COSE474-2024F: Final Project Report
"Automated Classification Actions for Soccer Match Highlights Using CLIP"

Yoo Jiyeon

1. Introduction

The rise of sports fans worldwide, paired with the ongoing success of multiple leagues, means that large amounts of match footage and related data are produced on a daily basis. As the technological landscape grows, this data has developed into a pillar of resource for both sports media and analytics, emphasizing the significance of tech solutions to create data management and analysis systems. As a way to keep the popularity, many sports teams advertise their matches on social media in order to attract the interest of people.

When it comes to soccer in particular, where a typical match can run for 90 minutes, encouraging potential fans to watch an entire match can be a tricky sell. But efficiently organizing and analyzing this massive dataset is another major challenge. Specifically, the detection and summarization of major events in soccer games is of particular interest for applications such as media highlight generation, player performance evaluation, and tactical analysis.

Therefore extracting match highlights is a highly commercializable objective and automating this process opens up a vast economic opportunity. In this project, it focus on building a model that can identify and classify key actions label in soccer matches automatically to meet this need and to explore how it can contribute to the sports industry in general.

Figure 1. Pair of image and labels for sample image from dataset.

Contributions.
(i) Development of a model for measuring similarity between action labels and target images for SoccerNet highlights: I propose a model that measures the similarity between reference images (key moments in soccer matches) and target images, along with a caption explaining the differences between these images.

(ii) Training the model using SoccerNet data with the CLIP model: I propose training the model by applying the differences in image features and captions between key soccer moments in the SoccerNet dataset to the CLIP model, enabling it to learn the relevant features for accurate highlight classification.

2. Related Works

SoccerNet dataset provides a standardized benchmark for action spotting in soccer videos, enabling advancements in video analysis (Giancola et al., 2018) **[1]**. It contains annotations for events such as goals, substitutions, and yellow/red cards. The dataset supports developing models for tasks like temporal action localization and video understanding, offering challenges with benchmark metrics. SoccerNet enables researchers to evaluate their methods on standardized data, fostering advancements in sports video analysis.

3. Methods

In this project, it applied the residual learning method using Resnet as an image encoder to the existing CLIP model to achieve better performance in image feature extraction. In addition, I attempted to achieve significant performance improvement in a limited amount of image data through data augmentation techniques.

In processing the dataset, the replay scene for one action was also trained with the same label, thereby training continuous images for one situation and images from various angles.

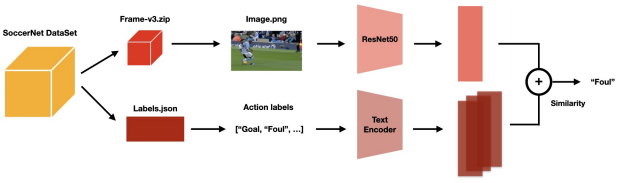

Figure 2. Project Pipeline.

3.1. Algorithms

(i) data preprocessing pseudo code:

Algorithm 1 DataSet Preprocessing

Data: ZIP file path zip_path, JSON file path $json\_path$
Result: List of image tensors $image\_tensors$ and their labels $labels$

Open ZIP file from zip_path Extract PNG file;
foreach $image\_name$ *in* $image\_names$ **do**
  Open PNG file $image\_file$ and convert to RGB
  Convert $image\_file$ to tensor $image\_tensor$
  Append $image\_tensor$ to $image\_tensors$ **if** $image\_name$ *contains* ' **then**
    Modify $image\_name$ to $base\_name$
  end
  Extract $label$ from JSON data using $base\_name$
  Append $label$ to $labels$
end

(ii) Implementation of CLIP pretrained model:

Algorithm 2 Training Loop for CLIP Model

Result: Trained CLIP model with updated parameters
for $epoch = 1$ **to** $EPOCHS$ **do**
  foreach $batch$ *in* $train\_loader\_basic$ **do**
    Load $images$ from batch["image"]
    Convert $batch["label"]$ to tensor $labels$; Create $texts$ by mapping $labels$ to $text\_prompts1$
    Prepare inputs for CLIP processor:
    Process $texts$ and $images$ using $clip\_processor$ and move to $DEVICE$
    Feed inputs to CLIP model and get outputs:
    Pass $inputs$ through $clip\_model$ and get $logits\_per\_image$ and $logits\_per\_text$
    Compute contrastive loss
    Backpropagation and optimization:
    Zero gradients in $adamw\_optimizer$
    Perform backpropagation with $loss$ and update model parameters
  end
  Update learning rate using scheduler
end

