***System Architecture***

A diagram of a software application

Description automatically generated

System architecture plays an important role in defining the structure and behavior of our project. It helps us to better understand how the project works and ensures that it functions as intended throughout its lifecycle. This includes defining the relationships between components, data flows, service compositions, and subsystems. by designing an effective system architecture that prioritizes scalability, and interoperability, among other objectives, and helps us to have greater confidence in the system's ability to meet its requirements and objectives [36].

System architecture has five layers : Application layer, Logic layer, E2E Video Summarizer layer, database layer and API layer as shown figure 3.1, we explain more details about each layer in the following section.

**Application Layer:**

The Android and iOS application provides users with comprehensive access to football match information. Users can easily register using their email and password or directly through their Gmail account. Once registered, they can log in to unlock a variety of features. These include live match results as they happen, detailed outcomes of past matches, and timely updates on upcoming matches. Users can also view league standings to track their favorite teams' positions.

The standout feature of our application is its ability to deliver match summaries and key actions immediately after each match concludes. This ensures that users stay informed about all the important moments and developments across various football games. Whether it's goals, penalties, or other significant events, our app keeps fans updated in real-time.

In essence, our application serves as an indispensable tool for football enthusiasts, offering not only live updates and historical data but also ensuring they never miss a beat when it comes to their favorite sport.

**Logic Layer:**

Tell Details about how you make authentication in detail

**Database Layer:**

The Database Layer acts as a central repository for storing and managing system data. It provides the infrastructure for administrators and database managers to control and manipulate system resources.

This layer is responsible for storing the generated videos, user data.

The Database Layer supports functionalities such as video creation, editing, deletion, and addition, facilitating seamless integration with the other layers of the system.

**API Layer:**

The API layer is the backbone of our application, supplying all necessary data regarding live matches, match statistics, league standings, and more. It acts as a central hub where our application retrieves real-time updates and comprehensive information about ongoing matches. This includes live scores, player statistics, team performance metrics, and detailed match summaries.

Furthermore, the API facilitates access to historical data, enabling users to delve into past match results, player performances over time, and statistical trends. This wealth of information empowers users to analyze and compare teams and players effectively.

In addition to match-specific data, the API also serves up-to-date league standings, reflecting the current positions of teams based on their performance throughout the season. This feature allows fans to track their favorite teams' progress and standings in real-time.

Overall, the API layer plays a crucial role in ensuring our application delivers accurate, timely, and comprehensive football data, enhancing the user experience by providing all the information needed to stay informed and engaged with the sport.

**E2e Spot Video Summarizer Layer:**

* **Video Pre Processing**

1. **Frame Extraction**

**Firstly, We get the complete video and extract frames of video so that all videos with different frame rate have the same frame rate . It doesn’t matter what the frame rate of video , We take always two frames per second from each video so It makes training more consistent. We calculate stride by dividing frame rate of complete video by wanted frame rate ,in our case wanted frame rate is 2 frames per second, and we get the frames using this stride.**

1. **Divide frames into groups**

**We will divide all frames extracted in the first step into groups so that each group has same number of frames because We will use each group as training example and we want in our model to calculate the score and class of each frame so that we want all training examples to have same number of frames to have same number of neurons in output layer and make our model end to end. In our case we divide frames into groups so that each group have 100 frames**

1. **Data Augmentation for each training example**

**We train E2E-Spot on 100-frame-long clips sampled randomly and use standard data-augmentations**

1. **Resizing**

Resizing is an important step in the E2E Spot Model, as it ensures that the input data has a consistent size and shape, which is necessary for the model to learn the action spotting task more consistently. Resizing involves changing the dimensions of driving video frames to a fixed size, such as 224x224 pixels. This can be done using image processing libraries such as OpenCV or Pillow. Resizing the images to a smaller size can also help reduce the computational load on the model during training and inference. By standardizing the input dimensions, the model can process data more efficiently, leading to improved performance and accuracy in spotting actions.

