Wrocław University
of Science and Technology

Project Report: Scuba Diving Signals Recognition

Author: Rafał Cielenkiewicz

Supervisor: Wojciech Domski

Wrocław, 2024

Table of Contents

1. Project Overview

This project aims to develop a system capable of recognizing scuba diving hand signals in real time. The system consists of two main components:

1. **Hand Detection**: Utilizes YOLO (You Only Look Once) to detect hands in video frames.
2. **Gesture Classification**: Employs a custom-designed Convolutional Neural Network (CNN) to classify detected gestures.

The primary objective was to collect and use a self-created dataset to train the models, ensuring they perform well in real-time scenarios.

2. Project Components

2.1. Dataset

- **Source**: Self-collected photos of various scuba diving gestures.
- **Size**: 1080 images per gesture (600 with background, 480 without background).
- **Gestures**:
  - Down
  - Level
  - Low
  - OK
  - Stop
  - Up
- **Challenges**:
  - Variations in lighting.
  - Diverse and distracting backgrounds.

2.2. Hand Detection

- **Model**: YOLO (You Only Look Once).
- **Functionality**: Detects hands in real-time video streams and provides bounding boxes for detected regions.
- **Output**: Cropped hand regions are prepared for the next stage of processing.

2.3. Gesture Classification

- **Model**: A custom Convolutional Neural Network (CNN). I am using three convolutional layers, as presented:

```
Conv2D(32, (5, 5), activation='relu', padding='same',
 input_shape=input_shape),
BatchNormalization(),
MaxPooling2D(2, 2),

Conv2D(64, (5, 5), activation='relu', padding='same'),
BatchNormalization(),
MaxPooling2D(2, 2),

Conv2D(128, (3, 3), activation='relu', padding='same'),
BatchNormalization(),
MaxPooling2D(2, 2),


Flatten(),
Dense(256, activation='relu'),
Dropout(0.25),
Dense(num_classes, activation='softmax')
```

- **Key Features**:
  - Accepts 150x150 pixel images as input.
  - Utilizes convolutional layers for feature extraction and pooling layers to reduce dimensions.
  - Employs dropout layers for regularization to prevent overfitting.
  - Includes fully connected dense layers for classifying gestures.
- **Training**:
  - Data augmentation techniques, such as rotation, zoom, and horizontal flipping, were applied to enhance model robustness.

3. Workflow

1. **Video Capture**: The system receives a live video stream as input.
2. **Hand Detection**: YOLO detects and localizes hands in each frame, providing bounding boxes.
3. **Preprocessing**: Cropped hand regions are resized to 150x150 pixels and normalized to improve model performance.
4. **Gesture Recognition**: The CNN predicts the gesture class and confidence score for the detected hand regions.
5. **Output Display**: The recognized gesture and its confidence level are displayed on the video feed.

4. Tools and Technologies

- **Programming Language**: Python
- **Frameworks and Libraries**:
  - TensorFlow/Keras: For developing and training the CNN.
  - PyTorch: For implementing YOLO-based hand detection.
  - OpenCV: For image preprocessing and video stream handling.

5. Results and Data

5.1. Hand Detection

The YOLO-based hand detection model demonstrated high accuracy and real-time performance. The dataset was annotated using Roboflow, and YOLOv5 was trained to provide optimal results.

