder, num_images=5):
    image_folder = os.path.join(folder, 'images')
    image_files = random.sample(os.listdir(image_folder), num_images)
    plt.figure(figsize=(15, 5))
    for idx, img_file in enumerate(image_files):
        img = cv2.imread(os.path.join(image_folder, img_file))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        plt.subplot(1, num_images, idx+1)
        plt.imshow(img)
```

```
        plt.title(img_file)
        plt.axis('off')
    plt.suptitle('Random Sample Images from Training Set', fontsize=16)
    plt.show()

show_random_images(os.path.join(base_path, 'train'), num_images=5)
```