4. Experiment

4.1. DataSet

I used the Actions and replay images dataset by SoccerNet and it consists of images of some actions from the action spotting annotations and the same moment in their corresponding replays. The SoccerNet dataset was initially constructed using 500 complete broadcast soccer game videos from 2014 to 2017. These videos were sourced from six major European leagues—Serie A, La Liga, Premier League, Ligue 1, Bundesliga, and the Champions League—and amounted to approximately 800 hours of footage (Wu et al., 2022).**[2]** In this project, only data corresponding to 40 games was extracted and used. The number of images corresponding to each game is as shown in the Table 1.

Table 1. The number of images included in the dataset and the number and ratio of each item.

DATA SET	#IMAGE	PROPORTION
TRAINING	2045	70%
TEST	684	20%
VALIDATION	269	10%

4.2. Computer Resouce

- Environment: Google Colab PRO
- CPU: Intel(R) Xeon(R) CPU @ 2.20GHz
- GPU: NVIDIA A100-SXM4-40GB
- OS: Linux

4.3. Result

Data Augmentation: Data Augmentation is a technique for generating new data with various data orientations. Data augmentation solves two concerns for researchers: first, it generates more data from a limited amount of data, and second, it minimizes overfitting.**[3]**

As you can see in the figure1, it can increase the diversity of your data set by rotating, flipping, and resizing a single image data.

Figure 3. Image after several transformations.

Originally, I tried to improve performance by applying data augmentation of four techniques, but due to the problem of system RAM, only resize and horizontal flip were used. The number of data for each item when data augmentation was applied is as follows. Although there is no significant difference in the epoch due to the lack of use of several augmentation techniques, it can be seen that this is an essential process when training a model using a small dataset.

- batch size: 8
- learning rate = 1e-5
- epoch = 10
- optimizer: AdamW

Table 2. Comparison the number of images included in the dataset and after do data augmentation

DATA SET	#IMAGE	DATA AUGMENTATION #IMAGE
TRAINING	2045	6135
TEST	684	2052
VALIDATION	269	807

Epoch	Basic	Augmentation
Epoch 1	1.8468	1.9313
Epoch 3	1.2573	1.4151
Epoch 5	0.9248	0.9986
Epoch 7	0.7601	0.8437
Epoch 9	0.7146	0.7462
Test Accuracy	26.90%	16.18%

Table 3. Loss values for basic and data augmentation dataset.

As a result, when comparing the test accuracy, it can be confirmed that the side that applied data augmentation showed lower accuracy. This is analyzed as being because the number of augmented data was too large compared to the number of original data, so the model overfitted the augmented data more than the original data.
When looking at table 2, the number of train data on the side where data augmentation was applied is about 4,000 more. This means that the ratio of augmented data with applied transformation in the train data set is higher. If the ratio of augmented data to original data is not even, you can see that the performance actually gets lower. Therefore, we can conclude that applying a lot of transforms indiscriminately does not have a good effect on the model's performance, and that it is important to maintain an appropriate ratio.

Optimizer and Scedular: A total of 4 optimizers are used in this project and compare their performance by using loss function decline rate. 4 categories of optimizers: 1) AdamW 2) SGD 3) Momentum 4) adaGrad. As shown figure, this is the result when four types of Optimizers were trained in the same environment with epochs of 10, learning rate of 1e-5, and batch size of 8. Through the figure, it can see that SGD and momentum have similar loss values, and adamW and adagrad have similar loss values.

Epoch	AdamW	SGD	Momentum	Adagrad
Epoch 1	1.8468	1.9337	1.9272	1.7396
Epoch 3	1.2573	1.7086	1.7026	1.2749
Epoch 5	0.9248	1.6229	1.6196	1.0538
Epoch 7	0.7601	1.5388	1.5207	0.9382
Epoch 9	0.7146	1.4898	1.4833	0.8709
Test Accuracy	26.90%	23.25%	22.95%	25.73%

Table 4. Loss values for different optimizers across odd epochs.