1. Cropping

Cropping is another essential preprocessing step in the E2E Spot Model. It focuses on removing irrelevant parts of the video frames and concentrating on the region of interest. This technique can enhance the model's ability to detect and recognize actions by eliminating background noise and distractions. Cropping can be implemented using image processing libraries like OpenCV, where specific coordinates are defined to extract the desired portion of the frame. By focusing on key areas, cropping not only improves the model's accuracy but also reduces the amount of data it needs to process, thereby optimizing computational resources and speeding up both training and inference phases.

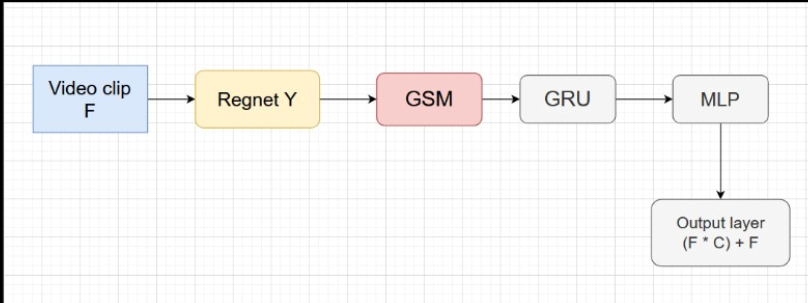
1. MixUP

MixUp is a data augmentation technique that plays a vital role in enhancing the robustness and generalization capability of the E2E Spot Model. It involves combining pairs of images and their corresponding labels to create new training samples. Specifically, two images are blended together with a certain weight, and their labels are also mixed in the same proportion. This can be done using libraries such as NumPy or custom functions within deep learning frameworks. MixUp helps the model to generalize better by providing it with more diverse training examples, thus making it less likely to overfit on the training data. This technique has been shown to improve the model's performance on unseen data, making it more effective in real-world action spotting scenarios.

* E2e Spot Model

This is the core component of this layer and it is responsible to get the tag for each frame . It is mainly made up of 4 components :RegNet Y,Gate Shift module (GSM),GRU, MLP .

E2E-Spot treats a video classification network as part of a sequence model, so that processing a clip of N frames results in 100 output features and N per-frame predictions. Figure 2 illustrates our pipeline. Frames from each RGB video group are first fed to a local spatial-temporal feature extractor F, which produces a dense feature vector for each frame This lightweight feature extractor incorporates Gate Shift Modules (GSM) [57] into a generic 2D convolutional neural network (CNN) [46]. The feature sequence is then further processed by a sequence model G, which builds a long-scale temporal context and outputs a class prediction for every frame, including a ‘background’ class to indicate when no event was detected.



1. Local Spatial-Temporal Feature Extractor and GSM

The first stage of our pipeline extracts spatial-temporal features for each frame. We strive to keep the feature extractor as lightweight as possible, but found that a simple 2D CNN that processes frames independently [9,23,60,63] is often insufficient for precise spotting. This is because a 2D CNN does not capture the spatially-local temporal correlations between frames. In videos that are densely sampled (24–30 FPS), this temporal signal is critical to learn features that can robustly differentiate otherwise very similar frames: for instance, the speed and travel direction of a ball, when each frame likely exhibits motion blur. To obtain more expressive, motion-sensitive features we implement F as a 2D CNN with Gate Shift Modules (GSM) [57]. We choose RegNet-Y [46], a recent and compact CNN, as the 2D backbone. Our feature extractor is similar to models for video classification [37, 57, 63], but with two key differences: (1) it samples frames densely and (2) it uses no final temporal consensus/pooling because our goal is to obtain one output per frame, rather than one for the whole video or multi-frame segment.

We choose RegNet-Y [46] over the more commonly used ResNet [26] family of 2D CNNs because the former is more recent and compact (RegNet-Y 200MF has 3.2M parameters vs. 11.7M parameters for ResNet-18), while exhibiting generally better performance on image classification benchmarks [64]. E2E-Spot, however, can be implemented with any 2D CNN architecture.

