République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

École Supérieure en Informatique -08 Mai 1945- Sidi Bel Abbès

RunRightAI : AI-Driven Sprinting Technique Assessment and Enhancement System.

Realized By:
HACHOUD Mohammed
FRIHAOUI Ayoub
BELLAGHA Ayoub

Supervised by:
Dr. KHALDI Belkacem

Presented on **June 12th, 2024** in the presence of the jury members:

Ms. : Nassima DIF Examiner 1
Ms. : Rabab BOUSMAHA Examiner 2

June 10, 2024

Abstract

This project aims to develop an advanced automated system for assessing and optimizing sprinting techniques, leveraging BlazePose's state-of-the-art real-time pose estimation technology. By capturing and analyzing key biomechanical parameters such as stride length, joint angles, and ground contact time from detailed pose data, the system will provide comprehensive and personalized feedback. This feedback will be tailored to the needs of runners, coaches, and sports researchers, facilitating targeted interventions to enhance performance and reduce injury risk.

Keywords: Data Collection ; Data Labeling; Blaze pose Media Pipe (DL); RAG (NLP); ; Translation

Contents

1 INTRODUCTION **6**
- 1.1 Context . . . 6
- 1.2 Problem Statement . . . 6
- 1.3 Proposed Solution . . . 6
- 1.4 Objectives . . . 7
- 1.5 Report Outline . . . 7
- 1.6 Summary and Conclusion . . . 8

2 Background **10**
- 2.1 Introduction . . . 10
- 2.2 Artificial Intelligence . . . 10
- 2.3 Deep Learning . . . 11
  - 2.3.1 Data Collection . . . 11
  - 2.3.2 Data Preprocessing . . . 12
  - 2.3.3 Data Augmentation . . . 12
- 2.4 Natural Language Processing (NLP) . . . 12
- 2.5 Transfer Learning . . . 12
  - 2.5.1 Definition . . . 12
  - 2.5.2 Transfer Learning: Steps for Adapting Pretrained Models to New Domains . . . 13
- 2.6 Object Detection . . . 15
  - 2.6.1 Object Detection Pretrained Models . . . 15
- 2.7 BlazePose: On-device Real-time Body Pose tracking . . . 19
  - 2.7.1 Model Architecture and Pipeline Design . . . 19
- 2.8 Retrieval-Augmented Generation . . . 21

3 State Of Art **23**
- 3.1 Comparative Analysis of Skeleton-Based Human Pose Estimation . . . 23
  - 3.1.1 Introduction . . . 23
  - 3.1.2 OpenPose . . . 23
  - 3.1.3 PoseNet . . . 24
  - 3.1.4 MoveNet . . . 24
  - 3.1.5 BlazePose . . . 26
  - 3.1.6 Results and Discussion . . . 26
  - 3.1.7 Conclusion . . . 27

4 Approach and Implementation **29**
4.1 Data Collection . . . 29
4.2 Data Labeling Augmentation . . . 30
4.3 Detection Cropping . . . 32
4.3.1 Plate Detection with YOLOv8 SAHI . . . 32
4.3.2 Coordination Transformation from Image to Real World . . . 32
4.3.3 Person Tracking and Detection . . . 33
4.4 Pose Estimation Metrics calculations . . . 34
4.4.1 Pose Estimation with Offset Adjustment and Normalization . . . 34
4.4.2 Projection and Velocity Calculation . . . 34
4.5 Report Generation using Retrieval-Augmented Generation for LLMs . . . 35
4.5.1 Architecture and Workflow . . . 35
4.6 Deployment . . . 36
4.6.1 Platform . . . 36
4.6.2 Platform Overview . . . 36
4.6.3 Technical Implementation: . . . 37
4.6.4 User Workflow . . . 37
4.7 Software Technologies . . . 39
4.7.1 Programming Language . . . 39
4.7.2 Development Environment . . . 39
4.7.3 Front-End Technologies . . . 41
4.7.4 Back-End Technologies . . . 41
4.7.5 Libraries . . . 42
4.7.6 communication Tools . . . 43
4.7.7 Other Tools . . . 43

5 Conclusion **45**

List of Figures

2.1 Similarity between biological and artificial neural networks 11
2.2 Transfer Learning 13
2.3 Transfer Learning steps 13
2.4 Object prediction and object segmentation 15
2.5 YOLO architecture 17
2.6 YOLO Release Date Timeline 18
2.7 YOLO Release Date Timeline 19
2.8 keypoint topology 20
2.9 Network architecture. 21

3.1 Overall process of OpenPose 24
3.2 Flow of PoseNet [2] 24
3.3 Flow of MoveNet [3] 25

4.1 Setup for Data Collection with Galaxy S20 5G Exynos 30
4.2 Annotation Process Using Roboflow for Labeling Plates in Collected Frames 31
4.3 homography estimation Matrix 33
4.4 Velocity Calculation 35
4.5 RAG Workflow 36
4.6 Python 39
4.7 JavaScript (JS) 39
4.8 Google Colaboratory 40
4.9 Kaggle 40
4.10 Visual Studio Code 40
4.11 ReactJS 41
4.12 Flask 41
4.13 Keras 42
4.14 Sickit-Learn 42
4.15 Numpy 43
4.16 Pandas 43
4.17 Github 43
4.18 Overleaf 44

List of Tables

3.1 Basic specifications of OpenPose, PoseNet, MoveNet, and BlazePose 26

Chapter 1

INTRODUCTION

1.1 Context

The pursuit of athletic excellence in sprinting is closely tied to understanding and optimizing the biomechanics of running. Biomechanical analysis is crucial for identifying and quantifying the subtle movements that impact an athlete's speed and efficiency. Traditional methods, such as subjective observation and limited sensor-based systems, face significant limitations including bias, high costs, and impracticality for widespread use, often lacking real-time feedback capabilities.

To address the need for more accessible and objective sprinting analysis, this research proposes a novel approach using smartphone-based human pose estimation. Leveraging advanced computer vision algorithms, this technology can extract detailed pose information from video footage. The project aims to develop an automated system utilizing smartphones and tripods to capture sprinters' movements during training sessions. Real-time pose estimation models will extract key biomechanical parameters, which will be presented through an intuitive feedback tool. This tool aims to improve technique, reduce injury risk, and enhance overall performance, making high-quality sprinting analysis more accessible and effective.

1.2 Problem Statement

Traditional sprinting technique analysis often relies on manual observation or limited sensor-based methods, resulting in subjective evaluations and incomplete data. These limitations hinder athletes' ability to receive comprehensive feedback and coaches' capacity to provide data-driven guidance. Existing automated solutions, such as motion capture systems, are prohibitively expensive and impractical for widespread use.

1.3 Proposed Solution

To address the limitations of traditional sprinting technique analysis, this project proposes an accessible and effective alternative by employing pose estimation models during sprinting. These models will be fine-tuned to accurately track relevant body landmarks throughout the entire sprinting sequence, focusing on key joint angles and segmental motions crucial for optimal technique, such as ground contact moments and stride frequency.

We aim to develop an advanced, cost-effective solution using smartphone-based video analysis combined with AI-driven motion tracking. Leveraging the high-resolution cameras and processing power available in modern smartphones, the system will capture athletes' movements and employ machine learning algorithms and computer vision techniques to provide detailed, objective analysis of sprinting form, including stride length, joint angles, and velocity.

The system will also generate comprehensive feedback reports. These reports will include visual representations of the athlete's movement, highlighting key performance metrics and areas for improvement. Coaches and athletes will receive insights into specific aspects of their sprinting technique, supported by data visualizations such as graphs and heatmaps.This comprehensive feedback will be accessible and affordable for athletes and coaches, facilitating data-driven performance improvement

1.4 Objectives

Objectivity and Accessibility

The automated system eliminates subjective bias and offers an accessible and affordable alternative to traditional analysis methods.

Real-time Feedback

Runners and sprinters can receive immediate feedback during training sessions, allowing for real-time adjustments and technique improvement.

Comprehensive Analysis

The system analyzes various bio-mechanical parameters, providing a holistic understanding of sprinting technique beyond basic observations

Performance Optimization

Personalized feedback empowers runners and coaches to focus on specific areas for improvement, leading to faster times and reduced injury risk.

Research Applications

The system can be used by researchers to conduct large-scale studies on sprinting technique and performance, advancing the understanding of this critical area.

1.5 Report Outline

Chapter 1

The context of the work, a description of the problem, and the goals of our work are all presented in Chapter 1.

Chapter 2

The fundamental concepts of sentiment analysis are covered in the second chapter. This serves as a background for our topic, which is about definitions and broad concepts.

Chapter 3

The state-of-the-art chapter, which serves as the basis for our study, gives an overview of recent developments and research in the subject. It highlights the procedures, approaches, and outcomes of pertinent papers and groups them according to themes or methodologies.

Chapter 4

In this fourth section, we will go over the various steps we took for the conception and development of our project.Along with discussing the methods we used for implementation.

Chapter 5

The last chapter sums up our work and presents some conclusions.

1.6 Summary and Conclusion

In conclusion, this chapter provided an overview of the problem we aim to address, highlighting the challenges faced in traditional sprinting technique analysis, such as subjective evaluations, incomplete data, and the high cost of existing automated solutions. We presented our motivation for developing a more accessible and cost-effective solution using smartphone-based video analysis combined with AI-driven motion tracking. The chapter also outlined the specific objectives of our project.

By understanding the problem landscape and setting clear goals, we are well-equipped to proceed with the implementation and evaluation of our solution. This foundation will guide us in developing a system that provides comprehensive, data-driven feedback to athletes and coaches, ultimately enhancing sprinting performance through detailed and objective analysis.

Chapter 2

Background

2.1 Introduction

This chapter sets the foundation for our project by exploring key definitions in the field of Artificial Intelligence (AI) and delving into specific concepts such as Deep Learning & Natural Language processing , as well as the remarkable techniques of YOLO (You Only Look Once) and Blaze Pose.

We aim to provide a comprehensive understanding of these fundamental terms and methodologies, serving as a starting point for our subsequent discussions and analyses. By elucidating these concepts, we establish a solid framework for the subsequent chapters, where we will delve into the practical application and implementation of these cutting-edge technologies.