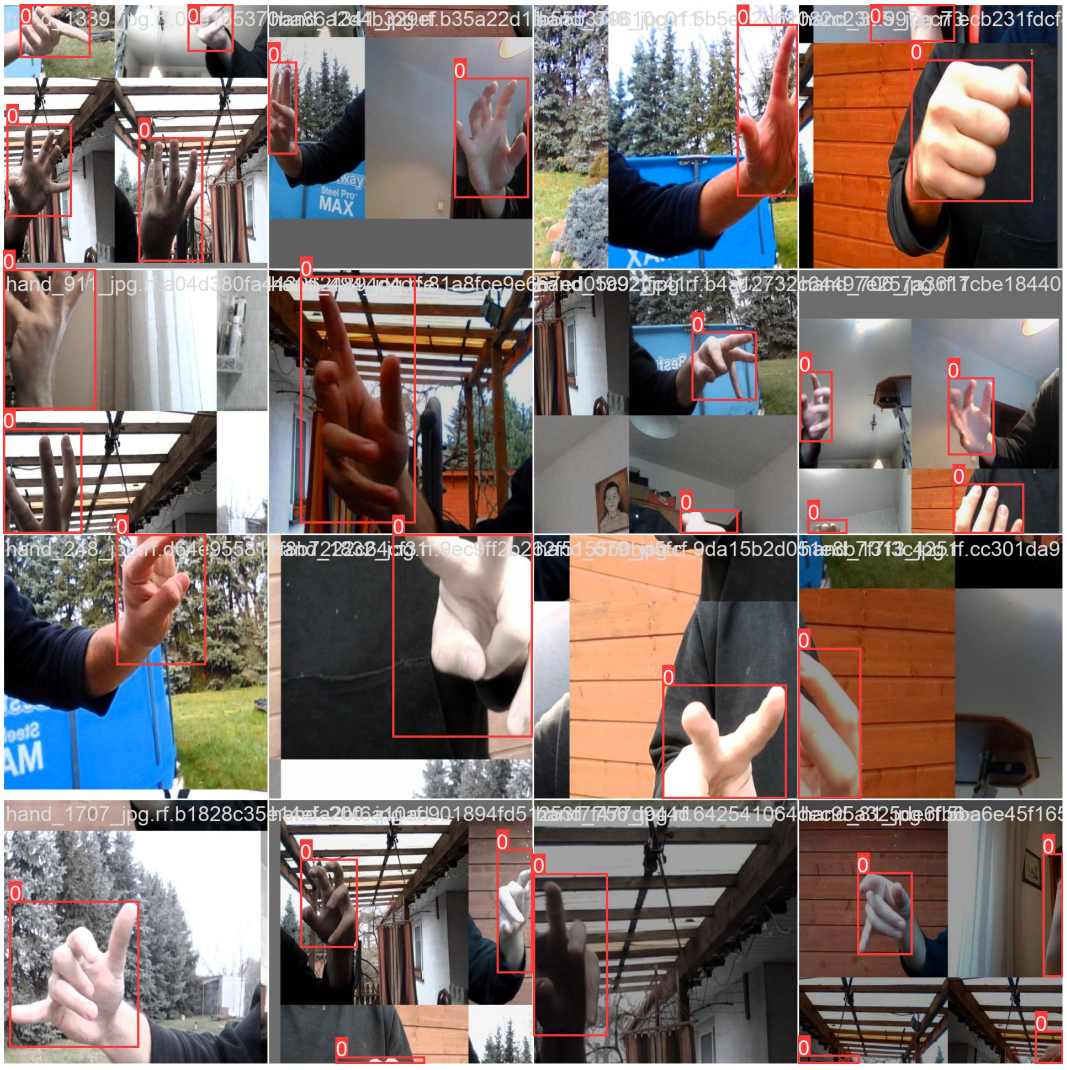

Figure 1: Example training data batch.