1. Long-term Temporal Reasoning Module

To gather long-term temporal information, we use a 1-layer bidirectional Gated Recurrent Unit (GRU [8]) network G, which processes the dense per-frame features produced by F. We set the hidden dimension of G to match that of F. We found that a single-layer GRU suffices and that more complex sequence models such as MS-TCN [20] or a deeper GRU do not necessarily improve accuracy We hypothesize that as a result of end-to-end training, the features produced by F capture subtle temporal cues that are specific to a given activity’s and task’s requirements. This shifts the burden of representations to F so that G only needs to propagate the temporal context.

1. MLP

Finally, we apply a fully connected layer and softmax on the GRU outputs to make a per-frame K+1 way prediction (including 1 ‘no-event’ background class).

**Create Summary Component**

After the E2e Spot Model predicts the output for each frame, we apply post-processing techniques to generate the summarized video. This process includes the following steps:

I. **Non-Maximum Suppression (NMS)** Non-maximum suppression is a crucial step in refining the model's predictions. It helps in eliminating redundant or overlapping detections, ensuring that the most relevant and precise predictions are retained. We evaluated the model predictions with and without non-maximum suppression. For temporally precise datasets, we used a window of ±1 frames to suppress non-maximum scores. However, for SoccerNetv2, which operates at 2 FPS, we use a window of ±2 frames. The effectiveness of NMS in a temporally precise setting depends on several factors, including frame-level tolerance, the specific dataset, and the model's characteristics. Therefore, the decision to apply NMS should be tailored to the specific application and task requirements, ensuring optimal performance and relevance of the summarized content.

II. **Frames Concatenation** After obtaining the predictions for each frame, we perform frames concatenation to compile the summarized video. We select and concatenate frames that have a prediction score greater than 0.5 and correspond to one of the desired action classes. This threshold ensures that only frames with significant confidence in their action detection are included, thereby enhancing the relevance and clarity of the summarized video. By focusing on high-scoring frames, we effectively create a concise and informative summary that highlights the key moments and actions within the original video.

III. **Additional Post-Processing Techniques** In addition to NMS and frame concatenation, other post-processing techniques can be applied to further enhance the quality of the summarized video. These may include:

* **Smoothing Transitions**: Applying techniques to smooth transitions between concatenated frames can help in creating a more visually coherent and pleasant viewing experience.
* **Frame Rate Adjustment**: Adjusting the frame rate of the summarized video to ensure it matches the desired playback speed, enhancing the temporal coherence and making the summary more natural to watch.
* **Highlight Enhancements**: Adding visual or audio highlights to key actions or moments can make the summary more engaging and easier to understand for viewers.

***References***

36. "Software Architecture & Design Introduction," [Online]. Available: https://www.tutorialspoint.com/software\_architecture\_design/introduction.htm. [Accessed 13 May 2023].

57. Sudhakaran, S., Escalera, S., Lanz, O.: Gate-shift networks for video action recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)

46. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollar, P.: Designing network design spaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)

37. Lin, J., Gan, C., Han, S.: TSM: Temporal shift module for efficient video understanding. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)

63. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Val Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: Proceedings of the European Conference on Computer Vision (ECCV) (September 2016)

8. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: Proceedings of NIPS Deep Learning and Representation Learning Workshop (2014)

20. Farha, Y.A., Gall, J.: MS-TCN: Multi-stage temporal convolutional network for action segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)

26. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)

64. Wightman, R.: PyTorch image models. https://github.com/rwightman/ pytorch-image-models (2019)

***System Implementation***

***1- Data Set***

***1.1 Dataset Introduction***

A dataset is a set of structured and interconnected data, is considered an essential component for the development of machine learning and data learning models and is considered a powerful tool for understanding data and using it to achieve various goals in various fields and applications.

Documentation for a dataset should include information about the data sources, the preprocessing techniques used, and the format of the dataset. It should also include any specific instructions for loading and using the dataset in a machine learning framework. Additionally, it is helpful to include information about any potential biases in the data and any limitations or caveats to be aware of when using the dataset.

Overall, creating a high-quality dataset is a critical step in building effective machine learning models. By following best practices for data collection, preprocessing, and organization, you can ensure that your dataset is clean, consistent, and relevant to your problem, making it easier to train and test your models effectively.

