CVT5: Using Compressed Video Encoder and UMT5 for Dense Video Captioning

Mohammad Javad Pirhadi, Motahhare Mirzaei and **Sauleh Eetemadi**
Iran University of Science and Technology at Tehran, Iran
{mohammad_pirhadi, m_mirzaei96}@comp.iust.ac.ir, sauleh@iust.ac.ir

Abstract

The dense video captioning task aims to detect all events occurring in a video and describe each event using natural language. Unlike most other video processing tasks, where it is typically assumed that videos contain only a single main event, this task deals with long, untrimmed videos. Consequently, the speed of processing videos in dense video captioning is a critical aspect of the system. To the best of our knowledge, all published work on this task uses RGB frames to encode input videos. In this work, we introduce the use of compressed videos for the first time in this task. Our experiments on the SoccerNet challenge demonstrate significant improvements in both processing speed and GPU memory footprint while achieving competitive results. Additionally, we leverage multilingual transcripts, which seems to be effective. The encoder in our proposed method achieves approximately $5.4\times$ higher speed and $5.1\times$ lower GPU memory usage during training, and $4.7\times$ higher speed and $7.8\times$ lower GPU memory usage during inference, compared to its RGB-based counterpart. The code is publicly available at https://github.com/mohammadjavadpirhadi/CVT5.

1 Introduction

In the video captioning task, the input video is typically assumed to be very short, containing only one main event. The desired output in this case is a textual description of that event. However, this assumption does not hold for most real-world scenarios, where input videos are long, and multiple events occur at different times. The dense video captioning task, first introduced by Krishna et al. (2017), aims to detect all events in a long, untrimmed video and generate a description for each event using natural language. This task is challenging because the model must not only recognize objects in the video but also understand the actions and interactions between them. Solving this task bridges the fields of computer vision and natural language processing, attracting increasing attention. Dense video captioning has potential applications for blind people, human-robot interaction, and more. However, the proposed methods must be fast and accurate, enabling the system to detect and describe events in a timely manner using reasonable computational resources.

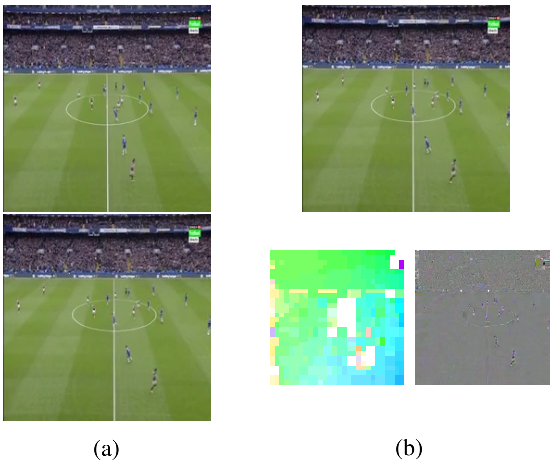

Figure 1: Comparison between two consecutive RGB (left) and compressed frames (right). The upper frames are identical and are the first frame of a video. The lower frames are the second frames of the same video. In the compressed format, the left frame represents the motion vector, and the right frame represents the residual.

To the best of our knowledge, all existing methods for dense video captioning use RGB frames to encode the input video. However, there is a significant amount of redundancy between consecutive RGB frames since most of the pixels remain unchanged. This redundancy demands considerable processing time and resources while providing minimal additional information. Video compression methods like H.264 significantly reduce the resources required for storage and transmission by keeping only a few frames intact and reconstructing the others using motion vectors and residuals. For instance, consider a video corresponding to half of a match in the SoccerNet dataset (Mkhallati et al., 2023). With a frame rate of 2 frames per second (FPS) and a resolution of 224×224, the video consists of 5,400 frames, requiring over $775MB$ of storage without compression. However, compression reduces this size to approximately $78.5MB$ making it about $10\times$ smaller. Figure 1 illustrates the comparison between two consecutive frames when using RGB frames versus compressed ones.

To address these challenges, we propose an end-to-end CNN-Transformer model to generate dense captions using videos in the compressed domain. Compressed videos consist of I-frames, motion vectors, and residuals, with minimal redundancy between consecutive frames. As a result, our model processes videos more efficiently, requiring less time and fewer resources during both training and inference. Additionally, we extract and utilize multilingual transcripts of the input video, which, as our experiments show, positively impact the results.

The contributions of this paper are summarized as follows:

1. We propose an end-to-end CNN-Transformer model for solving the dense video captioning task.
2. We leverage multilingual transcripts of the videos.
3. Our experiments on the SoccerNet dataset demonstrate significant improvements in processing speed during both training ($5.4\times$) and inference ($4.7\times$), along with a substantial reduction in GPU memory usage during both training ($5.1\times$) and inference ($7.8\times$).

2 Related Work

2.1 Dense Video Captioning

Most previous work on dense video captioning follows a two-stage approach: first, event proposals are generated, and then captions are created for these events. E2vid (Huang et al., 2020) separately extracts text and video features and passes them to a decoder to generate captions. The video frame features are extracted using a pretrained vision model on each RGB frame individually, which are then passed through a transformer. This paper employs three different pretraining tasks: text, text-video, and segment alignment and ordering. Similarly, PDVC (Wang et al., 2021) uses a pretrained RGB frame encoder followed by a transformer, along with N learnable queries to generate N events. After predicting the number of events in the video, it selects the most probable proposals as final events. This work utilizes a deformable transformer (Zhu et al., 2020) for faster convergence. GVL (Wang et al., 2023) also employs learnable queries after extracting features from RGB frames. This paper extracts features from ground truth labels and trains the model using two tasks: event-to-text generation and text-to-event generation and introduce semantic cost in addition to localization cost to enhance robustness against annotation noise. Vid2Seq (Yang et al., 2023) leverages a vast amount of YouTube videos available in the HowTo100M (Miech et al., 2019) and YT-Temporal-1B (Zellers et al., 2022) datasets to pretrain a transformer model with two objectives: generation and denoising. A key contribution of this work is the use of time tokens to generate events in a single stage, which proves effective. In our work, we utilize the common two-stage model as training a model for directly predicting event times requires a substantial amount of data.