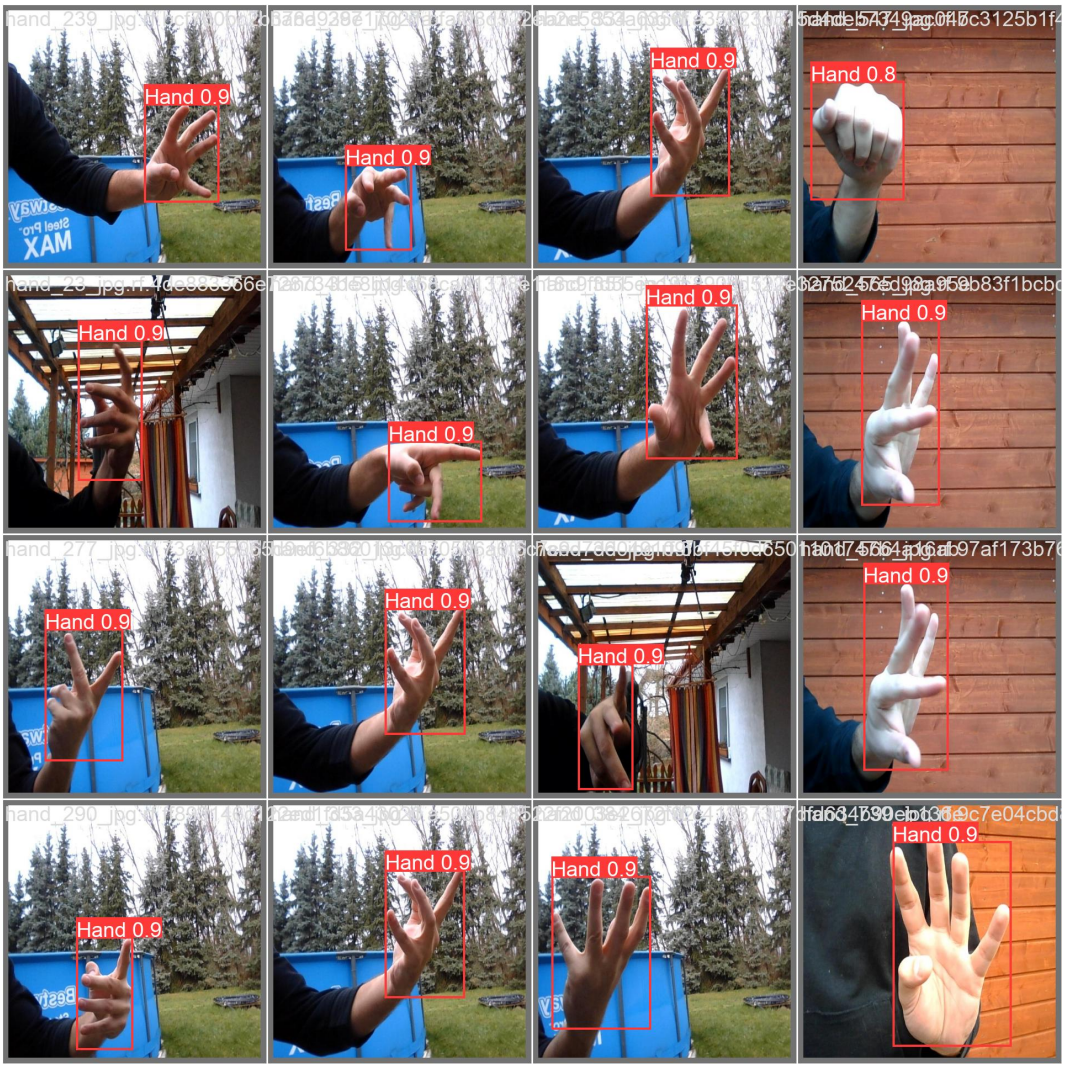

Figure 2: Example validation data batch.