***1.2 Dataset Challenges***

In the Beginning we spend too much time to get the dataset that contain label for each frame for many matches videos . It was very difficult task , Fortunately we find soccernet v2 [7] dataset that contain label for each frame for 500 video of matches in different countries such as England , Germany ,Spain . But there was a very big problem that dataset is about 800 Giga Bytes and We don’t have resources to deal with that . We decide to train the model in batches to over come this problem and It was a great solution and we trained the model using this approach

***1.3 Dataset Description***

SoccerNet v2 is a comprehensive and richly annotated dataset specifically designed for research in video understanding and action spotting within the context of soccer matches. It builds upon the original SoccerNet dataset by providing an expanded and more detailed collection of soccer videos and their corresponding annotations. Here are the key features and components of the SoccerNet v2 dataset:

1. **Content and Scope:**
   * SoccerNet v2 contains full-length soccer match videos sourced from various leagues and competitions. The dataset includes both men's and women's matches, providing a diverse range of soccer gameplay.
   * The dataset comprises thousands of hours of video footage, ensuring extensive coverage of different game scenarios and events.
2. **Annotations:**
   * Event Annotations: SoccerNet v2 provides detailed annotations for various types of events occurring during the matches. These include goals, substitutions, yellow and red cards, corner kicks, free kicks, and more .
   * Temporal Annotations: Each annotated event includes precise temporal information, indicating the exact start and end times of the event within the video. This level of detail is crucial for training and evaluating action spotting models.
   * Spatial Annotations: In addition to temporal annotations, some events are also annotated with spatial information, highlighting the regions of the frame where the event takes place.
3. **Data Format:**
   * The dataset is provided in a structured format that includes video files and accompanying annotation files. Annotations are typically available in JSON or CSV format, making it easy to parse and utilize the data for various machine learning tasks.
   * Videos are provided in standardized formats to ensure consistency in preprocessing and analysis.
4. **Use Cases:**
   * Action Spotting: SoccerNet v2 is ideal for training models to detect and classify specific actions within soccer matches. The detailed temporal annotations enable precise action spotting, which is essential for applications such as highlight generation and game analysis.
   * Video Understanding: Researchers can use SoccerNet v2 to develop and evaluate algorithms for broader video understanding tasks, such as activity recognition, player tracking, and game strategy analysis.
   * Computer Vision and Machine Learning: The dataset supports a wide range of computer vision and machine learning research, including supervised learning, unsupervised learning, and reinforcement learning.

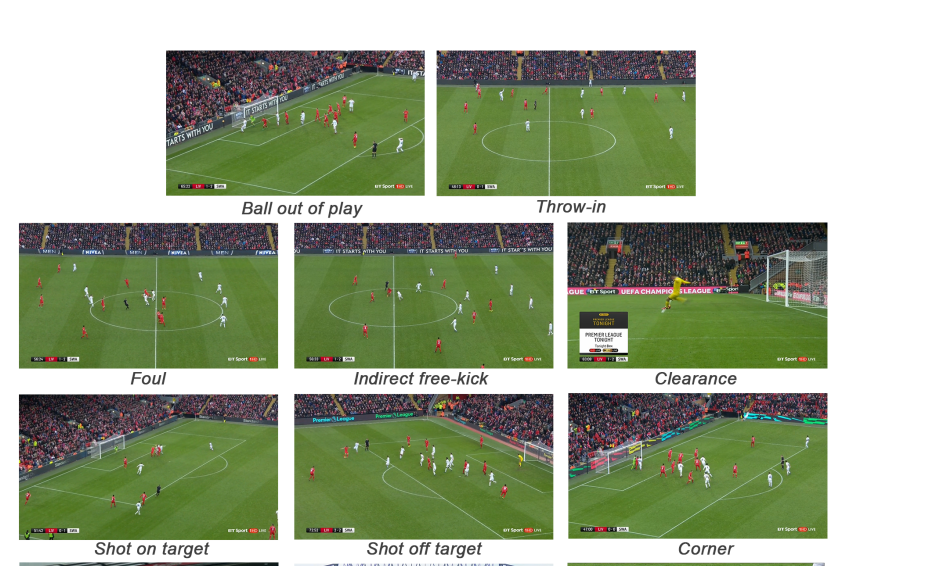
**5- Evaluation Metrics:**

SoccerNet v2 includes standardized evaluation metrics for assessing the performance of models on action spotting and other tasks. These metrics ensure fair and consistent comparison of different approaches and methodologies.