2.2 Artificial Intelligence

Artificial intelligence (AI) is a wide-ranging branch of computer science concerned with building intelligent machines capable of performing tasks that typically require human intelligence. While AI is an interdisciplinary science with multiple approaches, advancements in machine learning and deep learning, in particular, are creating a paradigm shift in virtually every sector of the tech industry. Artificial intelligent systems can perform tasks commonly associated with human cognitive functions — such as interpreting speech, playing games, and identifying patterns. They typically learn how to do so by processing massive amounts of data and looking for designs to model in their own decision-making. In many cases, humans will supervise an AI's learning process, reinforcing good decisions and discouraging bad ones. But some AI systems are designed to learn without supervision — for instance, by playing a video game repeatedly until they eventually figure out the rules and how to win.

In our project, we aim to leverage artificial intelligence technologies to revolutionize sprinting technique analysis. Artificial intelligence, encompassing machine learning and deep learning, represents a groundbreaking approach to understanding and optimizing sprinting bio mechanics.

2.3 Deep Learning

Deep learning is a sub-field of machine learning that focuses on developing and applying artificial neural networks with multiple layers, enabling models to learn hierarchical representations of data. Inspired by the structure and function of the human brain, these neural networks consist of interconnected neurons that process information.

In deep learning, neural networks are composed of numerous layers, allowing them to automatically learn and extract complex features from raw input data. These networks recognize patterns, make predictions, and perform tasks by adjusting the weights and biases of the interconnected neurons through a process called training.

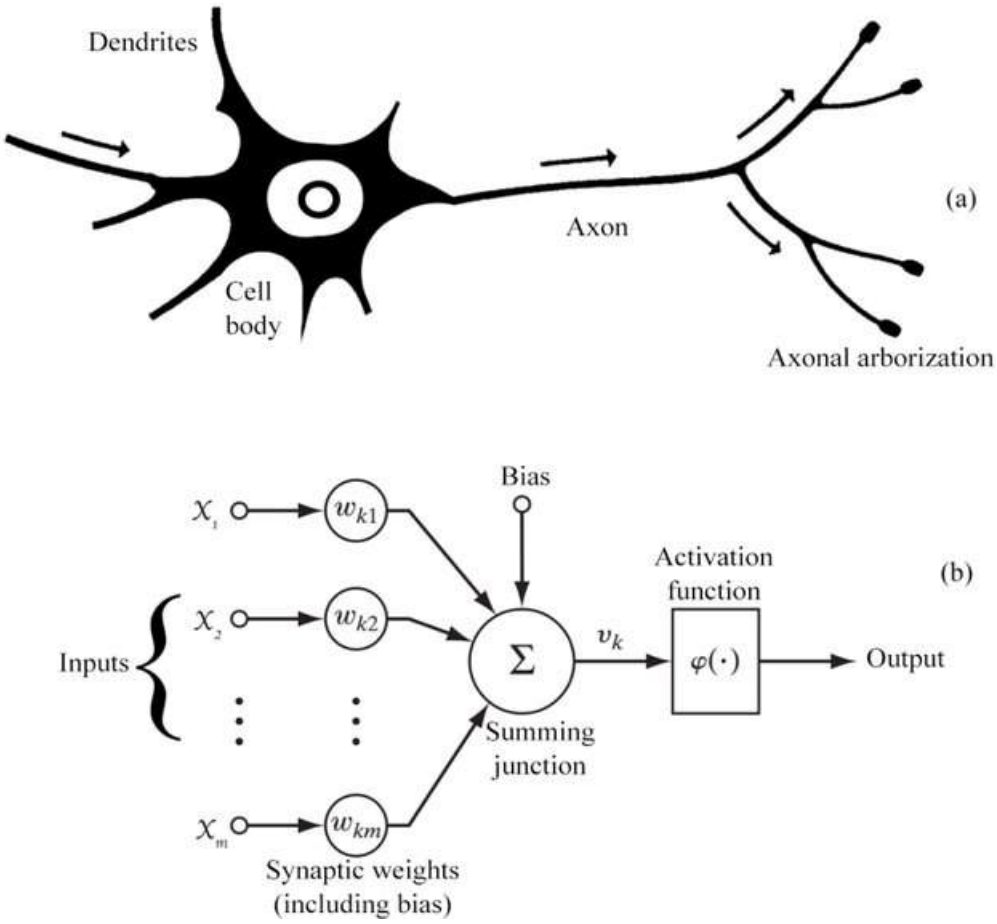


Figure 2.1: Similarity between biological and artificial neural networks

Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. This aligns with the goals of our project, which involves sprinting technique analysis through pose estimation and data extraction automatically without human involvement.

2.3.1 Data Collection

Data collection is a critical and foundational step in developing effective machine learning models, particularly in the context of sprinting technique analysis. This process involves

gathering raw data that will be used to train and evaluate the models. High-quality data collection is paramount, as it directly impacts the accuracy and reliability of the resulting models.

2.3.2 Data Preprocessing

Data preprocessing is an essential and fundamental step that enhances the quality of data and enables the extraction of valuable insights. It involves cleaning and organizing raw data to make it suitable for creating and training machine learning models. As the scale of data increases, the quality becomes increasingly important as it contributes to improving the model's ability to generalize effectively.

Deep learning models are capable of producing accurate conclusions from vast volumes of input data, even when they are not given specific data attributes to consider. Preprocessing steps may include data cleaning, normalization, and transformation to ensure that the input data is in an optimal format for model training.

2.3.3 Data Augmentation

Data augmentation is a technique used in machine learning and deep learning to artificially increase the size and diversity of a training dataset by creating modified versions of existing data samples. The goal of data augmentation is to introduce variability and enhance the performance and generalization ability of machine learning models.

In our project, data augmentation techniques may include rotating, flipping, scaling, and adding noise to the video frames of sprinting athletes. This not only increases the amount of training data but also helps the model become more robust and capable of handling various real-world scenarios. By augmenting the data, we ensure that our deep learning models can generalize well and provide accurate and reliable analysis across different conditions and athletes.

2.4 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on the interaction between computers and humans through natural language. It involves enabling machines to understand, interpret, and generate human language in a way that is both meaningful and useful. NLP encompasses a variety of tasks such as text analysis, sentiment analysis, language translation, and speech recognition.

In the context of our project, NLP plays a crucial role in enhancing the user experience and the overall functionality of our sprinting technique analysis system. This integration ensures that athletes and coaches receive comprehensive, actionable feedback in a user-friendly manner, further driving performance improvements and optimizing training outcomes.

2.5 Transfer Learning

2.5.1 Definition

Transfer learning is a machine learning technique that leverages knowledge learned from one task or domain to improve performance on a different but related task or domain. In

transfer learning, a pre-trained model, which has been trained on a large dataset for a specific task, is used as a starting point for a new task. The idea behind transfer learning is that the knowledge and representations learned by a model on one task can be valuable in solving a different task. Instead of training a model from scratch on a new task, transfer learning allows us to utilize the knowledge and features learned from previous tasks, saving computational resources and time. The process typically involves taking a pre-trained model, removing or freezing some of its layers, and adding new layers specific to the target task. These new layers are then trained on a smaller dataset specific to the new task. By starting with pre-trained weights, the model can capture general features and patterns that are transferable across tasks, thus improving performance and convergence speed on the new task.

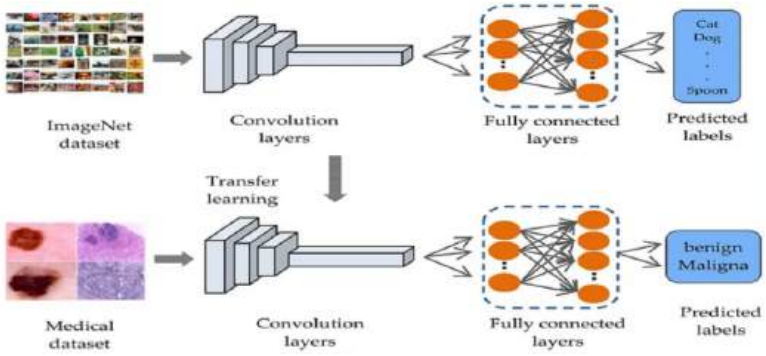


Figure 2.2: Transfer Learning

2.5.2 Transfer Learning: Steps for Adapting Pretrained Models to New Domains

This section provides an overview of the transfer learning process, outlining the sequential steps involved in adapting pre-trained models to new domains.

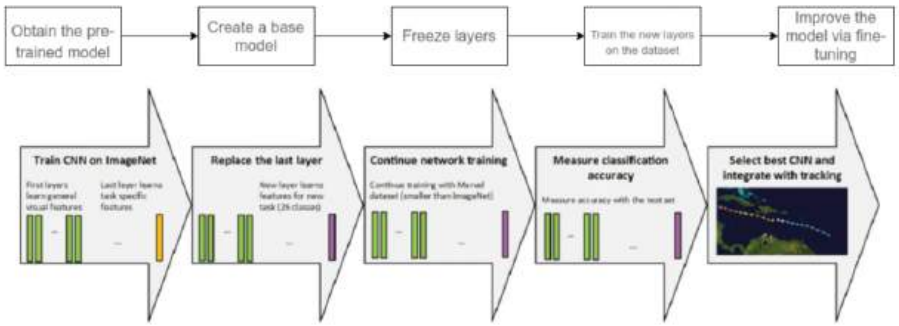


Figure 2.3: Transfer Learning steps

- **Select a Pre-Trained Model:** 1. **Select a Pre-Trained Model:**

  Choose a pre-trained model that has been trained on a large dataset for a related task. The choice of model depends on the specific problem domain and the available resources. Here are some of the pre-trained models you can use:

1. **ImageNet Models:**
   - VGG (e.g., VGG16, VGG19)
   - ResNet (e.g., ResNet50, ResNet101, ResNet152)
   - Inception (e.g., InceptionV3, InceptionResNetV2)
2. **Natural Language Processing (NLP) Models:**
   - Word2Vec
   - GloVe
   - BERT (Bidirectional Encoder Representations from Transformers)
   - Transformer-XL
3. Object Detection Models:
   - Faster R-CNN
   - YOLO (You Only Look Once)

- **Create a Base Model:** The base model is one of the architectures that we have selected in the first step to be in close relation to our task. We can either download the network weights which saves the time of additional training of the model. Else, we will have to use the network architecture to train our model from scratch. There can be a case where the base model will have more neurons in the final output layer than we require in our use case. In such scenarios, we need to remove the final output layer and change it accordingly.

