1. Project Title:

Sports Semantic Action Search System

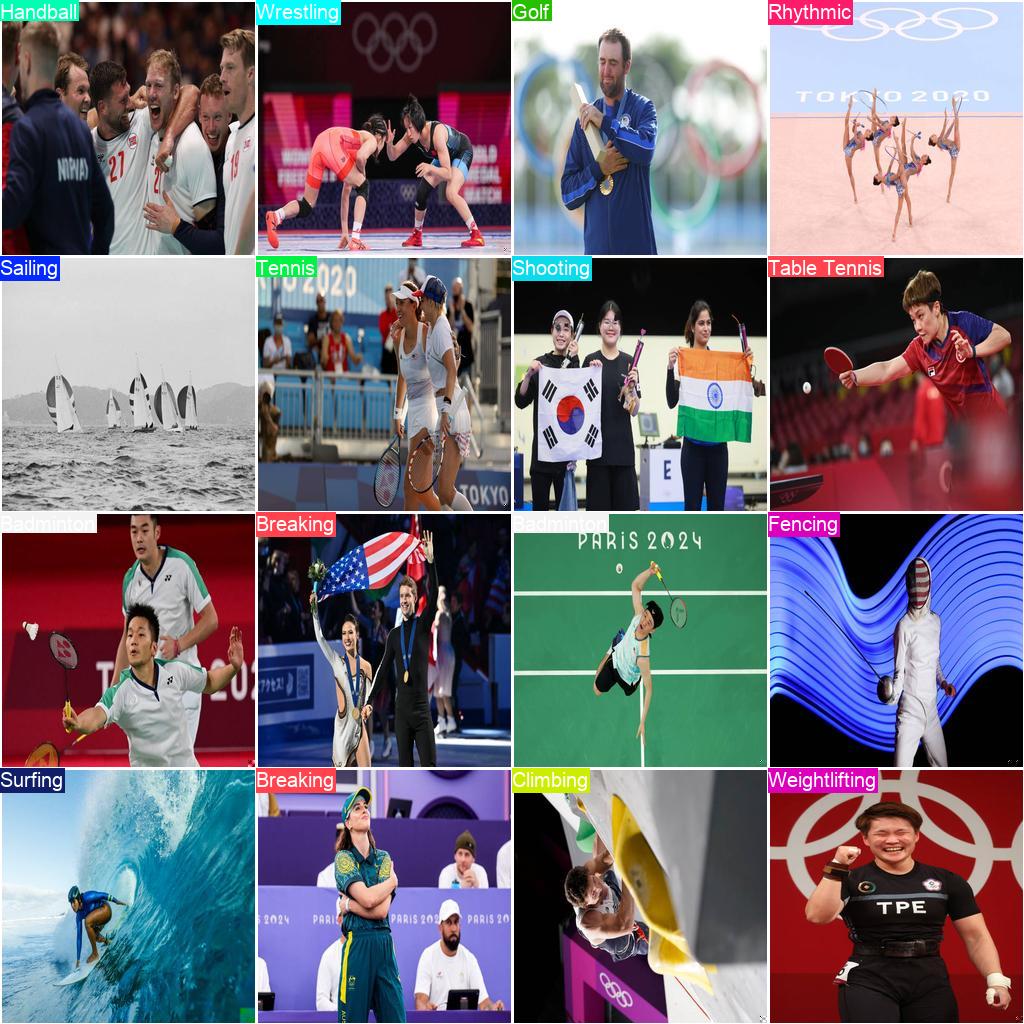
1. Name(s) of the Student(s) along with netid:
   1. Kuei-Yu Tsai kxt230002
   2. Yun-Hao Lee yxl230028
2. Goal of the Project:

Develop a semantic search system that enables users to input queries describing specific Olympic sports (such as basketball, badminton, table tennis, swimming, etc.) and retrieve relevant, timestamped segments from Olympic game videos where these events occur. The system will leverage advanced computer vision techniques to automatically recognize and extract relevant segments, ensuring accurate retrieval of specific sports actions within the videos, such as basketball plays, badminton rallies, or swimming strokes.