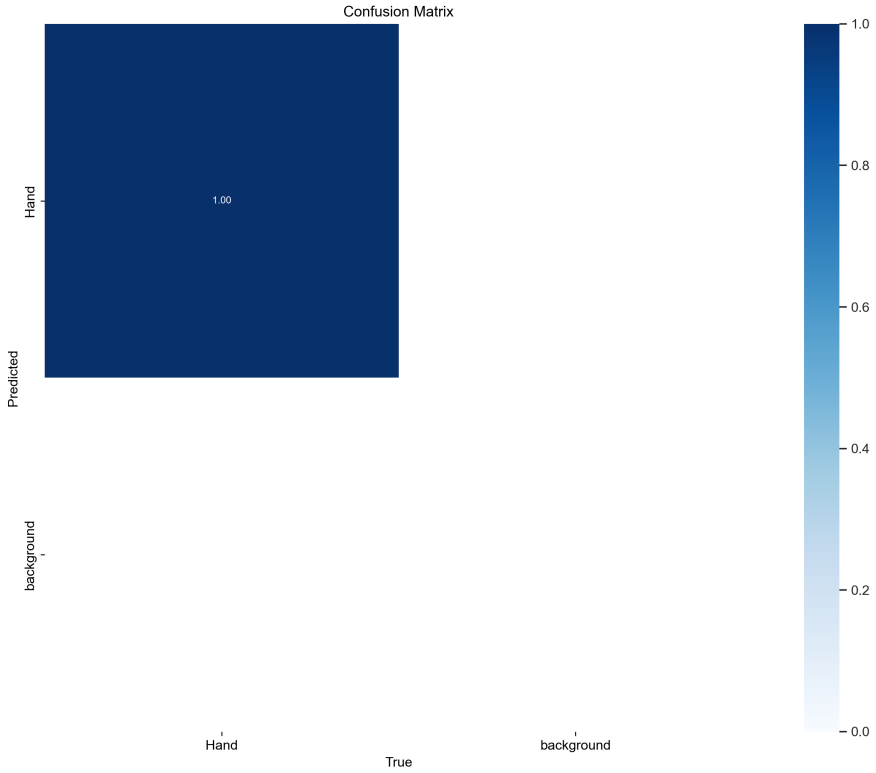

Figure 3: Confusion Matrix.

The figures demonstrate the effectiveness of the YOLOv5 model in detecting hands across diverse conditions, with good accuracy and acceptable real-time performance. The validation data highlights the model's ability to generalize well despite variations in lighting and background complexity. The confusion matrix, simplified due to the single-class nature of the detection task, confirms strong detection capabilities.

5.2. Gesture Recognition

Training the CNN required significant experimentation with the architecture and parameters. Adding 480 images without backgrounds significantly improved the clarity of gesture images and helped the model generalize better.

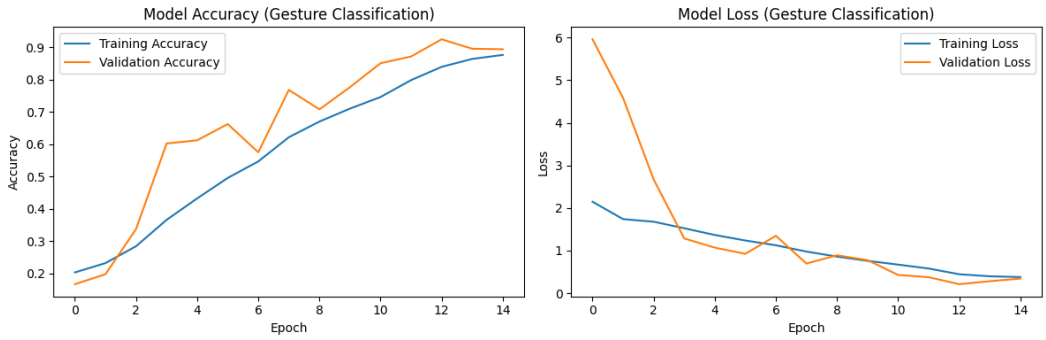

Figure 4: Accuracy and loss trends during training and validation.

(a) Image of a down gesture with background.

(b) Image of a down gesture without background.

Figure 5: Two example images showing the "down" gesture.

(a) Image of a stop gesture with background.

(b) Image of a stop gesture without background.

Figure 6: Two example images showing the "stop" gesture.

Gesture recognition results show that optimizing the CNN and enhancing the dataset were key to improving performance. Adding 480 images without backgrounds helped the model focus on gestures and generalize better. Figures illustrate clear examples of gestures with and without backgrounds, while accuracy and loss trends confirm decent progress during training and validation.

6. Conclusion

This project successfully integrates YOLO for hand detection and a CNN for gesture classification to create a functional real-time scuba diving signal recognition system. While the results are promising, the system occasionally struggles with:

- Poor lighting conditions.
- Low camera resolution.
- Complex or cluttered backgrounds.

