**Human Interation Detection**

# Introduction

Understanding human interaction is a challenging task in the field of computer vision. Using a deep learning approach, this model was developed to address those problems by learning motion, pose detection, and whole-body extraction as features. PyTorch was used to train this model, to help classify the sword fight, hugging, grappling, ballroom, and mma.

# Datasets

I use both images and videos for the dataset; however, I give priority to the video because certain sub folders contain both images and videos, while others only contain images. Here some of the label name and actual name of dataset.

0: Sword (04\_sword\_part1)

1: Hugging (001\_hugging)

2: Grappling (02\_grappling)

3: Ballroom (07\_ballroom)

4: MMA (12\_mma)

The InteractionDataset class processes this data by extracting a fixed number of frames (default: 20 timesteps) from each sample. For videos, frames are sampled uniformly; if fewer than 20 frames are available, the last frame is repeated. For image sequences, images are sorted and sampled similarly, ensuring consistency across formats.

# Proposed Methods

**Feature Extraction:**

1. ResNet-50 model:

* Use ResNet-50 pretrain model to extracted global appearance features from each sampled frame.
* Frames are resized to 224x224 and normalized before being fed to ResNet.
* Output: A sequence of feature vectors (dimension 2048) for each input sample, representing the overall scene appearance over time.

1. Local Patch Appearance Features:

* Using YOLO11l-pose pretrain model to detect human poses and extract keypoint locations (specifically: nose, wrists, ankles) for up to two most prominent persons (based on bounding box area) in each frame.
* Handling Missing Detections: If fewer than two persons are detected, placeholder data (zero keypoints and confidences) are used for the missing person(s) to maintain consistent data structures.

1. Motion and Posture Features:

* This part is used to study the human movement and key parts over time.
* Calculate Displacement (Motion):
  + If t > 0 and the keypoint positions at t and t-1 are valid (not NaN), calculate the change in x (dx) and y (dy) coordinates: dx = positions[t, p, k, 0] - positions[t-1, p, k, 0], dy = positions[t, p, k, 1] - positions[t-1, p, k, 1].
  + If t == 0 or positions are invalid, set dx = 0.0 and dy = 0.0.

**Training Model:**

LSTM: because this model is used to understand the sequence of movements rather than treating each frame in isolation. Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN), are specifically designed to process sequential data.

# Experiments and Results

## Setup

**Software & Libraries:** The implementation relies on several key Python libraries:

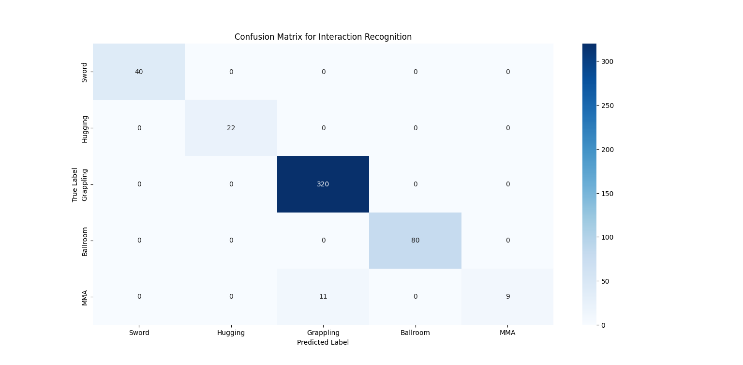
* **PyTorch (torch, torchvision):** The core deep learning framework used for model definition, automatic differentiation, training loops, and GPU acceleration. torchvision provides the pre-trained ResNet-50 model and image transformation utilities.
* **Ultralytics (ultralytics):** Used to load and run the YOLOv11-pose (yolo11l-pose.pt) model for human pose estimation.
* **OpenCV (cv2):** Essential for video and image input/output operations (reading frames, resizing, padding).
* **NumPy (numpy):** Used for numerical operations, particularly handling arrays for keypoint coordinates and intermediate feature storage before conversion to PyTorch tensors.

## Traing Protocols:

**LSTM-based model:** processes temporal sequences of the three feature types (full-body, local patches, motion/posture), combining them via fully connected layers for final classification into interaction categories.

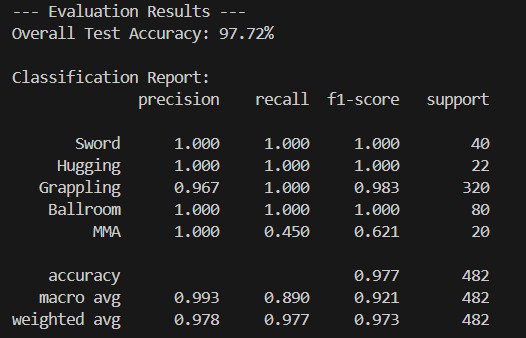
**Training and evaluation**: The script supports training on labeled data with a CrossEntropyLoss criterion and Adam optimizer, followed by testing on a separate dataset, reporting accuracy.

## Evaluation Protocols



The result show that all of the data prediction was received the perfect score across most of the data exclude MMA:

* "MMA", it only correctly guessed "MMA" 9 times out of 20. It often confused "MMA" with "Grappling" (11 times).



This model able to achieve the overall (97.72%). It performs exceptionally well in classifying "Sword," "Hugging," "Ballroom," and "Grappling" interactions, achieving near-perfect precision, recall, and F1-scores for most of these. However, the model shows weaker performance specifically in **recall for "MMA"**, meaning it misses a significant portion of actual "MMA" interactions, even though when it *does* predict "MMA", it's always correct.

To summarize this score, I feel that while the data in MMA is significantly less than that in grappling, the postures in grappling and MMA are very similar, which is causing this bias.

# Conclusion

This script provides a solid foundation for interaction recognition, blending established computer vision techniques (YOLO for pose estimation, ResNet for feature extraction) with temporal modeling (LSTM).

By integrating with full body, local, and motion help this model to capture both spatial and temporal aspects of interactions.

In my future study, I plan to use an openpose pretrained model and a way to validate the model:

* **OpenPose**: it excels at detecting human body, hand, and facial key points in fine detail, making it an ideal candidate for improving the model’s ability to recognize nuanced movements.
* Enhancing Model Validation and threshold:
  + **Early Stopping with Specific Thresholds:** It stop when validation loss stops decreasing for a set number of epochs. This ensures the model doesn’t overtrain and wastes computational resources.
  + **Cross-Validation**: To better assess how well the model generalizes, I’ll use k-fold cross-validation.