- **Freeze Layers:** Freezing the starting layers from the pre-trained model is essential to avoid the additional work of making the model learn the basic features. If we do not freeze the initial layers, we will lose all the learning that has already taken place.

- **Add new Trainable Layers:**The only knowledge we are reusing from the base model is the feature extraction layers. We need to add additional layers on top of them to predict the specialized tasks of the model. These are generally the final output layers.

- **Train the New Layers:** The pre-trained model's final output will most likely differ from the output we want for our model. For example, pre-trained models trained on the ImageNet dataset will output 1000 classes. However, we need our model to work for two classes. In this case, we have to train the model with a new output layer in place.

- **Fine-tune the Model:** One method of improving the performance is fine-tuning. Fine-tuning involves unfreezing some parts of the base model and training the entire model again on the whole dataset at a very low learning rate. The low learning rate will increase the performance of the model on the new dataset while preventing overfitting.

2.6 Object Detection

Object detection is a computer vision task that involves identifying and localizing objects within an image or video. The goal of object detection is to determine the presence of objects in an image, classify them into predefined categories, and provide bounding box coordinates that indicate their locations. Object detection combines two key components: prediction and segmentation.

1. **Object Prediction:**

   Object prediction in the context of object detection refers to identifying and classifying objects present in an image or video frame. The prediction step involves analyzing the visual content of the input and assigning labels or categories to the detected objects. This process typically employs machine learning techniques, such as deep neural networks, that are trained on large datasets to learn and recognize patterns associated with different objects

2. **Object Segmentation:**

   Object segmentation, also known as instance segmentation, goes beyond object prediction by not only identifying objects but also precisely delineating their boundaries at the pixel level. Segmentation provides a more detailed understanding of object shapes and enables the separation of overlapping instances or objects of the same class within an image. It assigns a unique label or identifier to each pixel, indicating which object it belongs to.

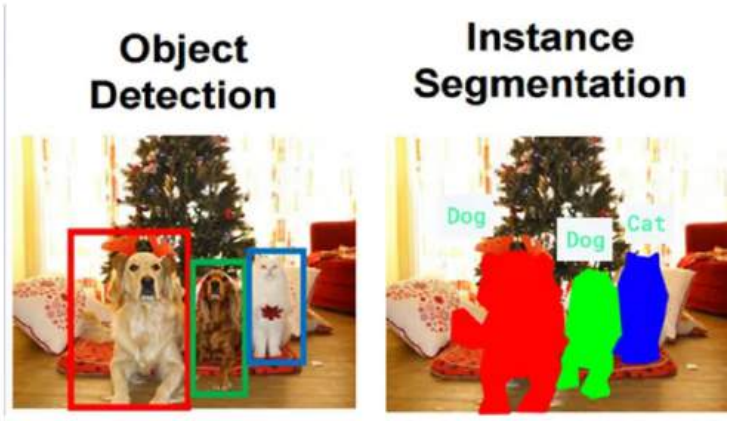


Figure 2.4: Object prediction and object segmentation

2.6.1 Object Detection Pretrained Models

(Common Objects in Context), PASCAL VOC (PASCAL Visual Object Classes), or Open 19 Images, to learn object representations and capture relevant features. They can be fine-tuned on specific datasets or tasks using **transfer learning techniques**, allowing for efficient adaptation to new domains or object categories These models leverage **convolutional neural networks (CNNs)** as their backbone to extract meaningful features from input images. They employ various techniques like anchor boxes, region proposals, multi-scale

feature maps, and specific loss functions to improve accuracy, speed, and robustness in object detection

There are several popular object detection models, each with its own architecture and characteristics. Some of the commonly used object detection models include:

1. **Region-based Convolutional Neural Networks (R-CNN):**

   (a) R-CNN: The original R-CNN introduced the concept of region proposals to identify potential object locations in an image. It extracts region proposals using selective search and then classifies each proposal using a CNN.

   (b) Fast R-CNN: Building on R-CNN, Fast R-CNN combines region proposal generation and object classification into a single network, improving speed and efficiency.

   (c) Faster R-CNN: Further enhancing Fast R-CNN, Faster R-CNN introduces a Region Proposal Network (RPN) that shares convolutional features with the object detection network, making the entire process end-to-end trainable.

2. **Single Shot MultiBox Detector (SSD):**

   SSD is a single-shot object detection model that predicts object bounding boxes and class probabilities directly from feature maps at multiple scales. It achieves real-time performance by utilizing a set of default anchor boxes with different aspect ratios and scales.

3. **You Only Look Once (YOLO):** YOLO is an object detection model that performs detection in a single pass through the network. It divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell. YOLO achieves real-time performance by considering global context and leveraging shared features across the image.

YOLO (You Only Look Once)

Unlike traditional object detection algorithms that require multiple passes over an image, YOLO performs detection in a single pass. It uses a single neural network to simultaneously predict bounding boxes and class probabilities for multiple objects in an image.

Some of the reasons why YOLO is leading the competition include its:

1. **Speed:** YOLO is extremely fast because it does not deal with complex pipelines. It can process images at 45 Frames Per Second (FPS). In addition, YOLO reaches more than twice the mean Average Precision (mAP) compared to other real-time systems, which makes it a great candidate for real-time processing.

2. **High detection accuracy:** YOLO is far beyond other state-of-the-art models in accuracy with very few background errors.

3. **Better generalization:** This is especially true for the new versions of YOLO, which will be discussed later in the article. With those advancements, YOLO pushed a little further by providing a better generalization for new domains, which makes it great for applications relying on fast and robust object detection.

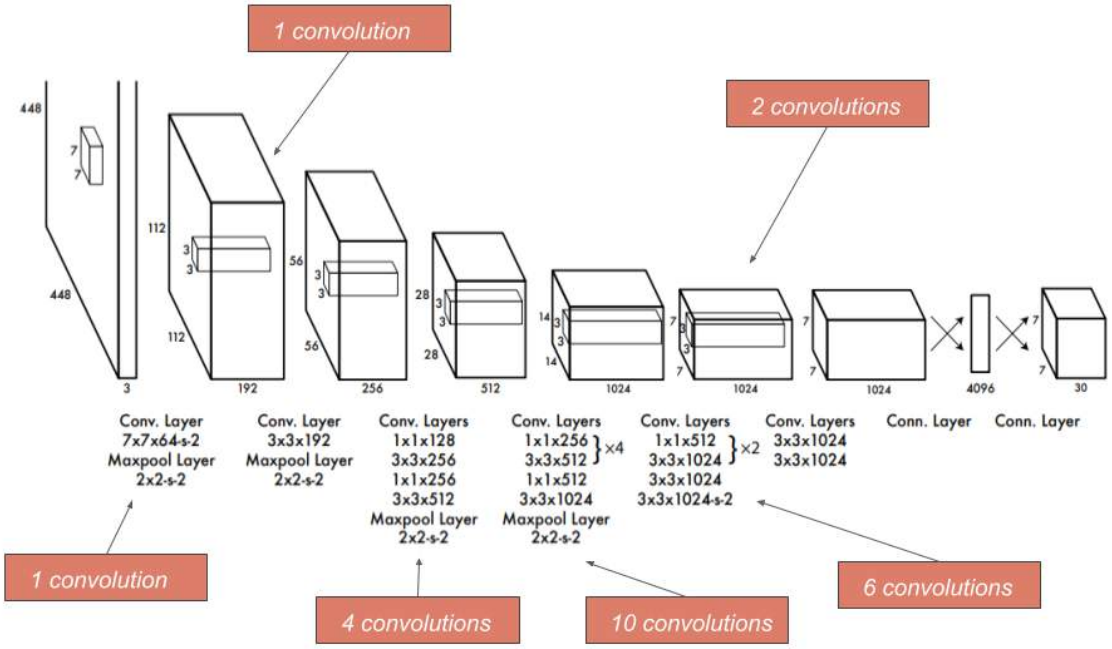


Figure 2.5: YOLO architecture

4. **Open source :** Making YOLO open-source led the community to constantly improve the model. This is one of the reasons why YOLO has made so many improvements in such a limited time.

YOLO has evolved through different versions, each with improvements in accuracy and speed:

1. **YOLOv1 (You Only Look Once v1):**
   - Introduced in 2015.
   - Pioneering version of YOLO that introduced the concept of single-stage object detection.
   - Divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell.
   - Achieved real-time object detection but suffered from localization errors and struggles with small objects.
2. **YOLOv2 (You Only Look Once v2) / YOLO9000:**
   - Released in 2016.
   - Improved the performance and accuracy of YOLOv1.
   - Incorporated anchor boxes to handle objects of different scales.
   - Introduced Darknet-19 architecture with additional convolutional layers.
   - Implemented multi-scale training and introduced the concept of "passthrough" layers.

- Trained on both the COCO dataset and ImageNet, resulting in YOLO9000's ability to detect over 9,000 object categories.

3. **YOLOv3 (You Only Look Once v3):**

   - Released in 2018.
   - Significantly improved the performance and accuracy over YOLOv2.
   - Utilized a variant of Darknet called Darknet-53 with a deeper architecture (53 layers).
   - Introduced multiple detection scales, enabling detection at different resolutions.
   - Integrated feature pyramid network (FPN) and skip connections for improved feature extraction.
   - Introduced the concept of "bounding box priors" to handle different aspect ratios of objects.
   - Achieved state-of-the-art performance in real-time object detection.

