**Detailed Summary of Fracture Identification Using YOLO and Log Data**

Vahid Dehdari

For this project, I experimented with several methods:

1. I tried the Faster R-CNN (Faster Region-based Convolutional Neural Network) model with a ResNet-50 backbone and a Feature Pyramid Network (FPN) for feature extraction.
2. I tested Mask R-CNN, which extends Faster R-CNN by adding a segmentation head to predict pixel-level masks for objects.
3. Finally, I experimented with the YOLO (You Only Look Once) segmentation model.

Among all the methods, I achieved the best results with YOLO. Therefore, this report focuses on fracture detection using the YOLO segmentation model.

This document provides a detailed explanation of the workflow and steps performed in the notebooks **YOLO\_train.ipynb** (for model training) and **YOLO\_prediction.ipynb** (for fracture detection and integration with log data). The process includes data preparation, model training, prediction, and post-processing to confirm fractures using anomalies detected in log data.

**1. Training the YOLO Model (YOLO\_train.ipynb)**

**1.1. Data Preparation**

1. **Dataset Structure**:
   * The dataset is organized into training and validation subsets (after shuffling), each containing images and their corresponding YOLO-format annotations.
   * Annotations are stored as .txt files with class labels and normalized bounding box coordinates.
2. **Data Augmentation**:
   * Basic augmentations like flipping, scaling, and rotations are applied to enhance the model's generalizability.
3. **Configuration**:
   * A YAML configuration file defines the paths to the dataset, number of classes (1 for fractures), and other training parameters.

**1.2. Model Configuration and Initialization**

1. **Model Selection**:
   * The YOLO model is selected (e.g., YOLOv8s, YOLOv8m, or YOLOv8l) depending on computational resources.
2. **Hyperparameter Tuning**:
   * Key hyperparameters used during training:
     + **Batch size**: 16
     + **Image size**: 640 x 640 pixels
     + **Learning rate**: 0.00009
     + **Epochs**: 200

**1.3. Training Execution**

1. **Process**:
   * The model is trained using the training data, and validation metrics (mean Average Precision - mAP) are calculated after each epoch.
   * Training loss, precision, recall, mAP, and F1 score are logged for monitoring.
2. **Output**:
   * The best model weights (best.pt) are saved based on the highest F1 score on the validation set.

**1.4. Validation and Results**

1. The trained model is evaluated on a validation set to ensure its ability to generalize to unseen data.
2. Visualizations of detected bounding boxes on validation images are saved for review.

**2. Prediction and Post-Processing (YOLO\_prediction.ipynb)**

**2.1. Loading the Model**

1. The trained YOLO model (best.pt) is loaded for inference.
2. The confidence threshold is set to 0.35 to filter out very low-confidence detections. This value is based on optimal F1 score during training.

**2.2. Test Data Preparation**

1. **Images**:
   * Test images are preprocessed (resized to 640 x 640 pixels) and fed into the model for prediction.
2. **Log Data**:
   * Depth-specific well log data is used alongside the images. Key features include:
     + Sonic travel time (DT)
     + Resistivity (ILD)
     + Porosity (POR\_diff)
3. **Anomaly Detection**:
   * An autoencoder model is applied to the log data to identify anomalies.
   * Anomalies are flagged based on reconstruction error exceeding a predefined threshold.

**2.3. Fracture Detection**

1. **Model Prediction**:
   * The YOLO model outputs bounding boxes, class probabilities, and confidence scores for each fracture.
2. **Depth Calculation**:
   * For each bounding box, the fracture depth is calculated using geometric relationships between the bounding box coordinates and the image scale.
3. **Post-Processing**:
   * Fractures are filtered based on:
     + Confidence score (threshold: 0.5)
     + Mask size (minimum area specific to wells T1 and T2)

**2.4. Integration with Log Data**

1. **Anomaly Confirmation**:
   * Detected fractures are cross-referenced with anomalies identified in the log data.
   * Fractures without corresponding anomalies are discarded to improve the reliability of predictions.
2. **Filtering and Sorting**:
   * Fractures are sorted by confidence scores in descending order.
   * Only the top fractures with the highest confidence and confirmed anomalies are retained.

**2.5. Visualization**

1. Original test images are displayed alongside predictions, with bounding boxes highlighting detected fractures.
2. Fracture depths and confidence scores are plotted for wells T1 and T2 to provide insights into prediction reliability.

**2.6. Result Compilation**

1. **Dataframe Creation**:
   * A dataframe is created to store the following for each well:
     + Fracture depth
     + Confidence score
     + Mask size (if available)
2. **Output Files**:
   * Results are saved as a CSV file for submission.
   * Model weights and configuration files are packaged into a ZIP file.

**3. Summary of Key Outputs**

1. **Trained Model**:
   * best.pt: The YOLO model weights with the highest F1 score.
2. **Prediction Results**:
   * CSV file containing fracture details (depth, confidence, etc.).
   * Visualizations of fracture detections.
3. **Packaged Results**:
   * ZIP file containing:
     + CSV file
     + Model weights
     + Configuration metadata

This process provides a robust pipeline for identifying fractures using a combination of YOLO-based image analysis and log data integration.