**Signature Recognition Documentation**

**Project Overview**

This project focuses on signature recognition using the YOLOv10 deep learning architecture. The goal is to train a model that can identify signatures and determine how closely a test image matches a known signature. The training process was completed on a dataset of 50 images, with the potential for training with a minimum of 25 images.

**Training Details**

* Training Time: 1.22 hours
* Hardware Used:
* GPU:
  + GTX 1070 (8GB VRAM)
  + CPU: Intel Core i7 6800k
  + RAM: 16GB
* Model: YOLOv10
* Epochs: 200
* Dataset: 50 images (40 training, 10 validation)
* Minimum Required Images: 25 images (more images improve accuracy)

**Requirements**

* Operating System: Any system that supports Python
* Python Version: 3.12 or above
* Libraries:
  + Ultralytics for YOLOv10
  + Tkinter for GUI
  + PIL for image processing

**How the Model Works**

The model uses YOLOv10 to detect and classify signature images. It is designed for scenarios where you need to verify a signature against a known set of signatures. The model is trained on one true class (the correct signature) and can compare new signatures against this class to determine similarity.

**Key Features:**

* Single True Class: The dataset should contain only one class representing the true signatures.
* Matching Confidence: The model outputs a confidence score indicating how well the test image matches the original signatures.

**Training the Model**

**Training Process**

1. Prepare Dataset: Collect a dataset of at least 25 images of the signature. The more images you provide, the better the model will perform.
2. Set Up the Environment: Clone the repository, install dependencies, and set up your environment by following the instructions in Readme.md of github repositry.
3. Train the Model: Use the provided GUI application to select the .yaml file for your dataset and start training. The model was trained for 200 epochs in this project, but you can adjust the number of epochs based on your needs.
4. Training Time: On the GTX 1070 with the provided dataset, the training took approximately 1.22 hours.

**GUI Application**

The GUI simplifies the process of training and testing the model:

* Train a New Model: Select the YAML file and train the model with your dataset.
* Test an Existing Model: Select a pre-trained weights file and run the model on a test image. The GUI will display the confidence score indicating how closely the test image matches the original signature.

**Testing the Model**

After training, you can use the GUI to test your model on new images:

1. Select Weights: Choose a pre-trained weights file.
2. Upload Test Image: Upload the signature image you want to test.
3. Get Results: The model will output the confidence score for how well the test image matches the known signature.

**Best Practices for Model Performance**

* Image Count: A minimum of 25 images is required for effective training. The more varied and accurate the signatures, the better the model's performance.
* Class Consistency: Ensure the dataset contains only one true class (the signature you want to recognize).
* Preprocessing: Clean and preprocess your images for best results. Resize images consistently to avoid variations that could affect model performance.

**Conclusion**

This project provides a simple and effective solution for signature recognition using YOLOv10. With the provided GUI, users can easily train a model on their dataset and test it on new signatures. The model outputs a confidence score that reflects how well a signature matches the known class, making it ideal for scenarios like signature verification and document authentication.