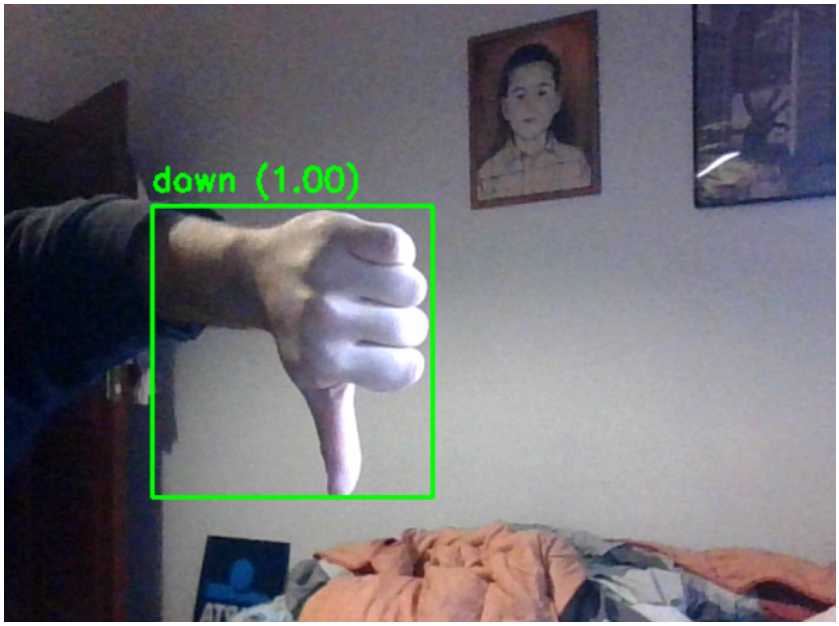

Figure 7: Down gesture.

Figure 8: Level gesture.

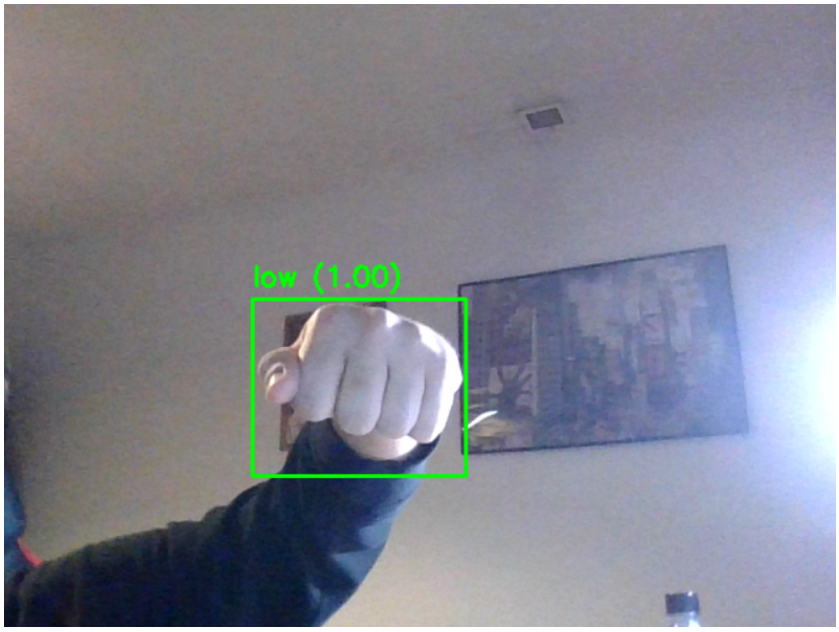

Figure 9: Low gesture.

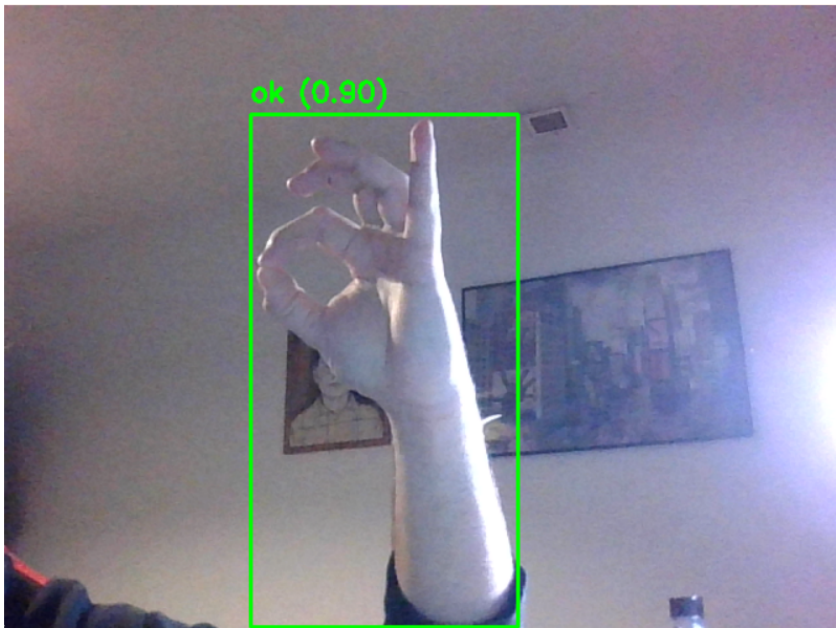

Figure 10: OK gesture.