**6- Community and Benchmarking:**

SoccerNet v2 serves as a benchmark dataset for the soccer video understanding research community. It facilitates the comparison of different models and techniques, promoting advancements in the field.The dataset is often used in challenges and competitions, encouraging collaboration and innovation among researchers.In summary, SoccerNet v2 is a vital resource for advancing the state-of-the-art in video understanding and action spotting within the context of soccer. Its rich annotations, extensive coverage, and standardized evaluation protocols make it an essential tool for researchers and practitioners in the field.

***1.4 Dataset Samples***



A collage of a football game

Description automatically generated

A screenshot of a computer code

Description automatically generated

***1.5 Dataset Preprocessing***

Functions that used in dataset Preprocessing

* Worker ( Video )
* It gets frame rate of video
* Then it calculates stride by dividing frame rate of video by desired frame rate which is 2
* It extract all frames in video and get frames using stride calculated in pervious strep
* It saves the frames in folder called frames
* ImageDataGenerator()  
  Make data augmentation
* Resizing
* Rotation
* Cropping
* MixUp
* Create traindataset()
* This function takes the frames folder and divide the frames into equally numbered groups each has 100 frame
* It takes annotations and get the annotations for frames in each group
* It converts annotations in Json format to corresponding output .

***2- Experiments & Results in model training***

I- ResNet & LSTM

* Training Procedure

Firstly we try to use Action Recognition Task to Summary Highlights in football matches .

We use Action Recognition model that consist of ResNet CNN for Feature Extraction [3] then LSTM [5] for learning sequence in videos inspired from [4] .We use soccerDB dataset in this model and we adjust hyperparameters used in [4] , We also make loss function for classes that isn’t back ground 5 times larger than background class , We also make strong data augmentation .

The Training procedure use this hyperparameters

* 3d-ResNet Input Configuration

The input shape is L\*w\*h\*C where L is number of stacked frames

We Experiment Using different values of L : L={10,16,18,26,30,32,64,128}

We found that optimal value is L = 64

* Learning Rate

Initial Learning Rate is 3\*10-3

Learning rate divided by 10 after every 10,000 iterations.

Decay Rate is 5 \* 10-4

* Optimizer

We use Stochastic Gradient Descent

* Batch Size

We use batch size = 16

* Evaluation Metric

The Evaluation Metric used for Action Recognition Model is Accuracy ,Precision, Recall.

Table 1 Shows the results of [4] and ours by making modification to [4]

|  |  |  |  |
| --- | --- | --- | --- |
| Paper/Metric | Accuracy | precision | recall |
| Lstm+ResNet [4] | 96.81% | 93% | 91% |
| LSTM+ResNet (ours) | 98.3 % | 97% | 96% |

Problems in this Model

Because the model is based on Action Recognition So Entire Action should be in video being predicted but Actually we split Complete Football Match Video into parts and make action recognition in each part .. what will happen if Action is at end of one part and continuing in start of next part ? The model will not predict the Action Correctly so we come to Action Spotting Task not Action Recognition Task .

2- E2e Spot Model

This is our final model we make it and This is the model we use it. we inspire from [8].

Modification We make :

1- Adjusting Huperparameter

2- Decrease Number of classes into 11 Class We are Interested In.

3- Train Data in Batches Which are more quickly

Training Procedure

 E2E-Spot model trained using 100-frame long clips and a batch size of 8 clips.

 Training batches randomly sampled from the training videos.

 Training cycle comprises 625 steps, equivalent to a pseudo-epoch of 500K frames.

 Each training cycle runs approximately:

* 8.5 minutes for the 200MF variant.
* 14 minutes for the 800MF variant on a single A5000 GPU[44].

