SERVE-SMART HACKATHON-Jagriti'26

Team-AI_Squad_NITW

Team Members

Aditya S (Team Leader)

M Avinash Reddy

S Sriharsha

Institute: National Institute of Technology, Warangal

Introduction

We built a ML-based classifier model to classify a given image among 12 different military/civilian-based classes. The code is easy to run with open-source runtime environment. The details of the model are mentioned in further sections.

ML-Model Architecture

- The model comprises to Two Layers.
- The top layer makes use of Weighted Voting Ensemble Model which takes the final decision by taking the weighted mean of the predictions made by constituent models in the architecture.
- At the bottom layer we decided to finally go ahead with the following 2 models based on testing of model performance.

1.YOLOv8n Training (Nano) – Standard model

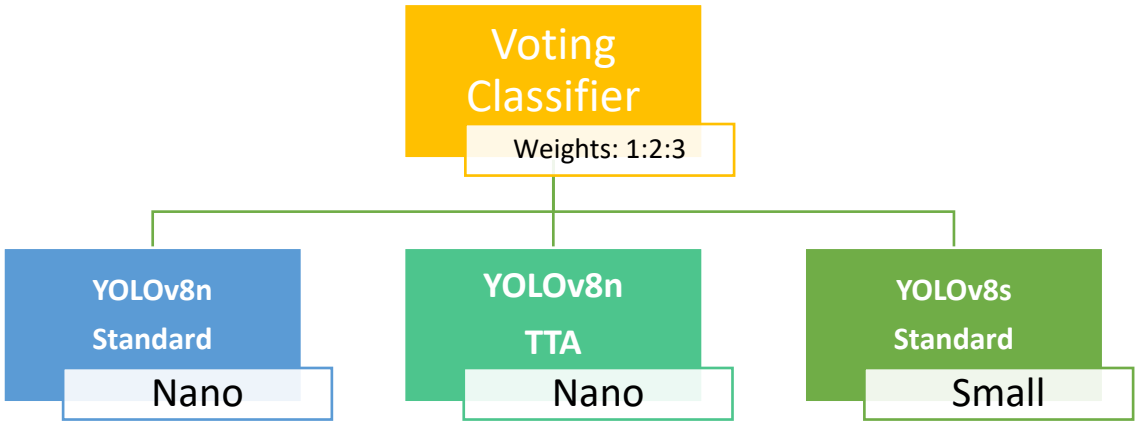
- 50 epochs
- Patience of 15
- resumable in case of interruptions

2. YOLOv8n (Nano) – Test Time Augmentation

- Another version YOLOv8n model being used for voting inference.
- Provides stability under varied conditions. Makes use of TTA pass.

3. YOLOv8s (Small Model) – Standard model

- ~11M parameters
- Higher accuracy than Nano
-



The final Weight distribution of the ensemble model was 1:2:3 for the corresponding predictions of these models based on experimentation.

Phases of Building Model

Training & Validating model: Use the training dataset to train the data on the models individually.

```
# ----------------------------------------------------------
# [5/8] TRAIN MODEL A (NANO - 50 EPOCHS - RESUMABLE)
# ----------------------------------------------------------
print("\n[5/8] Checking Model A (Nano)...")
run_name_n = 'military_nano_final'
last_pt_n = f'{PROJECT_PATH}/{run_name_n}/weights/last.pt'
best_pt_n = f'{PROJECT_PATH}/{run_name_n}/weights/best.pt'

if os.path.exists(last_pt_n):
    print("  ⚠️ Interrupted run detected! Resuming Model A...")
    model_n = YOLO(last_pt_n)
    model_n.train(resume=True)
elif os.path.exists(best_pt_n):
    print("  ✅ Model A already finished. Skipping.")
else:
    print("  🆕 Starting Model A from scratch...")
    model_n = YOLO('yolov8n.pt')
    model_n.train(data='/content/data.yaml', epochs=50, patience=15, imgsz=640,
batch=16,
                  project=PROJECT_PATH, name=run_name_n, exist_ok=True,
                  box=7.5, hsv_h=0.015, degrees=10.0, mosaic=1.0, verbose=True)
```

```
# ----------------------------------------------------------
# [6/8] TRAIN MODEL B (SMALL - 50 EPOCHS - RESUMABLE)
# ----------------------------------------------------------
print("\n[6/8] Checking Model B (Small)...")
run_name_s = 'military_small_final'
last_pt_s = f'{PROJECT_PATH}/{run_name_s}/weights/last.pt'
best_pt_s = f'{PROJECT_PATH}/{run_name_s}/weights/best.pt'

if os.path.exists(last_pt_s):
    print("  ⚠️ Interrupted run detected! Resuming Model B...")
    model_s = YOLO(last_pt_s)
    model_s.train(resume=True)
elif os.path.exists(best_pt_s):
    print("  ✅ Model B already finished. Skipping.")
else:
    print("  🆕 Starting Model B from scratch...")
    model_s = YOLO('yolov8s.pt')
    model_s.train(data='/content/data.yaml', epochs=50, patience=15, imgsz=640,
batch=16,
                  project=PROJECT_PATH, name=run_name_s, exist_ok=True,
                  box=7.5, hsv_h=0.015, degrees=10.0, mosaic=1.0, verbose=True)
```

2.Run inference on every image in test/images/ : We now make inferences on the images in testing data using weighted voting ensemble model.

```
    test_dir = os.path.join(true_root, 'test/images')
    image_files = sorted(glob.glob(os.path.join(test_dir, '*')))
    BATCH_SIZE = 50
    batches = [image_files[i:i+BATCH_SIZE] for i in range(0, len(image_files),
BATCH_SIZE)]

    print(f"   Processing {len(image_files)} images in {len(batches)} batches...
    weights = [1, 2, 3] # Nano, NanoTTA, Small

    for batch in tqdm(batches, desc="Inference"):
        try:
            # 1. Nano Standard
            res_n1 = model_n.predict(batch, imgsz=640, conf=0.15, augment=Fa
verbose=False)
            # 2. Nano TTA
            res_n2 = model_n.predict(batch, imgsz=800, conf=0.15, augment=Tru
verbose=False)
            # 3. Small Standard
            res_s1 = model_s.predict(batch, imgsz=640, conf=0.15, augment=Fa
verbose=False)

            # Fuse Results
            for i, r_base in enumerate(res_n1):
                boxes_list, scores_list, labels_list = [], [], []

                def extract(res):
                    if len(res.boxes):
                        boxes_list.append(res.boxes.xyxyn.cpu().numpy().tolist())
                        scores_list.append(res.boxes.conf.cpu().numpy().tolist())
```

Final Code: The final code has also been attached in the submission zip file. After pushing the dataset in the directory running the code will give the same output as attached in submission.

Final Directory structure of submission:

The final directory will consist of the following files:

1.Code.py (Source Code of model)

2.output.zip (Consists of all outputs of test data in a single zip file)

3.Report_AI_SQUAD_NITW.pdf (Report of the project)

Results:

On the train/validation dataset we achieved an accuracy of 54% which is significantly high and this can be understood from the fact that the probability of a random guess would have been just 8.33% since we classify among 12 classes. We beat this probability by approximately 6.5 times!

THANK YOU