Artificial Intelligence

Artificial Intelligence

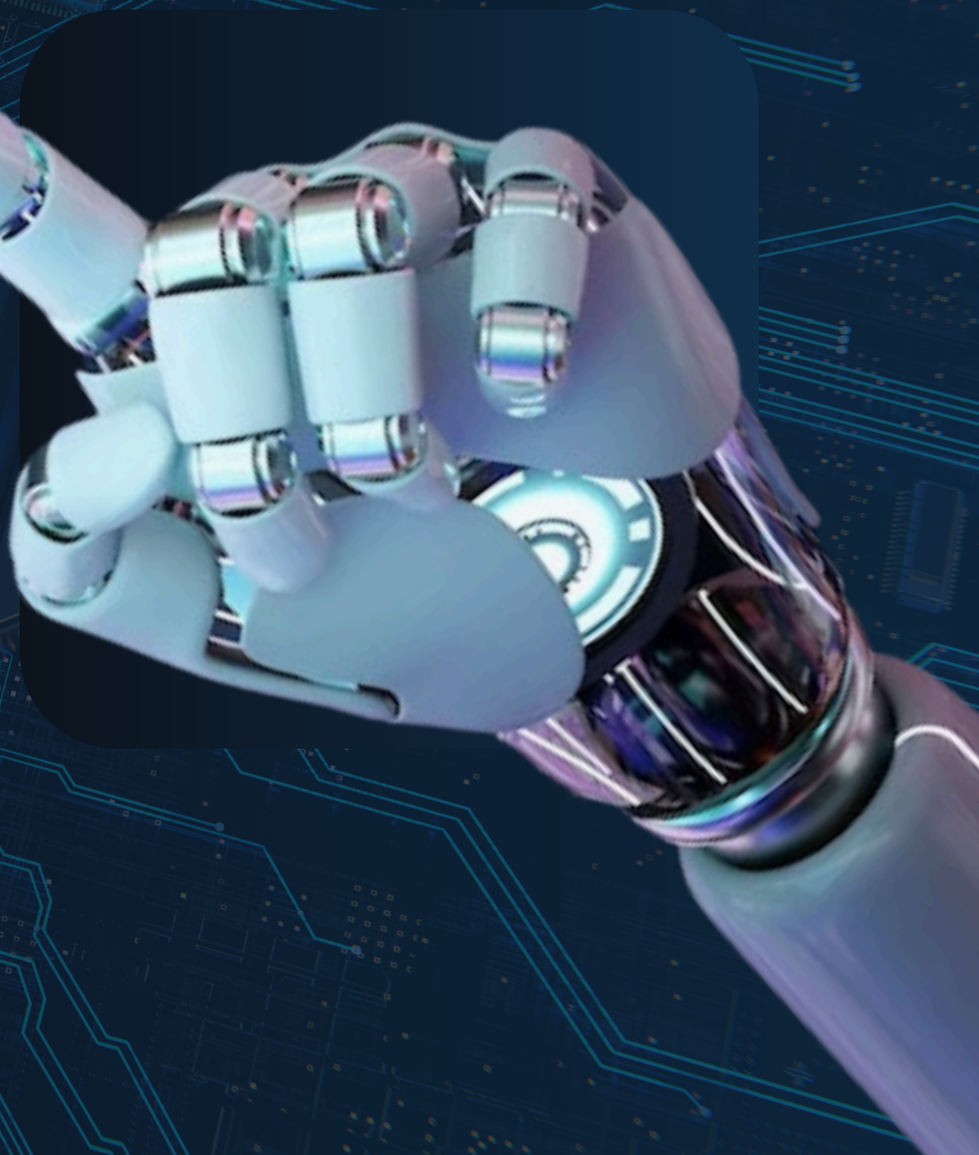
SIGN LANGUAGE (SIBI) REALTIME RECOGNITION