4. **YYOLOv4 (You Only Look Once v4):**

   - Released in 2020.
   - Built upon the success of YOLOv3 and aimed to further improve object detection performance.
   - Introduced various advancements, including CSPDarknet53 backbone, PANet, CIOU loss, and Mish activation function.
   - Improved network architecture for better speed and accuracy trade-offs.
   - Addressed some limitations of YOLOv3, such as handling small object detection and reducing false positives.

There are more versions of Yolo, versions 5,6,7 and 8

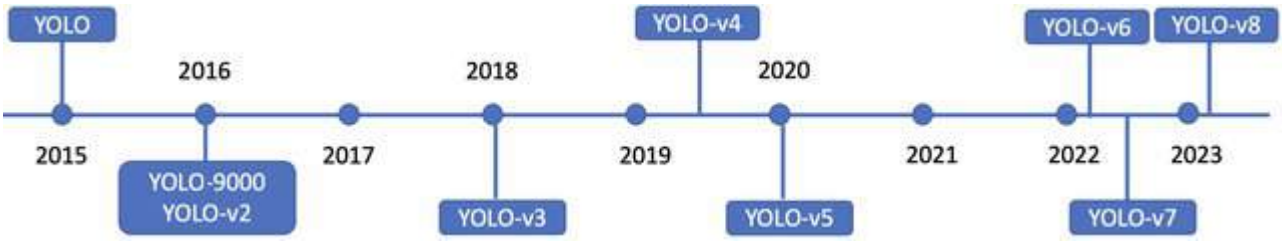


Figure 2.6: YOLO Release Date Timeline

2.7 BlazePose: On-device Real-time Body Pose tracking

Human body pose estimation from images or video plays a central role in various applications such as health tracking, sign language recognition, and gestural control. This task is challenging due to a wide variety of poses, numerous degrees of freedom, and occlusions. Recent work [10][7] has shown significant progress on pose estimation. The common approach is to produce heatmaps for each joint along with refining offsets for each coordinate. While this choice of heatmaps scales to multiple people with minimal overhead, it makes the model for a single person considerably larger than is suitable for real-time inference on mobile phones. In this paper, we address this particular use case and demonstrate significant speedup of the model with little to no quality degradation. In contrast to heatmap-based techniques, regressionbased approaches, while less computationally demanding and more scalable, attempt to predict the mean coordinate values, often failing to address the underlying ambiguity. Newell et al. [9] have shown that the stacked hourglass architecture gives a significant boost to the quality of the prediction, even with a smaller number of parameters. We extend this idea in our work and use an encoder-decoder network architecture to predict heatmaps for all joints, followed by another encoder that regresses directly to the coordinates of all joints. The key insight behind our work is that the heatmap branch can be discarded during inference, making it sufficiently lightweight to run on a mobile phone.

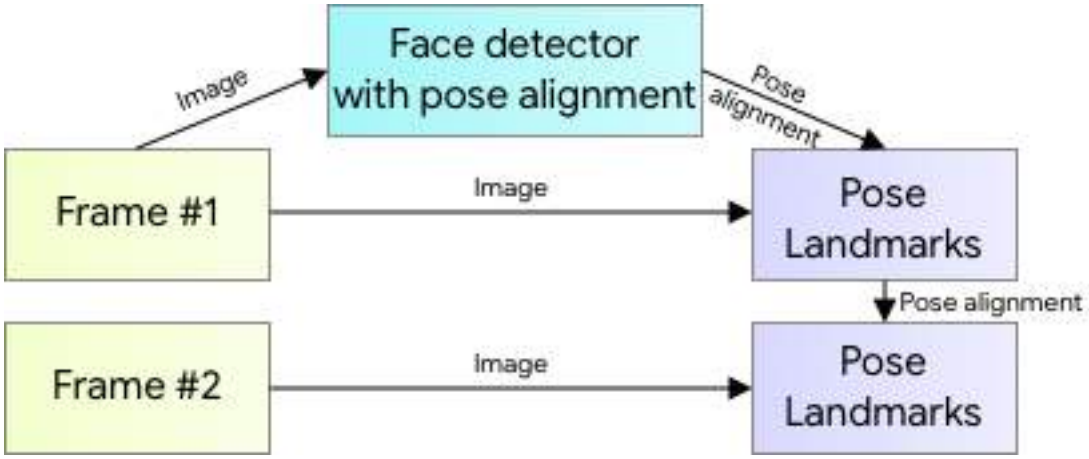


Figure 2.7: YOLO Release Date Timeline

2.7.1 Model Architecture and Pipeline Design

Inference pipeline

During inference, we employ a detector-tracker setup (see Figure 1), which shows excellent real-time performance on a variety of tasks such as hand landmark prediction [3] and dense face landmark prediction [6]. Our pipeline consists of a lightweight body pose detector followed by a pose tracker network. The tracker predicts keypoint coordinates, the presence of the person on the current frame, and the refined region of interest for the current frame. When the tracker indicates that there is no human present, we re-run the detector network on the next frame.

Person detector

The majority of modern object detection solutions rely on the Non-Maximum Suppression (NMS) algorithm for their last post-processing step. This works well for rigid objects with few degrees of freedom. However, this algorithm breaks down for scenarios that include highly articulated poses like those of humans, e.g. people waving or hugging. This is because multiple, ambiguous boxes satisfy the intersection over union (IoU) threshold for the NMS algorithm.

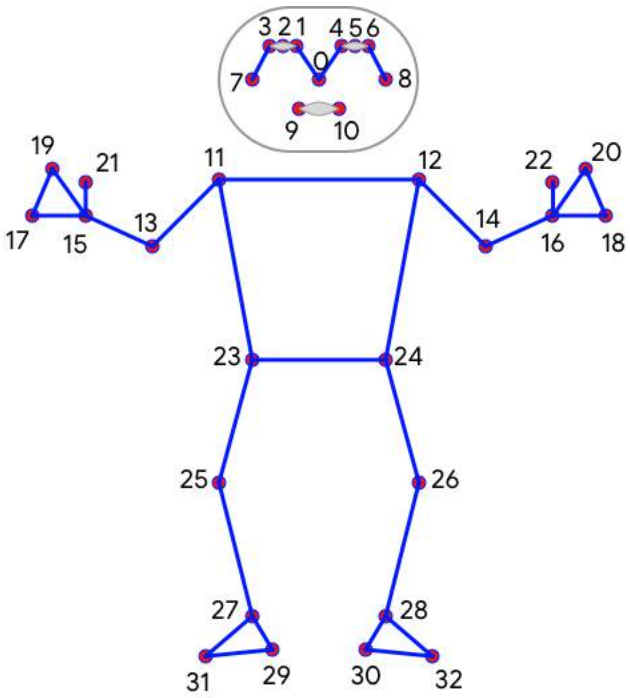


Figure 2.8: keypoint topology

To overcome this limitation, we focus on detecting the bounding box of a relatively rigid body part like the human face or torso. We observed that in many cases, the strongest signal to the neural network about the position of the torso is the person's face (as it has high-contrast features and has fewer variations in appearance). To make such a person detector fast and lightweight, we make the strong, yet for AR applications valid, assumption that the head of the person should always be visible for our single-person use case. As a consequence, we use a fast on-device face detector [2] as a proxy for a person detector. This face detetor predicts additional personspecific alignment parameters: the middle point between the person's hips, the size of the circle circumscribing the whole person, and incline (the angle between the lines connecting the two mid-shoulder and mid-hip points).

Neural network architecture

The pose estimation component of our system predicts the location of **all 33 person keypoints**, and uses the person alignment proposal provided by the first stage of the pipeline We adopt a combined heatmap, offset, and regression approach, as shown in Figure Below. We use the heatmap and offset loss only in the training stage and remove the corresponding

output layers from the model before running the inference. Thus, we effectively use the heatmap to supervise the lightweight embedding, which is then utilized by the regression encoder network. This approach is partially inspired by Stacked Hourglass approach of Newell, but in ourcase, we stack a tiny encoder-decoder heatmap-based network and a subsequent regression encoder network. We actively utilize skip-connections between all the stages of the network to achieve a balance between highand low-level features. However, the gradients from the regression encoder are not propagated back to the heatmaptrained features (note the gradient-stopping connections in Figure Below). We have found this to not only improve the heatmap predictions, but also substantially increase the coordinate regression accuracy

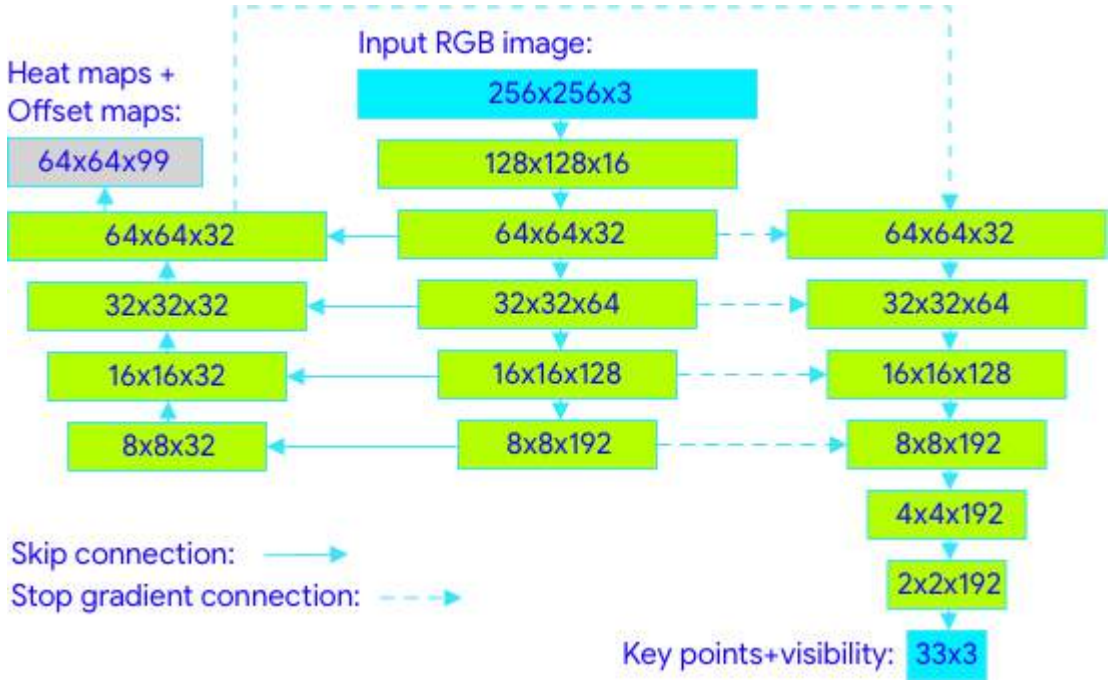


Figure 2.9: Network architecture.