1. Description of the Project:
   1. This project will implement a content-based video retrieval system that allows users to search for specific sports actions within Olympic game footage. The system will analyze video frames using a pre-trained deep learning model to detect and classify various game actions. Each classified action will be indexed with timestamps, enabling users to query the system with semantic descriptions of the actions they are interested in. The search results will be displayed as a list of relevant video segments along with their corresponding timestamps.
   2. Steps involved:
      1. Action Detection and Classification: Utilize computer vision techniques (transfer learning with YOLO model) to detect and classify different game actions in Olympic footage.
      2. Action Segmentation: Segment the video into time intervals that correspond to various sports actions based on the detected and classified activities.
      3. Query Processing: Implement a semantic search engine that processes user-generated queries and matches them to the appropriate video segments by understanding the context of the actions described.
      4. Result Presentation: Display the retrieved video segments along with timestamp information, allowing users to view relevant moments. Optionally, the system could provide direct playback of the selected video segments for enhanced user experience.
2. Why We Want to Do This Project & Technical Capabilities:

We aim to develop this project to address the need for efficient video search in sports, particularly within vast Olympic game footage. Manually finding specific moments is time-consuming, and a semantic search system can streamline the process for fans, analysts, and researchers by quickly retrieving relevant game actions. Leveraging advanced computer vision techniques like CNNs or YOLO for action detection and classification, along with semantic search capabilities, the system will allow users to query natural language descriptions and retrieve timestamped video segments, providing a fast, scalable solution for sports content retrieval.

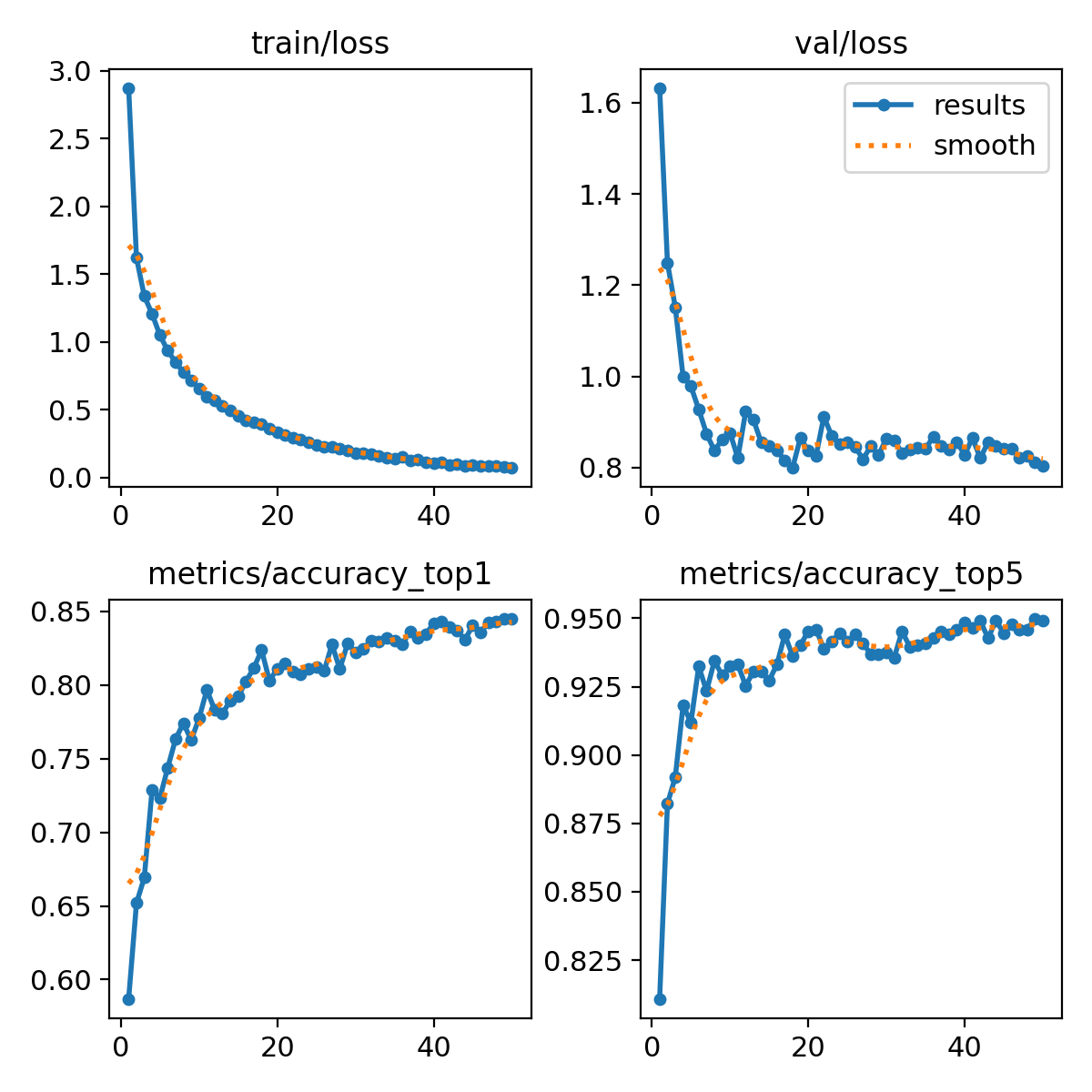
1. Wishlist:
   1. Advanced Feature: Enhance the NLP capabilities of the search engine to handle more complex queries, such as “Show me a sequence of swimming strokes followed by a flip turn” or “Find a badminton rally ending with a smash.”
   2. Optional Feature: Implement slow-motion playback for key Olympic moments, such as a high dive or sprint finish, or add a heatmap visualization of athlete movements during events like basketball or soccer matches.
2. Project Timeline:
   1. Weeks 1-2: Data Collection and Preprocessing (Partially completed now and definitely will be completed)
      1. Collect game videos and action annotations (Using [Getty Images](https://www.gettyimages.com/) to collect images from sports to train YOLOv11).
      2. Clean and preprocess data (Completed).
   2. Weeks 3-5: Action Detection Model Development (Completed with images)
      1. Train YOLOv11 to detect game actions (Completed).
      2. Validate performance on different game actions (Completed).
      3. Current results (Left for Label, Right for Prediction):