L200224288 Amri Zadi Hudaya

Artificial Intelligence

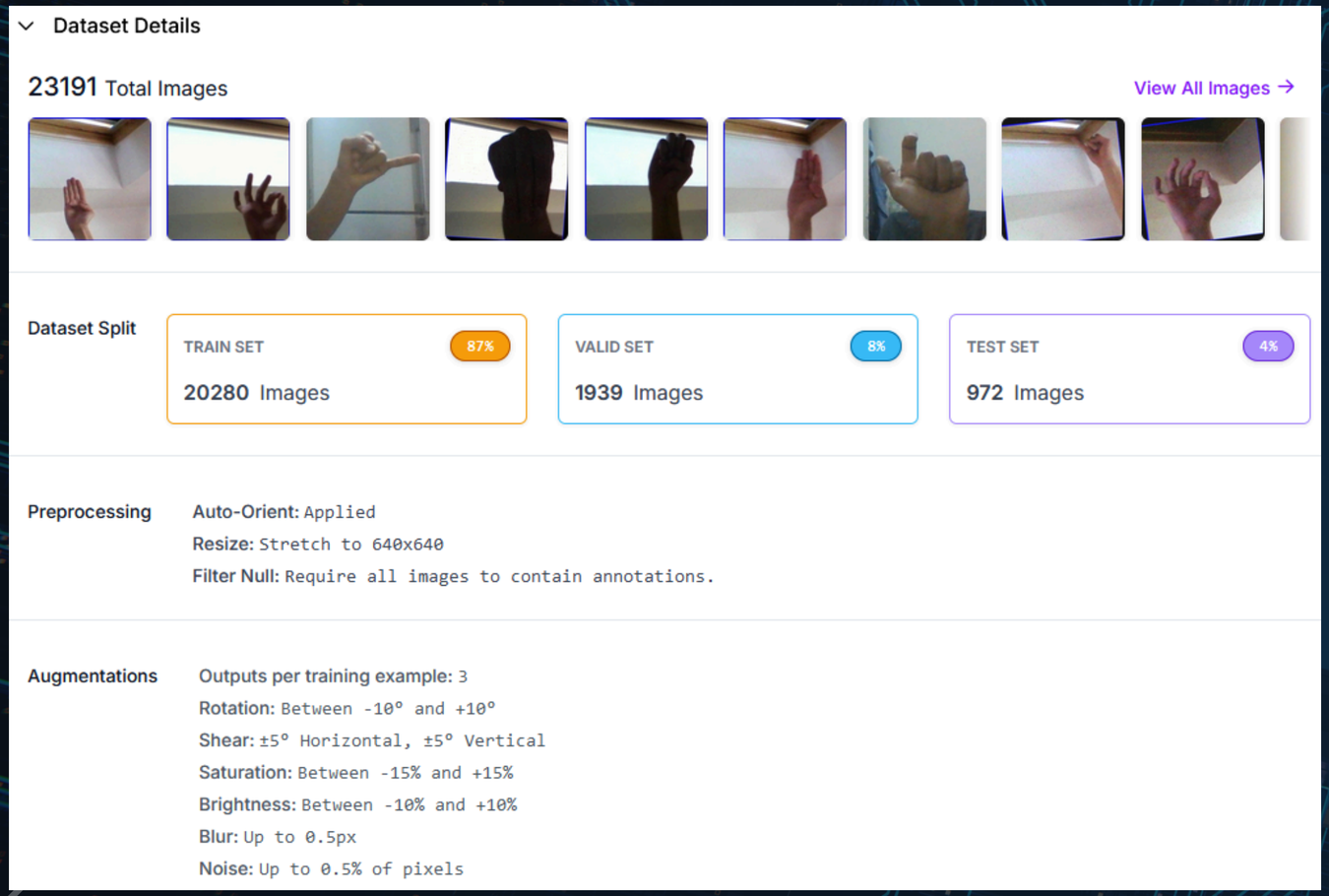
Introduction

This project is a prototype real-time translator for SIBI (Sistem Isyarat Bahasa Indonesia) built using the YOLO model and running on OpenCV. It focuses on translating 24 static hand signs representing the letters A, B, C, D, E, F, G, H, I, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, and Y. Letters J and Z, which require motion detection, are not implemented in this version. This project aims to demonstrate the potential of AI in facilitating communication through sign language translation.



Artificial Intelligence

DATASET



This dataset consists of 23,191 total images of hand signs, used for training a real-time SIBI translator. It is split into three subsets: 87% (20,280 images) for training, 8% (1,939 images) for validation, and 4% (972 images) for testing.

Preprocessing:

- Auto-Orient: Applied to standardize image orientation.
- Resize: Images are stretched to 640x640 pixels.
- Filter Null: Ensures all images contain valid annotations.

Data Augmentations:

Each training example generates three augmented versions with variations in:

- Rotation: ±10°.
- Shear: ±5° (horizontal/vertical).
- Saturation: ±15%.
- Brightness: ±10%.
- Blur: Up to 0.5px.
- Noise: Up to 0.5% of pixels.