2.8 Retrieval-Augmented Generation

The world of natural language processing (NLP) has witnessed tremendous advancements, notably in areas like conversational AI, language translation, and text summarization. One particularly promising development is Retrieval-Augmented Generation (RAG), which merges the strengths of retrieval-based and generation-based models to produce high-quality, coherent, and relevant text. The RAG process involves generating initial text from a prompt, retrieving related information from large external datasets, and using this information to refine the text. This approach offers numerous benefits, including improved quality, enhanced coherence, and task-specific flexibility. RAG's potential applications are vast, ranging from summarization and question answering to text classification. As research progresses, RAG is poised to significantly advance language understanding and generation capabilities.

Chapter 3

State Of Art

3.1 Comparative Analysis of Skeleton-Based Human Pose Estimation

3.1.1 Introduction

Prior to beginning the actual work on this project, we read various research papers about pose estimation and its applications. These papers provided us with a comprehensive understanding of the current advancements and methodologies in human pose estimation. Specifically, we explored several state-of-the-art models, including OpenPose, PoseNet, MoveNet, and BlazePose. Each of these models has unique features, advantages, and limitations, making them suitable for different applications and environments.

In this Chapter, we aim to compare these four human pose estimation architectures—OpenPose, PoseNet, MoveNet, and BlazePose—to determine which one is the most suitable for our specific requirements. We will evaluate these models based on their detection parts, number of key-points, detection methods, real-time performance, and platform compatibility. By conducting a thorough comparison, we will identify the model that best meets our needs, ensuring the effectiveness and efficiency of our pose estimation implementation.

3.1.2 OpenPose

In 2017, Cao et al. **10** at Carnegie Mellon University proposed OpenPose. This open source model marks the first attempt to estimate multi-person poses in the real time. It uses the bottom-up method to overcome the limitations of the top-down method. The flow of OpenPose is illustrated in Figure Below.

As shown in the figure below, two-dimensional (2D) images/videos are imported to the model. Then, a confidence map is obtained from the input. Through non-maximum suppression (NMS), the candidates for the body are identified in the confidence map. After identifying an object, a bounding box is created to contain the object, and the probability of the object is set as a score. Next, the scores are sorted in descending order, and the redundant bounding boxes are removed by the following criterion: the intersection of union (IoU) is greater than the threshold, which means the two bounding boxes identify the same object. The IoU refers to the ratio of the overlapping region to the combined region. The removal of redundant bounding boxes is known as NMS. On this basis, OpenPose creates

part affinity fields (PAFs), a set of flow fields representing the relationships between parts of many persons. Finally, bipartite matching is performed on the candidates using confidence maps and PAFs, producing full-body poses.

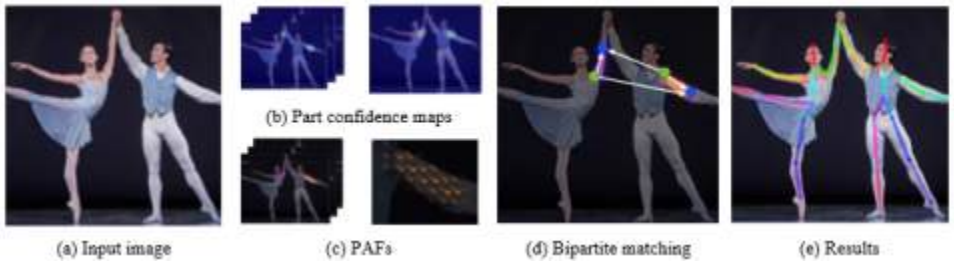


Figure 3.1: Overall process of OpenPose

3.1.3 PoseNet

PoseNet is an open source machine learning model created by Google Creative Lab. Capable of estimating human poses in the real time [1], the model works on the recently released COCO person key-point detection dataset, which tracks the key-points of the entire body.

Single-person pose estimation uses four elements as input: an input image, an image scale factor, a horizontal flip, and an output stride. The single-person detection algorithm is faster and simpler than the multi-person detection algorithm. Besides the four elements above, three other elements are required in the input of multi-person pose estimation: the maximum number of detected poses, the threshold for pose confidence score, and the NMS radius. These additional elements improve the accuracy of multi-person pose estimation. The estimation results include pose confidence scores and key-points.

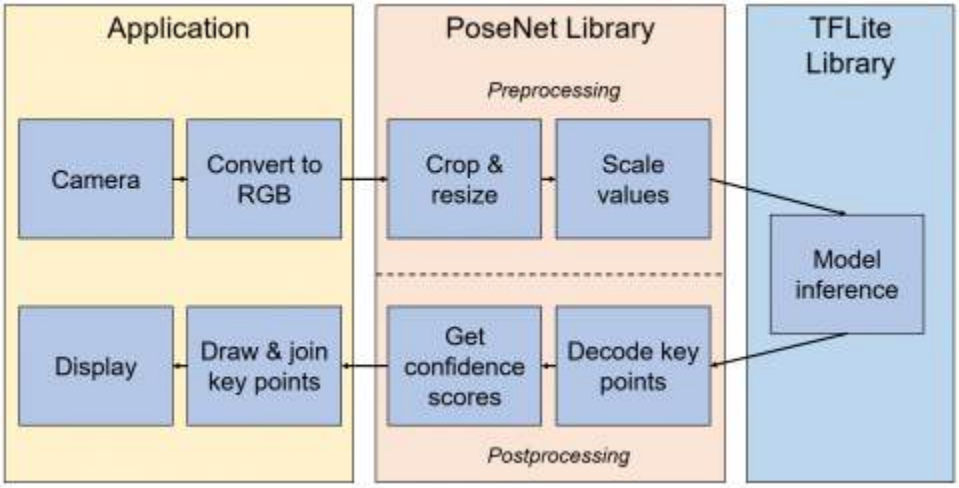


Figure 3.2: Flow of PoseNet [2]

3.1.4 MoveNet

MoveNet [3], [4] is a Google-based inference model developed by IncludeHealth, a digital health company [5]. IncludeHealth unveiled the model in 2021, and solicited help from

Google to support remote treatment of patients. Similar to PoseNet, the web version of MoveNet uses TensorFlow.js, and the mobile version uses TensorFlow Lite. There are two versions of MoveNet: the performance-oriented Lightning [6], and the accuracy-oriented Thunder [7]. The two models differ in input size and depth multiplier. In terms of input, Lightning receives a video or an image of a fixed size (192×192) and three channels, and employs 1.0 depth multiplier. In contrast, Thunder receives an input of the size 256×256 and three channels, and employs 1.75 depth multiplier. The depth multiplier changes the number of channels of the input video/image, which generally adopts the red-green-blue (RGB) format. Yet feature maps can also be regarded as one channel in each layer. Meanwhile, Thunder has 1.75 times more layers for deep learning than Lightning. Hence, it performs 1.75 times more calculations.

MoveNet is a bottom-up model relies on TensorFlow object detection API and MobileNet V2 as a feature extractor. In the TensorFlow object detection API, there are multiple detection models supporting TensorFlow 1 and TensorFlow 2. MoveNet follows CenterNet **18**, [8], a detection API [9]. Different from the standard anchor (bounding box)-based detection model, CenterNet takes the center point as the only anchor, and searches for and classifies objects by processing regional proposals, rather than inferring objects from the IoU value. Without requiring NMS, CenterNet recognizes the difference between objects in one stage, and exhibits a high performance. As shown in the figure, MoveNet calculates

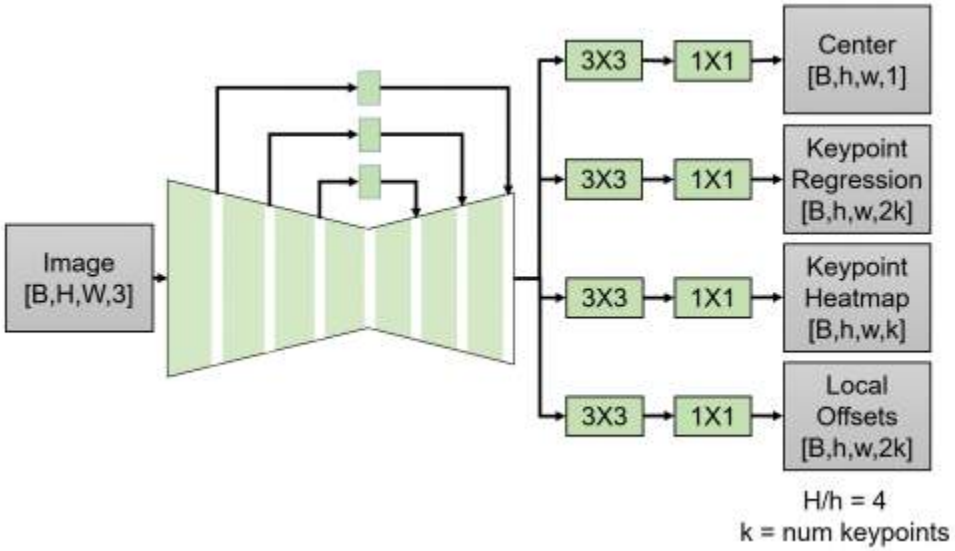


Figure 3.3: Flow of MoveNet [3]

all four processes simultaneously. Once a person center heatmap is prepared to identify each person, the location with the highest score is selected. Then, a set of key-points for the person is initialized based on the key-points obtained through regression. A person is identified, when the regression fits the arrangement of prepared key-points into a person. Furthermore, each pixel is multiplied by a weight, which is inversely proportional to the distance from the regressed key-point. In this way, the key-points from the persons in the background are excluded from the computation. In the end, the set of key-points is finalized according to the maximum heatmap values in each key-point channel.

