**Gloved vs Bare Hand Detection**

**Project Overview**

This project implements an object detection pipeline to automatically detect **gloved** and **bare hands** in images. It is designed for safety compliance in industrial settings, where ensuring workers wear gloves is critical. The system can process images from cameras or snapshots and outputs both **annotated images** and **JSON logs** of detections.

The output consists of:

* Annotated images showing bounding boxes around hands.
* JSON logs storing details of each detection, including class, confidence, and bounding box coordinates.

**Dataset**

* **Source:** The dataset was obtained from **Kaggle.** The dataset was downloaded and saved in Google Drive, then mounted in Colab.

Link - <https://www.kaggle.com/datasets/yashdev01/gloves-and-bare-hands-datasets?select=hands>

* **Classes:**
  1. gloved\_hand (gloves)
  2. bare\_hand (hands)

**Preprocessing**

* Images are converted into YOLOv8 format, which includes:
  + .jpg images
  + .txt labels in YOLO format: <class\_id> <x\_center> <y\_center> <width> <height> normalized between 0–1
* Dataset is split into:
  + **80% train, 10% validation, 10% test**  
    This ensures the model can learn patterns, validate performance, and test generalization.

**Model**

**Choice**

* YOLOv8n (Nano) was selected because:
  + It provides fast training and inference, suitable for limited hardware and demonstration purposes.
  + Pretrained on COCO dataset, allowing transfer learning to improve performance on small datasets.
* Alternative models like Faster R-CNN, SSD, or EfficientDet could also be used, but YOLOv8 strikes a good balance between speed and accuracy.

**Training**

* Epochs: 5 (enough for demonstration; more epochs recommended for production)
* Image size: 640x640 (standard for YOLOv8)
* Batch size: 16
* Project folder: /content/glove\_detection
* Hardware: Colab GPU or equivalent

**Fine-tuning**

* The pretrained weights (yolov8n.pt) were fine-tuned on the custom dataset.
* Training adjusts model weights to recognize hands with or without gloves specifically.

**Evaluation Metrics**

After training, the model is evaluated using the **validation set**. Evaluation metrics include:

| **Class** | **Images** | **Instances** | **Precision** | **Recall** | **F1 Score** | **mAP50** | **mAP50-95** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **gloved\_hand** | 56 | 56 | 0.996 | 1.000 | 0.998 | 0.995 | 0.987 |
| **bare\_hand** | 73 | 73 | 0.999 | 1.000 | 0.999 | 0.995 | 0.995 |

**Detection / Inference Pipeline**

**Overview**

The detection\_script.py handles detection on any folder of images. Steps:

1. Load the trained YOLOv8 model.
2. Read all images from the input folder.
3. Perform detection using model predictions.
4. Annotate images with bounding boxes and class labels.
5. Save annotated images in detection\_output/.
6. Save JSON logs in logs/.

**CLI Arguments**

The detection script (detection\_script.py) supports several **command-line arguments** to make it flexible and easy to use without modifying the code.

* --input: Folder of images to process
* --output: Folder for annotated images
* --logs: Folder for JSON detection logs
* --weights: Path to trained YOLOv8 weights
* --confidence: Minimum confidence threshold (0–1)
* --workers: Number of threads for parallel processing

**Performance Enhancements**

* **ThreadPoolExecutor** is used for **multi-threaded inference**, making detection faster.
* Annotated images allow visual verification, while JSON logs provide structured data for further analysis.

**What Worked and What Didn’t**

**What Worked**

* **Gloved vs Bare Hand Detection:** The YOLOv8 model successfully detects both classes with good accuracy on most validation images.
* **Annotation and JSON logs:** Each image is correctly annotated with bounding boxes, and logs are saved in structured JSON format.
* **CLI interface:** Flexible command-line arguments allow running detection on any folder of images without changing the code.
* **Multi-threading:** Using the --workers argument speeds up inference for large datasets.
* **Evaluation metrics:** Precision, recall, and mAP are correctly calculated after training, providing quantitative performance insights.

**What Didn’t Work / Limitations**

* **Small dataset:** Limited number of images may reduce generalization to new hand poses or lighting conditions.
* **Challenging scenarios:** Occluded hands, overlapping hands, or unusual angles may sometimes be missed or misclassified.
* **No augmentation:** Training without image augmentation may reduce robustness.
* **Inference on videos:** Current script handles images only; real-time video stream detection is not implemented.