This setup helps enhance the model's robustness by diversifying the input data

Artificial Intelligence

TRAINING MODEL

Download dataset from Roboflow

```
from roboflow import Roboflow

# Download dataset from Roboflow
api_key = "WM2DGQ6I2MwnMn0ViPz5"  # Replace with your Roboflow API key
workspace = "amzah"
project_name = "sibi_letterlevel_dataset-gniwb"
version_number = 2

print("Downloading dataset from Roboflow...")
rf = Roboflow(api_key=api_key)
project = rf.workspace(workspace).project(project_name)
version = project.version(version_number)
dataset = version.download("folder")  # Path to downloaded dataset
```

Training Model Using YOLOv11

```
# Train with YOLO v11 using a configurable model
model_choice = "yolo11m-cls"  # Choose from "yolo11n-cls", "yolo11s-cls", "yolo11m-cls", "yolo11l-cls"
epochs = 100
img_size = 640
patience = 5
dataset_path = "./sibi_letterlevel_dataset-2"

print("Training YOLO model...")
os.system(
    f"yolo task=classify mode=train model={model_choice}.pt data={dataset_path} epochs={epochs} imgsz={img_size} patience={patience}"
)
```

This training configuration sets up a YOLO v11 model for classification using the yolo task=classify command. with key parameters:

1. Model Selection (model_choice):
   - The Model is yolo11m-cls, indicating a medium-sized YOLO v11 classification model.
2. Hyperparameters:
   - epochs = 100: The model will train for 100 epochs.
   - img_size = 640: Input images are resized to 640x640 pixels.
   - patience = 5: Training will stop early if no improvement is seen after 5 epochs.

Artificial Intelligence

INFERENCE

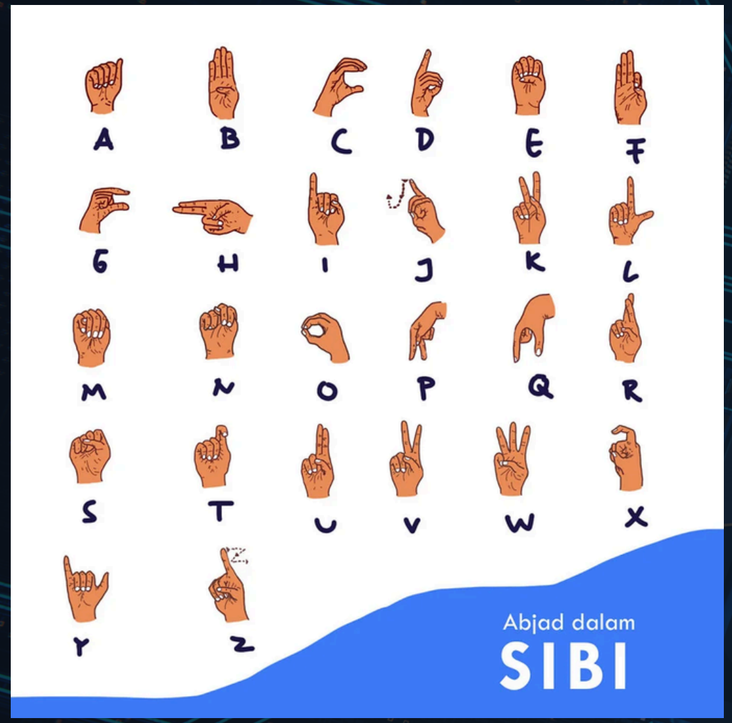
```python
import cv2
from ultralytics import YOLO


def detect_realtime():
    # Muat model yang telah dilatih
    # Sesuaikan path model
    model = YOLO(
        r'./runs/classify/train/weights/best.pt')

    # Buka kamera
    cap = cv2.VideoCapture(0)

    while True:
        ret, frame = cap.read()
        if not ret:
            print("Gagal membuka kamera!")
            break

        # Deteksi objek dalam frame
        results = model(frame)

        # Gambar bounding box pada frame
        annotated_frame = results[0].plot()

        # Tampilkan hasil
        cv2.imshow('Sign Language Translator', annotated_frame)

        # Tekan 'q' untuk keluar
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```

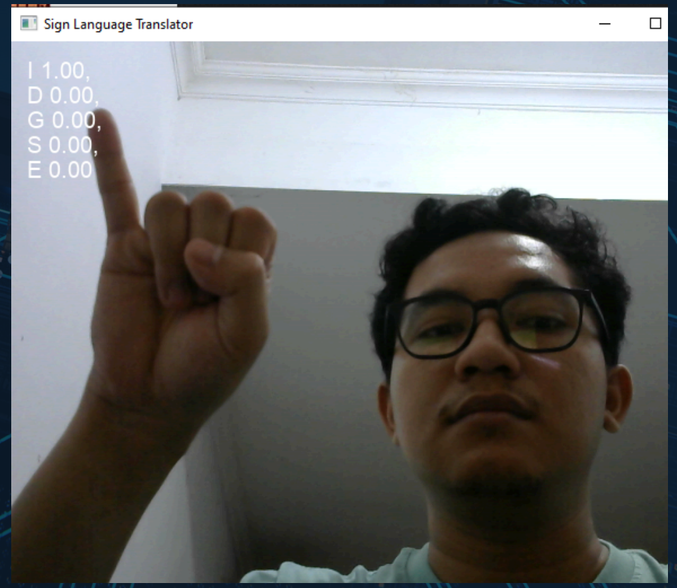
uses OpenCV (cv2) and YOLO (You Only Look Once) from the ultralytics library to detect objects in real-time via a webcam.

1. Imports
   - OPEN-cv2 python:
   - YOLO from ultralytics:
2. detect_realtime() function
   - Load the YOLO model:
   - Access the webcam:
   - cv2.VideoCapture(0) capturing real-time video frames.
3. Main processing loop
   - Continuously captures video frames from the webcam using cap.read().
   - The YOLO model processes each frame (results = model(frame)) to detect objects in the image.
   - Annotate frames:
   - Display the annotated frame:
   - The processed frame is displayed in a window titled "Sign Language Translator" using cv2.imshow.
   - Exit condition: The program listens for the 'q' key. If pressed,
4. Release resources
   - The webcam is released (cap.release()), and all OpenCV windows are closed (cv2.destroyAllWindows()).

Artificial Intelligence

DEMO





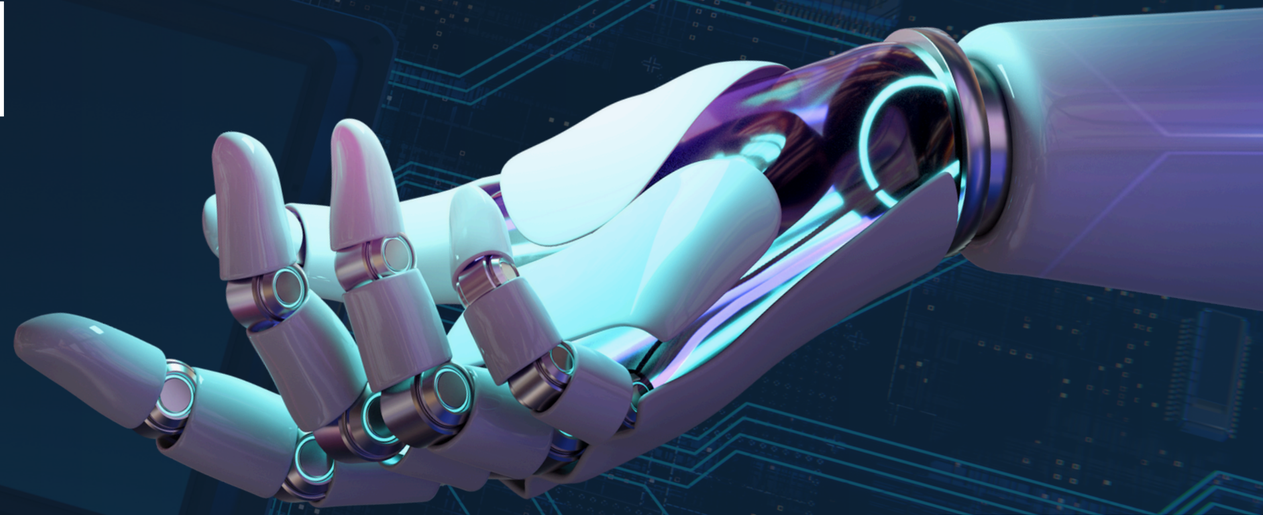