3.1.5 BlazePose

BlazePose is a high-fidelity pose estimation model developed by Google, widely used in applications such as Google Fit and YouTube Stories. This model is optimized for mobile devices, ensuring real-time performance, which is essential for applications requiring detailed pose information. BlazePose employs a top-down approach and is implemented using MediaPipe. It tracks 33 key-points on the body, providing comprehensive pose information, particularly beneficial for fitness and physical activity applications. Designed to run efficiently on mobile devices, BlazePose supports multi-person detection within the input frame, making it comparable to other advanced pose estimation models. Additionally, it boasts cross-platform compatibility, supporting various platforms including Android, iOS, and web applications via TensorFlow Lite and MediaPipe.

3.1.6 Results and Discussion

The table below compares the basic specifications of OpenPose, PoseNet, MoveNet, and BlazePose. It can be observed that the number of key-points is the most prominent difference between these models. PoseNet provides a total of 17 key-points: 5 in the face and 12 in the body. MoveNet has the same key-points as PoseNet. OpenPose supports 137 key-points: 25 in the body, including the foot, 21 in each hand, and 70 in the face. BlazePose tracks 33 key-points, offering a balance between the simplicity of PoseNet/MoveNet and the detailed tracking of OpenPose. Overall, OpenPose can track the body in greater detail than the other three models. Another difference lies in the pose estimation method: PoseNet and BlazePose use the top-down method, while OpenPose and MoveNet adopt the bottom-up method. The top-down method estimates a pose within a person's bounding box after detecting the person. On the contrary, the bottom-up method predicts the key-points of persons in the input and estimates the pose from the correlation between the key-points.

Feature/Model	**OpenPose**	**PoseNet**	**MoveNet**	**BlazePose**
Detection Parts	Body, Foot, Hand, Face	Body, Part of Face	Body, Part of Face	Body, Part of Face
Detection No.	Many	One	One	Many
Key-Points	137	17	17	33
Operation	Real-time	Real-time	Real-time	Real-time
Method	Bottom-up	Top-down	Bottom-up	Top-down
Platforms	Various (CPU, Unity)	TensorFlow Lite (Android, iOS)	TensorFlow Lite (Android, iOS)	MediaPipe (Android, iOS, Web)

Table 3.1: Basic specifications of OpenPose, PoseNet, MoveNet, and BlazePose

BlazePose, with its 33 key-points and real-time performance, stands out as a robust option for detailed yet efficient pose estimation. It bridges the gap between the high-detail tracking of OpenPose and the simpler, faster models like PoseNet and MoveNet. Each model has its strengths and is suited to different applications based on the required detail, computational efficiency, and platform compatibility.

3.1.7 Conclusion

Throughout this chapter, we have presented a comprehensive overview of BlazePose, a high-fidelity pose estimation model developed by Google. By analyzing BlazePose and its features, we have gained valuable insights into the advancements achieved in pose estimation techniques and their practical implications across various domains. BlazePose's real-time performance, detailed key-point tracking, multi-person support, and cross-platform compatibility make it a versatile and powerful tool for applications such as fitness tracking, augmented reality, and other interactive experiences. This discussion highlights BlazePose's significant contributions to the field of pose estimation and its potential to enhance user interfaces and experiences in diverse applications.

Chapter 4

Approach and Implementation

4.1 Data Collection

For our project, we collected data ourselves using a Samsung Galaxy S20 5G Exynos. This device was chosen for its advanced camera capabilities and processing power, which are essential for capturing high-quality video data for pose estimation. We recorded the videos in 4K resolution at 60 frames per second (fps) to ensure that the data was detailed and accurate, providing a robust foundation for our pose estimation models.

The high resolution and frame rate of the Galaxy S20 5G Exynos allowed us to capture intricate movements and subtle pose variations, which are crucial for the precision of pose estimation. By utilizing this setup, we were able to generate a comprehensive dataset that includes a wide range of poses and activities, ensuring the versatility and robustness of our models. The data collection process involved recording multiple individuals performing various actions in different environments, providing diverse and realistic scenarios for training and testing our pose estimation models.

Figure 4.1: Setup for Data Collection with Galaxy S20 5G Exynos

4.2 Data Labeling Augmentation

After collecting the video data using the Galaxy S20 5G Exynos, the next crucial step in our approach was data labeling. Accurate labeling of the data is essential for training robust pose estimation models. In our case, we used Roboflow, a powerful tool for data annotation and preprocessing, to label the plates and other relevant objects in the frames extracted from our video recordings.

Roboflow offers an intuitive interface and a range of features that streamlined our data labeling process:

1. **Frame Extraction** We selected key frames from the recorded 4K 60fps videos to ensure a diverse and representative dataset. Extracting frames at regular intervals helped in capturing a wide variety of poses and activities.

2. **Annotation** Using Roboflow, we annotated the plates and any other significant objects in each frame. The tool's user-friendly annotation interface allowed us to accurately label the locations and boundaries of the objects with minimal effort.

3. **Dataset Management** Roboflow enabled us to organize our annotated data efficiently. We could easily manage different versions of the dataset, track changes, and ensure that the labeling was consistent across all frames.

4. **Data Augmentation** To enhance the robustness of our model, we leveraged Roboflow's data augmentation features. By applying transformations such as rotation, scaling, and flipping, we created a more diverse training dataset, which helped improve the model's ability to generalize to different poses and conditions.

5. **Exporting Data** Once the labeling and augmentation processes were complete, Roboflow allowed us to export the annotated dataset in formats compatible with various machine learning frameworks, such as TensorFlow and PyTorch. This facilitated seamless integration with our pose estimation model training pipeline.

By utilizing Roboflow for data labeling, we ensured that our dataset was accurately and efficiently annotated, providing a strong foundation for training high-performance pose estimation models. This meticulous approach to data preparation significantly contributed to the effectiveness and accuracy of our models in recognizing and interpreting human poses from the video data.

In total, we generated 216 labeled images from the selected frames of our video recordings. These labeled images included detailed annotations of the plates and other relevant objects, capturing a wide range of poses and activities. This comprehensive and well-annotated dataset was pivotal in training our pose estimation models to achieve high accuracy and reliability in real-world applications.

Figure 4.2: Annotation Process Using Roboflow for Labeling Plates in Collected Frames

4.3 Detection Cropping

4.3.1 Plate Detection with YOLOv8 SAHI

With the labeled dataset ready, the next step in our process was to train a model for plate detection using YOLOv8 and SAHI. This step was crucial for accurately identifying and localizing plates within each frame, which is essential for precise pose estimation and subsequent analysis.

1. **Training with YOLOv8:** YOLOv8 (You Only Look Once version 8) is a state-of-the-art object detection framework known for its speed and accuracy. We used the YOLOv8 framework to train our model on the labeled dataset. This involved configuring the model parameters, defining the architecture, and setting up the training process. The model was trained to detect and return the coordinates of each plate in the images.

2. **Slicing with SAHI:** SAHI (Slicing Aided Hyper Inference) is a technique that enhances object detection performance by slicing the images into smaller sections and performing inference on these slices. This is particularly useful for detecting small objects or objects that are densely packed. After training the YOLOv8 model, we used SAHI to improve the detection accuracy by processing the images in smaller sections and aggregating the results to get precise coordinates of each detected plate.

3. **Returning Coordinates:** The combined use of YOLOv8 and SAHI allowed us to accurately detect plates in the images and return the coordinates of each bounding box. These coordinates are essential for further processing and analysis in our pose estimation models.

Assigning Positions to Plates: To further enhance the utility of the detected plate coordinates, we assigned a predefined position to each plate. This assignment helps in categorizing the plates based on their locations within the image, facilitating more structured analysis. The predefined positions are as follows: **Bottom-right**, **Top-right**, **Midpoint**, **Top-left**, **Bottom-left**

By leveraging the advanced capabilities of YOLOv8 and SAHI, we were able to train a highly accurate plate detection model. This model serves as a critical component in our overall approach, enabling precise localization of plates and enhancing the performance of our pose estimation system.

4.3.2 Coordination Transformation from Image to Real World

To accurately map the detected and tracked persons from image coordinates to real-world coordinates, we employ the homography method with projection on a 2D plane. This transformation is essential for precise analysis and pose estimation, as it allows us to understand the spatial relationships and movements of individuals in a real-world context.

Homography Estimation

A homography matrix (H) is a 3x3 matrix that maps points on a plane from world coordinates to corresponding image coordinates. This matrix allows us to represent the transformation between two planes, assuming that all the points we are mapping are coplanar. therefore,

we can represent the world **coordinates with 2D vectors**. The homography matrix H is a 3*3 matrix that contains the parameters h11, h12, h13, h21, ... , h33. If we manage to figure out the values of h11 through h33 somehow, we can use the following formula to map the world coordinates to image coordinates

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$$

Figure 4.3: homography estimation Matrix

In summary, homography estimation using OpenCV provides a robust method for mapping points between world and image coordinates, enhancing the accuracy of our plate detection and pose estimation models.

4.3.3 Person Tracking and Detection

In our approach to identifying and tracking individuals who are running, we employ the YOLO (You Only Look Once) model for initial person detection, followed by a custom algorithm for tracking and velocity analysis. Here is a detailed breakdown of our method:

1. **Person Detection with YOLO:** We leverage the powerful YOLO algorithm to detect persons in each frame of the video. YOLO's fast and accurate object detection capabilities allow us to identify multiple individuals in real-time.

2. **Cropping Detected Individuals:** For each detected person, we crop the corresponding region from the frame. This step isolates each individual, enabling focused analysis on their movements without interference from the surroundings.

3. **Tracking Across Frames:** We implement a custom tracking algorithm to follow each detected person across consecutive video frames. This algorithm assigns unique identifiers to each individual, ensuring consistent tracking even in dynamic scenes.

4. **Velocity Analysis:**

   - **Calculate Movement Speed:** By analyzing the position of each tracked individual across frames, we calculate their velocity. This involves measuring the displacement of the person over time and converting it into speed.
   - **Running Detection** We define a threshold velocity to differentiate running from walking or standing. Individuals whose velocity exceeds this threshold are classified as running.