* + - 1. Training Summary:
         1. Model: YOLOv11-classification
         2. Task: Classification
         3. Epochs: 50
         4. Batch Size: 16
         5. Image Size: 256
         6. Training Data: 6174 images across 33 classes
         7. Validation Data: 1554 images across 33 classes
         8. Optimizer: AdamW with dynamically determined learning rate and momentum
         9. Loss Metrics:

Initial loss started at 2.875 in epoch 1 and steadily decreased to around 0.406 in epoch 17.

* + - * 1. Top-1 Accuracy: Improved from approximately 58.7% to 81.1% by epoch 17.
        2. Top-5 Accuracy: Improved from 81.1% to 94.4% by epoch 17.
      1. Key Features:
         1. TensorBoard: Enabled for visualizing training progress.
         2. Caching: New caches were created for both training and validation datasets.
         3. Layer Summary: Model has 151 layers with around 1.6 million parameters.
      2. Training Environment:
         1. CPU: Apple M2
         2. Python Version: 3.8.10
         3. PyTorch Version: 2.4.1+cu118
      3. Data Analysis:



* + - * 1. Training Loss:

The training loss decreases from 2.875 in epoch 1 to 0.07 in epoch 50, indicating that the model is learning effectively over time.

There is a general downward trend, suggesting that the model is improving and minimizing error during training.

* + - * 1. Top-1 Accuracy:

The top-1 accuracy increases from 58.69% in epoch 1 to around 84.49% in epoch 50. This is a significant improvement, suggesting that the model becomes better at correctly identifying the primary class.

There are minor fluctuations in accuracy during the epochs, but the overall trend is upward.