```
0: 640x640 G 0.82, E 0.09, D 0.07, O 0.01, T 0.01, 403.0ms
Speed: 20.0ms preprocess, 403.0ms inference, 0.0ms postprocess per image at shape (1, 3, 640, 640

0: 640x640 G 0.68, E 0.14, D 0.13, T 0.02, R 0.01, 404.0ms
Speed: 21.0ms preprocess, 404.0ms inference, 0.0ms postprocess per image at shape (1, 3, 640, 640

0: 640x640 G 0.63, E 0.23, D 0.10, O 0.01, T 0.01, 412.0ms
Speed: 20.0ms preprocess, 412.0ms inference, 0.0ms postprocess per image at shape (1, 3, 640, 640

0: 640x640 G 0.57, E 0.35, D 0.05, O 0.01, T 0.00, 439.0ms
Speed: 23.0ms preprocess, 439.0ms inference, 0.0ms postprocess per image at shape (1, 3, 640, 640

0: 640x640 G 0.82, E 0.09, D 0.08, T 0.01, O 0.00, 419.0ms
Speed: 20.0ms preprocess, 419.0ms inference, 0.0ms postprocess per image at shape (1, 3, 640, 640
```

Artificial Intelligence

Thank You

FOR YOUR ATTENTION

Any Question??



L200224288 Amri Zadi Hudaya