By combining YOLO for robust person detection with our custom tracking and velocity analysis algorithm, we achieve accurate identification of individuals who are running in the video. This approach is effective for various applications, including surveillance, sports analysis, and crowd monitoring.

4.4 Pose Estimation Metrics calculations

4.4.1 Pose Estimation with Offset Adjustment and Normalization

Frame Cropping

In our pose estimation pipeline, each frame is cropped to focus on the region of interest (ROI) containing the subject. The coordinates of the ROI are recorded as offsets, which are then used to adjust the pose estimation results to match the original full-frame coordinates.

Offset Addition

After the frame cropping , we added Offsets to the cropped frame coordinates to adjust for the original position in the full frame. This step is essential to accurately map the joint coordinates back to their correct locations in the context of the entire frame.

Pose Estimation

The cropped frames are fed into the Blazepose estimation model which is usedd to analyze the cropped frames and identify the coordinates of the joints. The model returns normalized (0 to 1) coordinates of the joints within the cropped frame.

Coordinate Conversion:

The normalized coordinates from the cropped frame are converted back to full-frame coordinates using the recorded offsets. This conversion is crucial to ensure that the joint coordinates are accurately mapped onto the original full-frame scale, maintaining their true positions within the entire frame.

This conversion process ensures that the joint coordinates are not only accurate within the cropped frame but also correctly positioned within the full-frame context. This accuracy is vital for applications where the spatial relationship of joints to other elements in the frame is important. By preserving this spatial accuracy, we can provide reliable and precise pose estimation results that are useful for various downstream tasks.

4.4.2 Projection and Velocity Calculation

After converting the normalized coordinates back to full-frame coordinates, we further process the joint positions to analyze motion. Specifically, we project the hip joint coordinates onto a 2D segment plane using a homography matrix. This matrix, obtained earlier, allows us to transform the 3D coordinates to a 2D plane for velocity calculation.

Velocity Calculation

With the new projected coordinates **(xproj, yproj)**, we can calculate the velocity of the hip joint. Velocity is determined by the change in position over time. Given the projected coordinates from consecutive frames, the velocity calculation is as follows:

- **Input Coordinates**: Projected coordinates of the hip joint (x_{proj}^t, y_{proj}^t) at time t and $(x_{proj}^{t+1}, y_{proj}^{t+1})$ at time $t+1$.
- **Time Interval**: Δt, the time difference between consecutive frames.
- **Velocity Calculation**:

$$v_x = \frac{x_{proj}^{t+1} - x_{proj}^t}{\Delta t}$$

$$v_y = \frac{y_{proj}^{t+1} - y_{proj}^t}{\Delta t}$$

  - The overall velocity v can be computed as:

$$v = \sqrt{v_x^2 + v_y^2}$$

This approach allows us to accurately track and analyze the motion of the hip joint, providing valuable insights into the subject's movement dynamics. By leveraging the homography matrix for projection and calculating the velocity from the projected coordinates, we can achieve precise motion analysis in the context of the 2D segment plane.

Figure 4.4: Velocity Calculation

4.5 Report Generation using Retrieval-Augmented Generation for LLMs

4.5.1 Architecture and Workflow

- **Retriever Module**: Retrieves relevant chunks of text from a PDF document to provide context for the generative model Using the **Chroma** vector store with **FastEmbedEmbeddings** for efficient retrieval.
- **Generator Module**: Generates detailed sprint performance analysis reports based on the retrieved information using **ChatGoogleGenerativeAI** with a prompt template tailored for sprint performance analysis.
- **Integration**: The retriever fetches relevant text chunks, which are then passed to the generator along with a structured prompt. The generative model uses this context to produce a detailed report.
- **Training and Fine-Tuning**:The generative model (**ChatGoogleGenerativeAI**) is pre-trained and fine-tuned on relevant data to generate accurate and contextually appropriate responses.

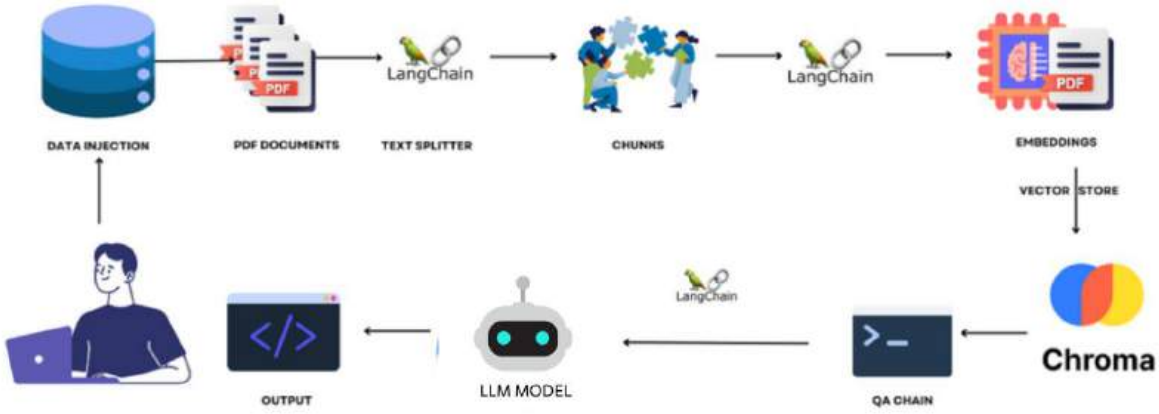


Figure 4.5: RAG Workflow

4.6 Deployment

4.6.1 Platform

we are developing a web application to analyze and report on the performance of sprinters. The main objective is to provide detailed metrics and insights based on video footage of sprinting sessions. By leveraging modern web technologies and machine learning algorithms, the platform processes video data to extract key performance metrics and generate comprehensive reports.

4.6.2 Platform Overview

Video Upload and Processing

Users can upload videos of sprinting sessions. The platform processes these videos to extract important metrics such as stride length, stride frequency, acceleration, velocity, and touchdown time.

Data Visualization

The extracted metrics are visualized using charts and graphs, allowing users to easily understand and interpret the data. This includes the velocity over time and a moving average to smooth out the data for better insights.

Performance Metrics Calculation

The platform calculates key performance metrics from the video data. This includes calculating the mean velocity of the sprinter during the session.

Sprint Report Generation

Based on the processed data, the platform generates a detailed sprint report. This report includes all relevant metrics and visualizations to provide a comprehensive overview of the sprinter's performance.

4.6.3 Technical Implementation:

Frontend

The frontend is built using React, a popular JavaScript library for building user interfaces. It includes components for video upload, data visualization (using Chart.js), and displaying the generated report.

Backend

The backend is implemented using Node.js with the Express framework. It handles video processing requests, communicates with the machine learning model for extracting metrics, and manages the data flow between the frontend and backend.

Machine Learning

A machine learning model processes the uploaded videos to extract performance metrics. This model analyzes the video frames to identify key actions and measure the relevant metrics.

API Integration

The platform integrates with a backend API to send video data for processing and to retrieve the generated metrics and reports.

4.6.4 User Workflow

Upload Video

The user uploads a video of a sprinting session through the platform's interface.

Video Processing

The platform processes the video to extract performance metrics using a machine learning model.

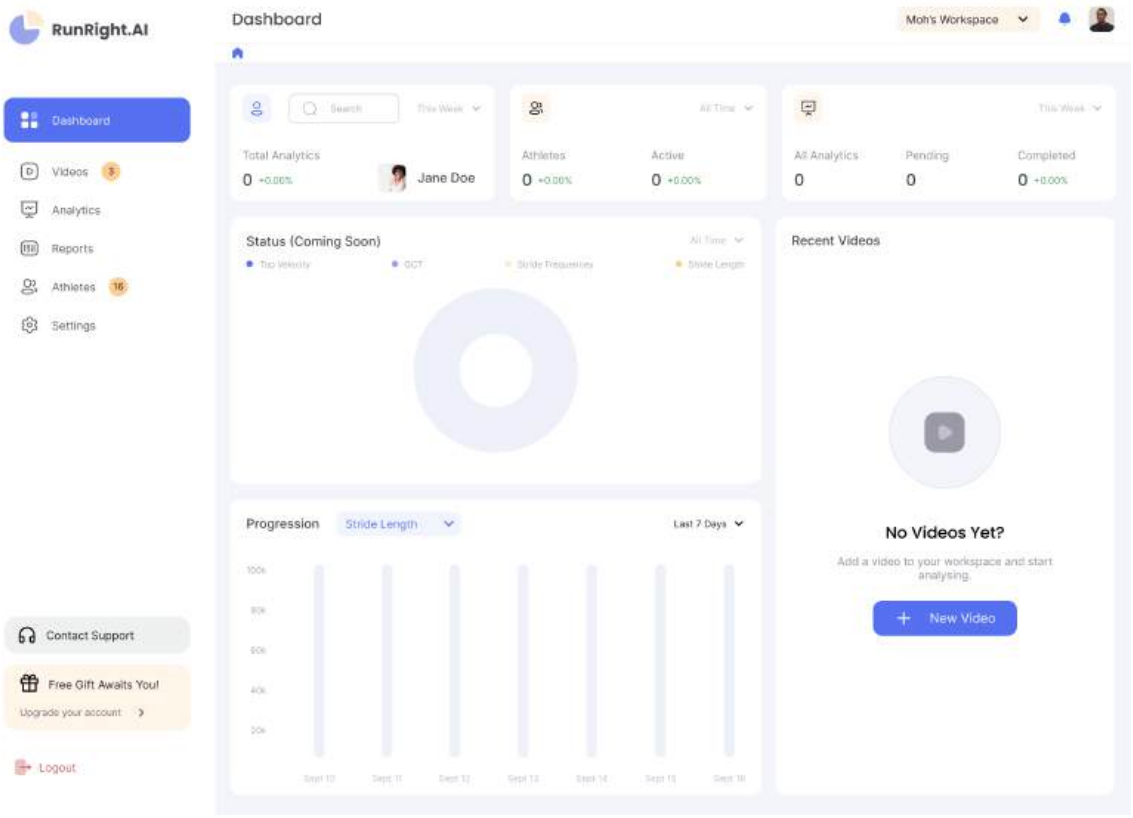
Data Visualization

The extracted metrics are visualized in real-time on the platform.