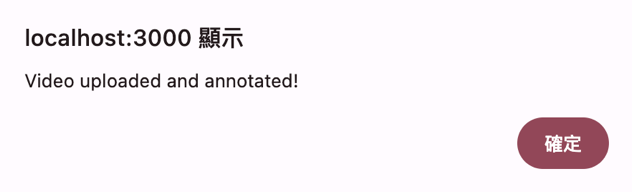
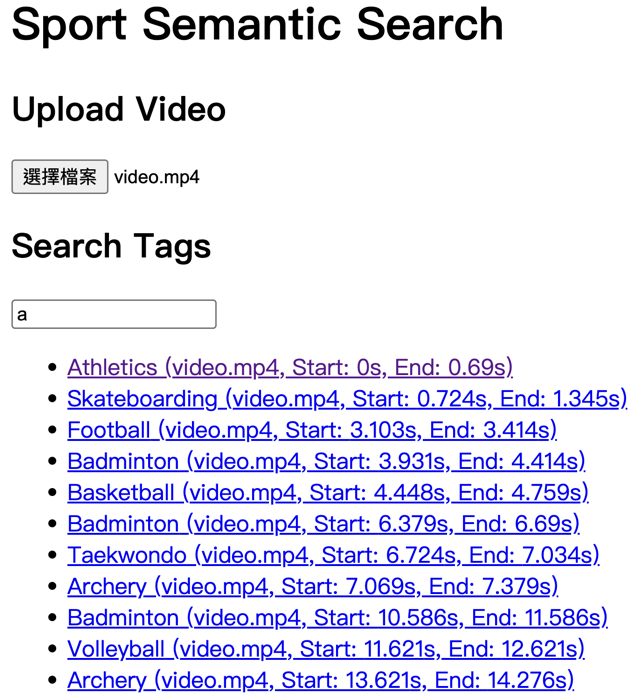
* + - * 1. Top-5 Accuracy:

The top-5 accuracy also shows a positive trend, moving from 81.08% in epoch 1 to about 94.91% in epoch 50. This means that even when the model doesn’t make the correct prediction as the top class, it often includes the correct class among its top 5 predictions, which is a positive sign of the model's predictive capability.

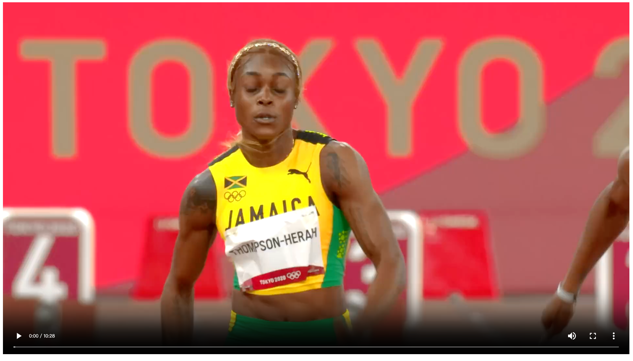
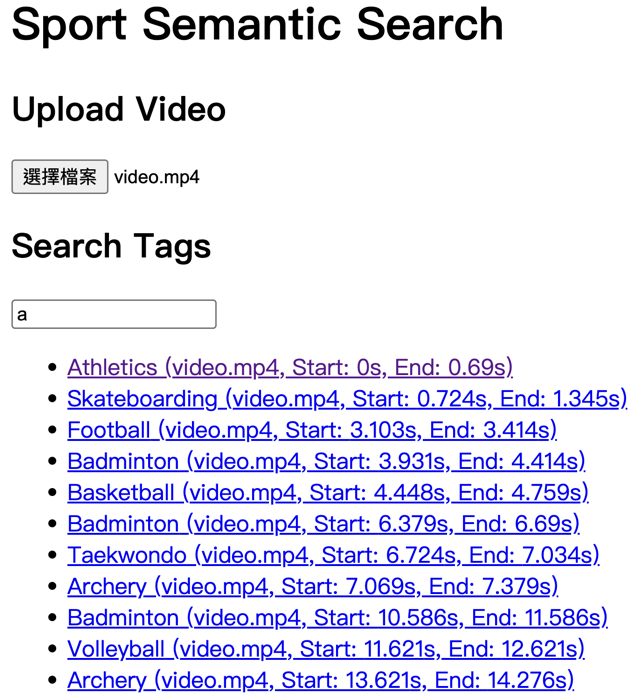
* + - * 1. Validation Loss:

The validation loss starts at 1.633 in epoch 1 and decreases to 0.804 in epoch 50. While the training loss decreases significantly, the validation loss shows a more modest decrease. This might indicate potential overfitting, as the training loss is dropping faster than validation loss.