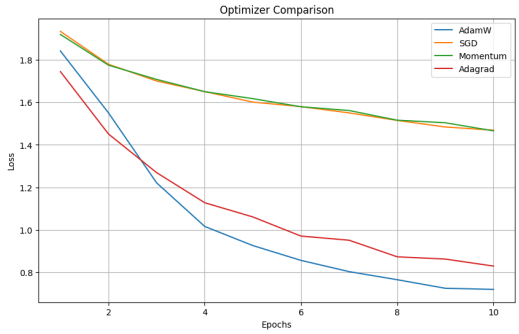

Figure 4. Comparison loss value with optimizer.

The loss values of adamW and adagrad were significantly lower than those of the other two optimizers, but there was no significant difference in test accuracy. AdamW and Adagrad were able to reduce the loss more effectively and achieve high test accuracy compared with other two optimizers, by using Adaptive Learning Rate.

These are some things it can be analyze when comparing the results of optimizers: **Adaptive Learning Rate:** AdamW and Adagrad dynamically adjust the learning rate to help the model learn effectively from various data distributions. **Regularization Effect:** It is likely that the L2 regularization effect of AdamW and the learning rate reduction strategy of Adagrad prevented overfitting of the model and improved generalization performance. On the other hand, SGD and Momentum were found to be inefficient in complex data distributions due to their simple optimization methods. However, the reason why models did not obtain good test accuracy overall is that, it is thought to be due to the poor learning performance of the model itself and structural limitations of the model.

It was an attempt to apply parameters and learning rates optimized for the model by applying various optimizers and comparing them, but considering the low test accuracy that was recorded as a result, it seems difficult to say that it was successful model learning. In order to improve this, additional experiments will be needed to improve performance through optimization of the optimizer's hyperparameters and improvement of the model structure.

5. Future Works

The limitation of the current model is that the type of input is limited to static images. In addition, since classification is performed only within the given label, it is difficult to flexibly determine various situations of soccer game. Therefore, in order to develop this project, it will be important to use full-time soccer game video as input, divide the frame into

given units, and preprocess the data as images. In addition, by adding sound information to the image and text pair information, the accuracy of action spotting can be increased. For example, audio information, such as the characteristic that the decibel level of the crowd rises higher than average in a goal situation, can also play an important role in improving the performance of action spotting.

6. Appendix

GitHub History

https://github.com/jiyeonyooo/20242R0136COSE47402

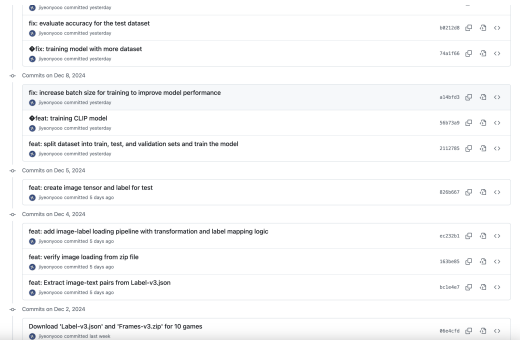

Figure 5. GitHub history.

Overleaf History

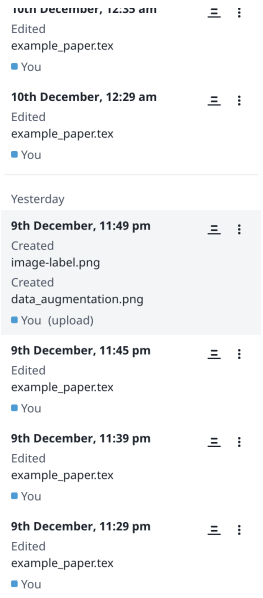

Figure 6. Overleaf history.

7. References

[1] Giancola, S., Amine, M., Dghaily, T., & Ghanem, B. (2018). Soccernet: A scalable dataset for action spotting in soccer videos. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 1711-1721).

[2] Giancola, S., Cioppa, A., Deliège, A., Magera, F., Somers, V., Kang, L., ... & Li, Z. (2022, October). SoccerNet 2022 challenges results. In Proceedings of the 5th International ACM Workshop on Multimedia Content Analysis in Sports (pp. 75-86).

[3] Maharana, K., Mondal, S., & Nemade, B. (2022). A review: Data pre-processing and data augmentation techniques. Global Transitions Proceedings, 3(1), 91-99.