Generate Report

The user can generate a comprehensive sprint report based on the processed data. This report includes all relevant metrics and visualizations, providing detailed insights into the sprinter's performance.

This platform aims to provide athletes and coaches with valuable insights into sprint performance, helping them to identify strengths, areas for improvement, and track progress over time. By leveraging advanced technology, the platform offers a powerful tool for enhancing athletic training and performance analysis.



4.7 Software Technologies

4.7.1 Programming Language

Python

Python is a popular computer programming language used to create software and websites, automate processes, and analyze data. Python is a general-purpose language, which means it may be used to make many various types of applications and isn't tailored for any particular issues. It has become one of the most popular programming languages in use today due to its versatility and beginner-friendliness. It was the second-most popular programming language among developers in 2021, according to a survey by the industry analyst firm RedMonk.

Figure 4.6: Python

JavaScript

JavaScript (JS) is a lightweight, object-oriented programming language used for web development. It enables interactivity on web pages, supports event-driven and asynchronous programming, and allows manipulation of the DOM. JavaScript is versatile, cross-platform compatible, and has a rich ecosystem of libraries and frameworks. It is essential for creating dynamic and interactive web applications.

Figure 4.7: JavaScript (JS)

4.7.2 Development Environment

Google Colaboratory

Colaboratory, often shortened to "Colab", is a product of Google Research. Colab al- lows anyone to write and execute the Python code of their choice through the browser. It is a particularly suitable environment for machine learning, data analysis and ed- ucation. In more technical terms, Colab is a hosted Jupyter notebook service that requires no configuration and provides free access to IT resources, including GPUs.

Figure 4.8: Google Colaboratory

Kaggle

Kaggle is an online community platform for data scientists and machine learning enthusiasts. Users of Kaggle may work together, access and share datasets, use notebooks with GPU integration, and compete with other data scientists to solve data science problems. With the aid of its powerful tools and resources, this online platform—which was established in 2010 by Anthony Goldbloom and Jeremy Howard and was bought by Google in 2017 — aims to assist professionals and students in achieving their ob- jectives in the field of data science. Over 8 million people have registered on Kaggle(2021).

Figure 4.9: Kaggle

Visual Studio Code

Microsoft created the free open source text editor known as Visual Studio Code (often referred to as VS Code). Windows, Linux, and macOS all support VS Code. VS Code has recently become one of the most widely used development environment tools, it is a lightweight but powerful source code editor. From CSS, Go, and Dockerfile to Java, C++, and Python, VS Code supports a broad range of programming languages. In addition, VS Code enables users to install new extensions like code linters, debuggers, and support for cloud and web development.

Figure 4.10: Visual Studio Code

4.7.3 Front-End Technologies

ReactJs

An open-source JavaScript package called React.js is used to create user interfaces for single-page apps. It manages the view layer for both online and mobile applications. Reusable UI components may be made using React as well. Jordan Walke, a soft- ware developer for Facebook, developed React at first. In 2011 and 2012, Facebook's newsfeed and Instagram both used React for the first time. With the aid of React, programmers may build substantial online apps that can modify data without refreshing the page. React's primary goals are to be quick, scalable, and easy to use. It only functions with the application's user interfaces. This relates to the MVC template's view. It may be used in conjunction with a number of different JavaScript libraries or frameworks, including Angular JS in MVC.

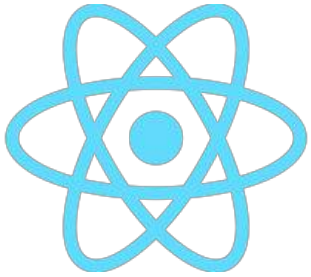


Figure 4.11: ReactJS

4.7.4 Back-End Technologies

Flask

Web application development is made simple with the help of the Python package Flask. It is a micro-framework without an ORM (Object Relational Manager) or similar capabilities, and as such, has a compact and simple-to-extend core. This indicates that flask offers you the technologies, libraries, and tools necessary to create the web application. Due to its minimal weight and focus on providing only necessary components, Flask is referred to as a micro-framework. It is a WSGI (Web Server Gateway Interface) web app framework that just offers the elements required for web development, such as routing, request processing, sessions, and so on. The developer can create a new module or employ an extension for the additional features, such as data management.

Figure 4.12: Flask

4.7.5 Libraries

Keras

In contrast to other APIs, Keras was designed for human beings, not machines. Best practices for decreasing cognitive load are followed by Keras, which provides consistent and simple APIs, reduces the amount of user activities necessary for typical use cases,and offers clear actionable error signals. Additionally, it includes a lot of development instructions and documentation.

Figure 4.13: Keras

Sickit-Learn

One of the most reliable libraries for machine learning in Python is Scikit Learn or Sklearn. It is open source and based on Matplotlib, SciPy, and NumPy. Through a standardized Python interface, it offers a variety of machine learning and statisti- cal modeling methods, such as dimensionality reduction, clustering, regression, and classification. It also offers a wide range of additional tools for model development, selection, evaluation, and data preprocessing.

Figure 4.14: Sickit-Learn

NumPy

The main Python library for scientific computing is called NumPy. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Pandas

Pandas is a Python open source library that is mostly used for machine learning and data science applications. It is based on the Numpy library, which supports multi-dimensional arrays. In the Python ecosystem, Pandas is one of the most widely used data wrangling packages. It integrates well with a variety of other data science modules, and it is typically

Figure 4.15: Numpy

available in all Python distributions, including those sold by commercial vendors like ActiveState's ActivePython and those that come with your operating system.

Figure 4.16: Pandas

4.7.6 communication Tools

Github

For software developers, GitHub is a web-based platform for version control and collaboration. By offering a web interface to the Git code repository and collaboration management tools, GitHub makes social coding easier. Linus Torvalds developed the distributed software management tool known as Git initially for the Linux Kernel Development. Software developers can use GitHub as a professional social networking platform.

Figure 4.17: Github

4.7.7 Other Tools

Overleaf

Writing, revising, and publishing scientific publications online is considerably quicker and simpler using Overleaf, an online LaTeX and Rich Text collaborative writing and publishing tool. Overleaf was developed with the intention of accelerating, opening up, and improving access to science and research. It centralizes the whole scientific documentation process, from conception through writing, review, and publishing. An intuitive LaTeX editor is offered by Overleaf so that numerous people may work simultaneously on any document. In order to give users a real-time preview of each created document, LaTeX code output is automatically compiled.

Figure 4.18: Overleaf

Chapter 5

Conclusion

This report laid out the groundwork for RunRightAI, an ambitious project aiming to revolutionize sprint analysis using readily available smartphone technology. We delved into the limitations of traditional methods and proposed a solution leveraging pose estimation, object detection, and AI-driven analysis for a comprehensive and accessible system. While we established a strong theoretical foundation and detailed our approach for data collection, labeling, and model training, we were unable to fully realize the project within the given timeframe. This was primarily due to unforeseen technical challenges encountered during the implementation phase, particularly in accurately mapping joint movements to real-world coordinates and integrating the RAG model for report generation. Despite these setbacks, the project holds significant potential. Future work should focus on: Refining the homography-based coordinate transformation for improved accuracy in real-world scenarios. Optimizing the RAG model for generating insightful and actionable performance reports tailored to sprinting. Developing a user-friendly interface for seamless interaction with the system, targeting both athletes and coaches. By overcoming these remaining hurdles, RunRightAI has the potential to become a valuable tool for athletes and coaches at all levels, democratizing access to advanced sprint analysis and fostering data-driven performance improvement.

Bibliography

[1] *Real-time human pose estimation in the browser with tensorflow.js*, 2017. [Online]. Available: https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5.

[2] *Tensorflow. pose estimation*, Feb. 17, 2022. [Online]. Available: https://www.tensorflow.org/lite/examples/pose_estimation/overview.

[3] *Tensorflow. movenet: Ultra fast and accurate pose detection model.* [Online]. Available: https://www.tensorflow.org/hub/tutorials/movene.

[4] *Tensorflow. pose estimation and classification on edge devices with movenet and tensorflow lite.* [Online]. Available: https://blog.tensorflow.org/2021/08/pose-estimation-and-classification-on-edge-devices-with-MoveNet-and-TensorFlow-Lite.html.

[5] *Tensorflow. next-generation pose detection with movenet and tensorflow.js.* [Online]. Available: https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html.

[6] *Tfhub.dev. movenet/singlepose/lightning.* [Online]. Available: https://tfhub.dev/google/movenet/singlepose/lightning/4.

[7] *Tfhub.dev. movenet/singlepose/thunder.* [Online]. Available: https://tfhub.dev/google/movenet/singlepose/thunder/4.

[8] *Github. xingyizhou/centernet.* [Online]. Available: https://github.com/xingyizhou/CenterNet.

[9] *Github. tensorflow/models.* [Online]. Available: https://github.com/tensorflow/models/tree/master/research/object_detection.

[10] *Medium arcticle. five popular cnn architectures clearly explained and visualized*, 2020. [Online]. Available: https://towardsdatascience.com/%205-most-well-known-cnn-architectures-visualized-af76f1f0065e.

[11] *Medium arcticle. what is convolutional neural network (deep learning).* 2020. [Online]. Available: https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5..

[12] O. HABIMANA, Y. LI1, R. LI1, X. GU1, and G. YU2, "Sentiment analysis using deep learning approaches: An overview," 6 December 2019.

[13] *Sentiment140*, 2020. [Online]. Available: http://help.sentiment140.com/site-functionality.

[14] *Tweets airline*, 2020. [Online]. Available: https://www.kaggle.com/crowdflower/twitter-airline-sentiment.

[15] *Tweets semeval*, 2020. [Online]. Available: http://alt.qcri.org/semeval2017/.

[16] *Amazon review dataset*, 2007. [Online]. Available: http://jmcauley.ucsd.edu/data/amazon/.