* 1. Weeks 6-8: Semantic Search Engine Development
     1. Applying Image-Trained Model for Video Annotation
        1. Use the trained image-based action detection model to process video frames, focusing on accurately identifying action sequences within specific sports categories.
        2. For each video segment:
           1. Detect and annotate distinct actions in each frame, labeling the start and end frames to define action boundaries within each subclass of sports.
           2. Store metadata about each annotation, including timestamps, and action labels, for efficient retrieval.



* + 1. Building the Video Segment Database
       1. Design and implement a database schema tailored to index annotated video segments and their associated action labels.
       2. Define indexing and tagging criteria to enhance search efficiency, supporting fast access and retrieval of video segments based on specific actions and timestamps.
       3. Establish relationships between video segments, actions, and their metadata to facilitate advanced querying and filtering.
    2. Developing User Query Processing Module
       1. Create a module that translates user queries into actionable search terms that align with detected actions and indexed video segments.
       2. Build a semantic search component that interprets user intent and matches it to specific action types within the relevant sports categories.
       3. Develop a feature to suggest related categories based on query context, refining results to match users’ search intentions within the specified sport subclass.



* + 1. Integrating Action Detection Model with Search Engine
       1. Connect the action detection model to the semantic search engine, ensuring seamless data flow between video annotation, database indexing, and query processing.
       2. Ensure compatibility between model outputs and search engine inputs, addressing any inconsistencies that may arise in action labeling or segment indexing.
    2. System Structure:

https://github.com/xxittysnxx/sport-semantic-system

* + - 1. Frontend (Client Side)
         1. Video Upload:

Users upload a video via an HTTP POST request to the /upload route.

The video file is included in the request under the video key.

* + - * 1. Search Query:

Users can search for specific action tags through the frontend interface.

The search query is sent as a GET request to the /search route.

* + - 1. Backend (Server Side)
         1. Flask API:

The backend is built using Flask and exposes two main routes:

/upload for handling video uploads.

/search for handling search queries related to video annotations.

* + - * 1. Video Upload & Processing:

When a video is uploaded:

The video is saved to a specified directory.

The file path is stored in the database under the videos table.

* + - * 1. YOLO model is used for action detection on video frames:

Each video frame is processed to detect actions using a pre-trained YOLO model (best.pt).

Detected actions are stored with timestamps and labels for each frame.

* + - * 1. Annotation Creation:

Once all frames are processed:

The system identifies continuous frames with the same action and groups them into annotations.

Start and end times are assigned based on consecutive frames with the same action.

Annotations are saved to the annotations table in the database.

* + - * 1. Search Query Handling:

When a user types a search query:

The query is extracted, and a search is performed in the annotations table.

The query uses the SQL LIKE operator to find matching tags.

Results are returned, including metadata about the associated video, the detected action, and the timestamp range for the action.