 Datasets trained and cycle durations:

* SoccerNet-v2:
  + 200MF model: 150 cycles.
  + 800MF model: 200 cycles.

 AdamW optimizer[43] used with a base learning rate of 10-4 including 3 linear warmup cycles and cosine decay[42].

Loss Function is cross entropy loss and To mitigate imbalance arising from the rarity of precise events (< 3% of frames), we boost the loss weight of the foreground classes (5×) relative to the background

Evaluation Metric

We measure Average Precision within a tolerance of δ frames (AP @ δ). AP is computed for each event class, and mAP is the mean across classes. We focus on tight tolerances such as δ = 1 and δ = 2. Precise temporal events are rare as a percentage of frames (0.2–2.9%), so metrics such as frame-level accuracy are not meaningful for precise spotting.

Table 2 Shows Results and Comparisons

|  |  |  |
| --- | --- | --- |
| Model/Metric | MAP (Tight) | MAP (Loose) |
| Baseline | 58% | 61% |
| E2eSpot[8] | 78.6% | 80.5% |
| E2eSpot Fine Tuning Only(ours) | 79.5% | 81.25% |
| E2eSpot with 11 Classes | 86.7% | 87.5% |

Table2: Tight means with tolerance 0-5 seconds and loose with tolerance 30-60 seconds only.

***3- Software Tools Used***

We use many

• Dataset tools

➢ Firebase

Firebase is a powerful tool for developers, as it provides a wide range of pre-built features and services that can be easily integrated into our application. Firebase helps us to save time and effort by handling many of the complex tasks associated with application development, such as data storage, authentication, and analytics. Additionally, firebase offers a scalable and reliable infrastructure that can support our application of any size

• Deep learning models tools

➢ Google colab

Google colab is a cloud-based platform for developing and running python code in a jupyter notebook environment. It is a free service provided by google that allows users to write and run python code in their web browser, without requiring any local installation or setup, and we used it to create our deep learning model.

➢ Python

Python is a versatile, high-level programming language that is widely used for a variety of applications, including web development, data analysis, machine learning, and scientific computing and we used python to create our deep learning model

Desktop application tools

➢ Pycharm

Pycharm is a popular code editor developed by JetBrains that used to create our desktop application using frameworks.

➢ Tkinter

Tkinter is a Python library that provides a set of tools for creating graphical user interfaces (GUIs). It is a built-in library in Python and requires no additional installation, and we used it in our project to design desktop application.

• Mobile Application Tools

➢ Android Studio

Android Studio is an integrated development environment (IDE) developed by Google that is designed specifically for developing Android applications. Android Studio provides a range of tools and features that can be used to create our mobile application using flutter.

➢ Flutter

Flutter is an open-source mobile application development framework developed by Google that allows developers to build high-quality, cross-platform mobile applications for iOS and Android using a single codebase. Flutter provides a rich set of pre-built widgets and tools that can be used to create our android application.

• Other Tools

➢ GitHub

GitHub is a web-based platform that provides a range of tools and features for software development, version control, and collaboration, we used it to manage and track changes to our code

References

7- SoccerNet-v2: A Dataset and Benchmarks for Holistic Understanding of Broadcast Soccer Videos

4- Soccer Video Summarization using Deep Learning

3-Deep Residual Learning for Image Recognition [Kaiming He](https://arxiv.org/search/cs?searchtype=author&query=He,+K), [Xiangyu Zhang](https://arxiv.org/search/cs?searchtype=author&query=Zhang,+X), [Shaoqing Ren](https://arxiv.org/search/cs?searchtype=author&query=Ren,+S), [Jian Sun](https://arxiv.org/search/cs?searchtype=author&query=Sun,+J)

5- Hochreiter, S., & Schmidhuber, J"urgen. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

8- Spotting Temporally Precise, Fine-Grained Events in Video

42. Loshchilov, I., Hutter, F.: SGDR: Stochastic gradient descent with warm restarts. In: Proceedings of the International Conference on Learning Representations (ICLR) (2017)

43. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: Proceedings of the International Conference on Learning Representations (ICLR) (2019)

44. Nvidia: Nvidia RTX A5000 data sheet (2021)