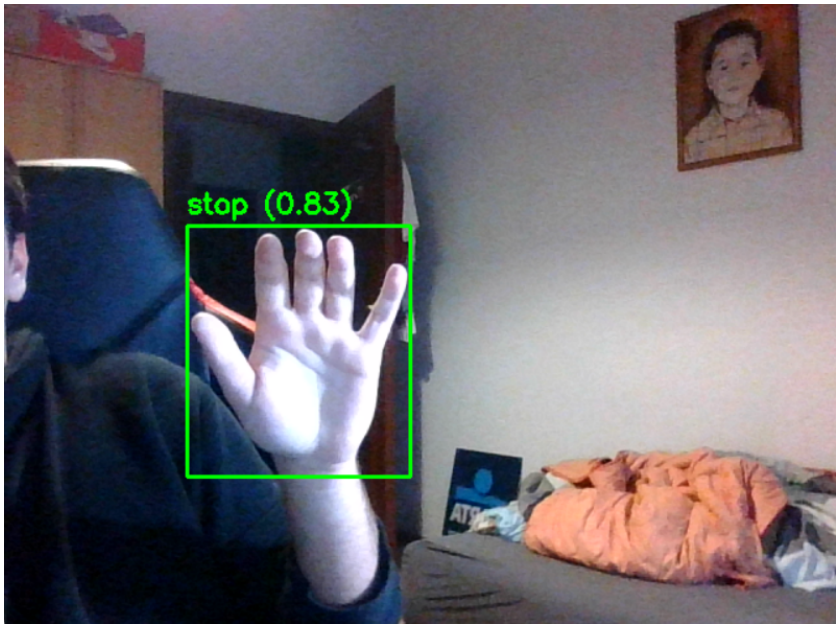

Figure 11: Stop gesture.

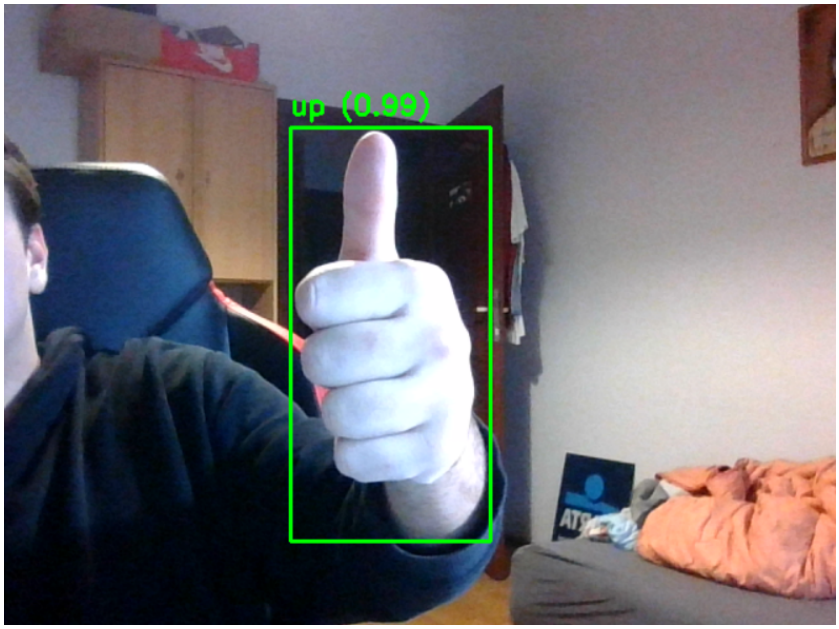

Figure 12: Up gesture.

The program successfully detects gestures and demonstrates its functionality in recognizing scuba diving signals in real time. However, it occasionally makes mistakes, particularly in challenging conditions. Achieving accurate recognition often required careful adjustments to the position of the hands and optimization of lighting conditions in the images.