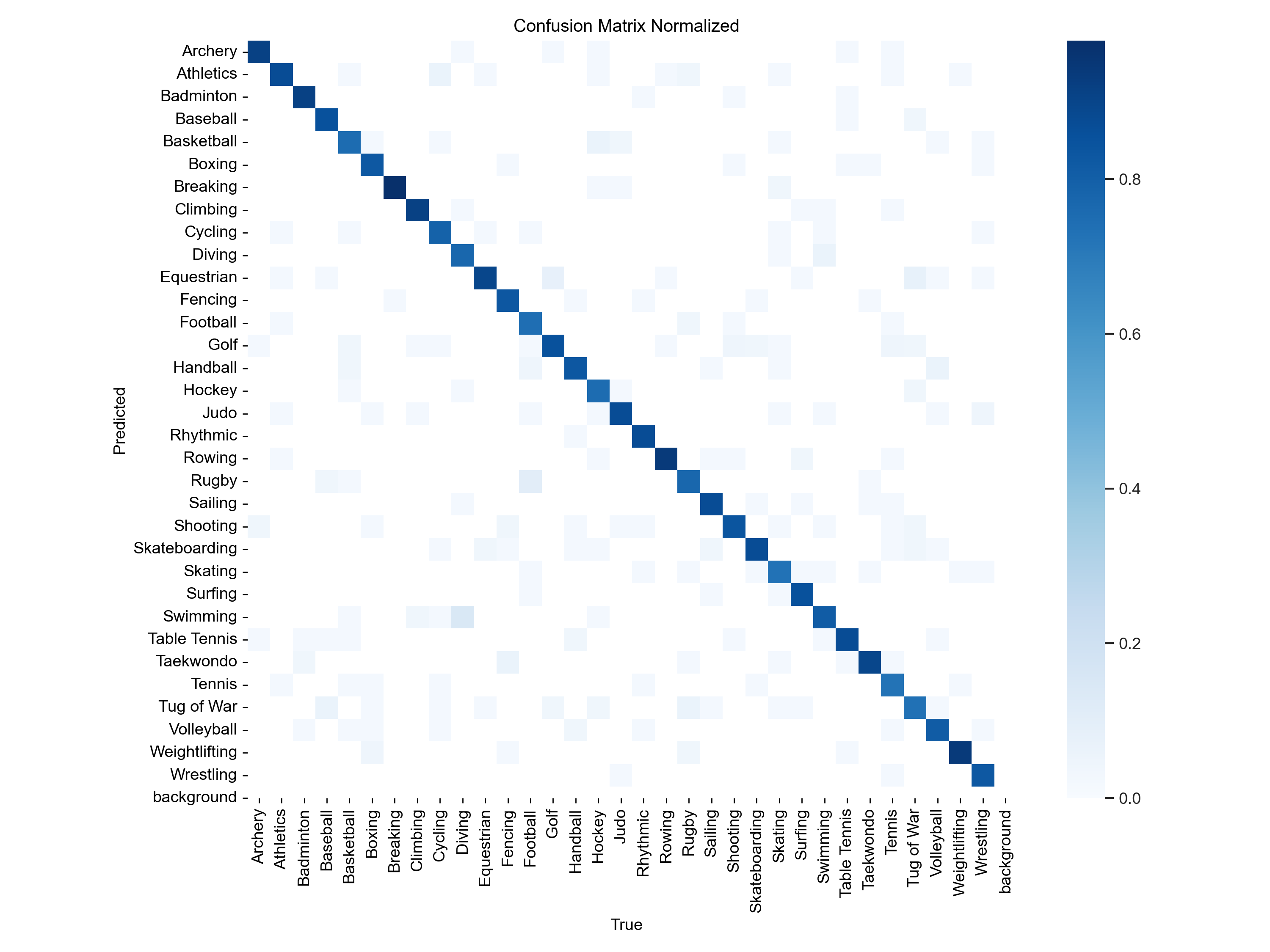
* + - 1. Database Structure
         1. MySQL:

videos table: Stores video metadata (e.g., video file path).

annotations table: Stores the detected annotations, including action labels, start and end times, and a reference to the associated video.

* + 1. Workflow:
       1. **Video Upload:**
          1. User selects video file ->
          2. Frontend sends HTTP POST request to /upload ->
          3. Flask backend receives video ->
          4. Video saved to server directory ->
          5. File path inserted into 'videos' table ->
          6. YOLO model detects actions on video frames ->
          7. Annotations created based on detected actions ->
          8. Annotations stored in 'annotations' table ->
          9. Success response to user.
       2. Search Query:
          1. User submits search query ->
          2. Frontend sends GET request to /search ->
          3. Flask backend extracts query parameter ->
          4. Backend queries 'annotations' table using SQL LIKE ->
          5. Matching results retrieved from database ->
          6. Backend sends results as JSON response ->
          7. Frontend displays search results to user.
  1. Weeks 9: Testing
     1. Conducting End-to-End Testing with Real Footage and User Queries
        1. Perform comprehensive testing on actual game footage to validate the model’s accuracy and the search engine’s response times and relevancy.
        2. Simulate a variety of user queries to assess the search engine’s ability to handle diverse input types and detect potential performance bottlenecks.
  2. Week 10: Final Refinements and Demo
     1. Resolving Remaining Issues
        1. Address any outstanding technical challenges identified during integration and testing, ensuring smooth operation across all modules.
        2. Fine-tune model parameters, database indexing, and query processing to maximize accuracy and speed.
        3. Implement a final pass of error-checking and validation to ensure robustness in both the action detection model and search engine.
     2. Preparing for Final Project Presentation
        1. Create a comprehensive presentation that showcases the project’s goals, technical implementation, and results.
        2. Prepare demo scenarios that highlight the system’s key features, such as action detection accuracy, seamless search integration, and user-friendly query responses.
        3. Develop a summary of lessons learned, challenges faced, and future improvement areas for the system.

1. Challenges Anticipated and Mitigation Strategies:
   1. Action Misclassification:
      1. The normalized confusion matrix provides a clearer view by presenting the proportion of correct and incorrect classifications as percentages. This matrix helps identify the relative accuracy per class, normalizing for class imbalance. A higher percentage on the diagonal suggests that certain actions are more accurately classified compared to others.



* + 1. Challenge: Actions like Equestrian and Equestrian Jumping may appear similar, leading to misclassification, though they’re different in Olympics.
       1. Mitigation: If actions can overlap, combine classification approach that allows the model to predict classes for a single instance.
    2. Challenge: Some just leading to misclassification.
       1. Mitigation: Threshold Optimization: Skipped the frame of misclassification and remove short classification in videos.
       2. def is\_short\_interruption(results, current\_index, current\_tag, lookahead=3):  
           for offset in range(1, lookahead + 1):  
           if current\_index + offset < len(results) and results[current\_index + offset]["sport"] == current\_tag:  
           return True  
           return False
  1. Non-related Images in the Dataset.
     1. Impact on Model Learning:
        1. In some sports datasets, there may be images unrelated to the actual sport, like posters or items with the same name. These non-related images can negatively impact the model's learning during training.
        2. Mitigation: Remove Non-related Images and train with correct data: Manually review the entire dataset to identify and delete unrelated images, replacing them with relevant images as needed.
  2. Large Frame Size and Model Adaptation:
     1. Processing large video frames with an image-based model may result in high computational cost and inefficiency.
        1. Mitigation:
           1. Downsample frames and use frame sampling to reduce computational load.
           2. Incorporate temporal models smaller to capture sequential dependencies between frames.
  3. Annotation Consistency Across Actions:
     1. Detecting and annotating action boundaries consistently, especially when actions are similar across sports subclasses.
        1. Mitigation: Use remove short misclassification frame to differentiate similar actions and validate results.

1. Future Challenges with Expectations:
   1. Optimizing Query Performance:
      1. Challenge: Complex queries could lead to slow response times with large video datasets.
      2. Mitigation:
         1. Use indexing strategies to speed up lookups and consider caching frequently queried data.
   2. Accurate Translation of User Queries:
      1. Challenge: Ambiguous or vague user queries can be difficult to interpret accurately.
      2. Mitigation:
         1. Learn NLP techniques to extract keywords and understand user.
   3. Semantic Search and Intent Matching:
      1. Challenge: Understanding user in complex sports-related queries.
      2. Mitigation:
         1. Implement a semantic search engine with embeddings (e.g., BERT, Word2Vec) to match related terms and refine results.
2. Roles of Each Student:
   1. Kuei-Yu Tsai: As the lead on data processing and training development, responsible for designing full stack systems to executing data workflows, as well as documenting the project’s progress.
   2. Yun-Hao Lee: Leads the development of the YOLOv11 learning model, focusing on coding and implementation to drive the model's functionality and performance and reporting the project.