2.2 Compressed Video Processing

Compressed videos have primarily been used in the action recognition task. Wu et al. (2017) first demonstrated that processing videos in the compressed domain improves both model speed and accuracy in the context of action recognition. Previous work on compressed domain action recognition can be categorized into three main approaches: using I-frames + residuals (e.g. Battash et al., 2020 and Abdari et al., 2019), using I-frames + motion vectors (e.g. Wang and Torresani, 2022 and He et al., 2022), and using I-frames + motion vectors + residuals (e.g. Wu et al., 2017 and Mou et al., 2024). There are also a few rare works that utilize macro-blocks (e.g. Chadha et al., 2017) or DCT coefficients (e.g. Ming et al., 2023). However, the common methods primarily rely on the three main categories mentioned above. Our best model uses I-frames and motion vectors only, as our experiments show that, at least for the SoccerNet dataset, including residuals has a negative effect.

3 Method

As mentioned above, the goal of our proposed method is to leverage compressed videos to en-

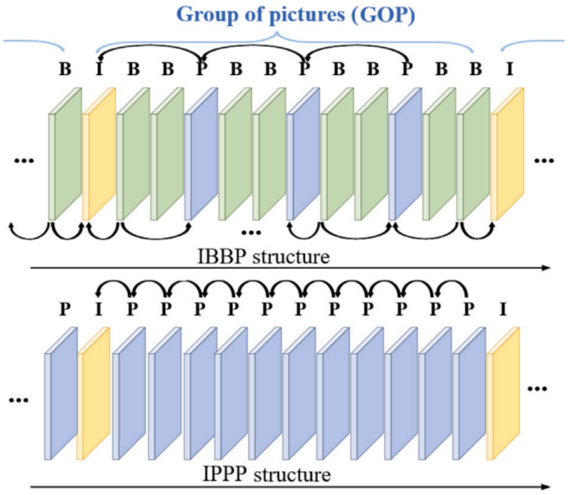

Figure 2: Two different possible structures of compressed videos. P-frames only refer to their previous frame, while B-frames can refer to any frame before or after them within their GOP. (Ming et al., 2024)

hance processing speed and reduce GPU memory usage. To achieve this, we first explain the structure of compressed videos in section 3.1. Then, we introduce the proposed model in section 3.2.

3.1 Compressed Video Structure

Modern video codecs like H.264 utilize temporal redundancy between successive video frames to compress video data. These codecs break down the video into multiple groups of pictures (GOPs) based on the differences between frames. Each GOP can be reconstructed independently without relying on other GOPs. Higher rates of change result in smaller GOPs, while lower rates of change produce larger GOPs. The first frame of each GOP is always an I-frame, which is a complete RGB image that can be reconstructed independently. The remaining frames can be either P-frames (predictive coded frames) or B-frames (bi-predictive coded frames). Both types of frames consist of a motion vector and a residual. The motion vector represents the movement of each macro-block in the current frame relative to the reference frame, and the residual captures the difference in color between frames after applying the motion vector to the reference frame. The reconstruction process can be formulated as follows:

$$F_{rec} = ApplyMV(F_{ref}, mv) + res \qquad (1)$$

where F_{rec} is the reconstructed frame, $ApplyMV$ applies the motion vector to the reference frame, F_{ref} is the reference frame, mv is the motion vector and res is the residual. Each macro-block typically consists of a group of 4×4 pixels, so the number of elements in the motion vector is $16\times$ lower than in the original RGB frame. The difference between P-frames and B-frames is that a B-frame can refer to any frame before or after it within its GOP, whereas a P-frame only refers to the previous frame. Using B-frames provides higher compression rates but makes it more challenging for the model to learn patterns. Therefore, we configure the FFMPEG package (Tomar, 2006) to use only P-frames.

3.2 Model Architecture

Figure 3 presents an overview of the proposed architecture. After preprocessing, extracting I-frames, motion vectors, and residuals, and dividing the input video into chunks, each chunk is processed through the following stages: 1. Encoding I-frames, motion vectors, and residuals separately. 2. Encoding the past, present, and future periods separately using a common transformer encoder (Vaswani et al., 2017). 3. Encoding the entire chunk and predicting each event individually. 4. If an event has occurred, generating a caption using the encoded frames of the entire chunk and transcript features from the present period.

Each of these steps is explained in more detail in the following sections.

3.2.1 Preprocess Videos

First, we preprocess the original videos using the FFMPEG package. After this processing, the videos are H.264 encoded, include only I-frames and P-frames, and have a resolution of 224×224 with a frame rate of 2. Typically, the H.264 codec creates GOPs of varying lengths based on the rate of change between frames. We investigated the impact of this behavior by comparing it to a setup where we enforced a GOP size equal to our short memory length. The experiments indicate that using a dynamic GOP size benefits the model. For more details, refer to section 4.6.3.

3.2.2 Extract I/P-frames

We extract I-frames, motion vectors, and residuals using the tool described in (Shen, 2023). I-frames and residuals are saved in .jpg format, while motion vectors are saved in .png format. This is because

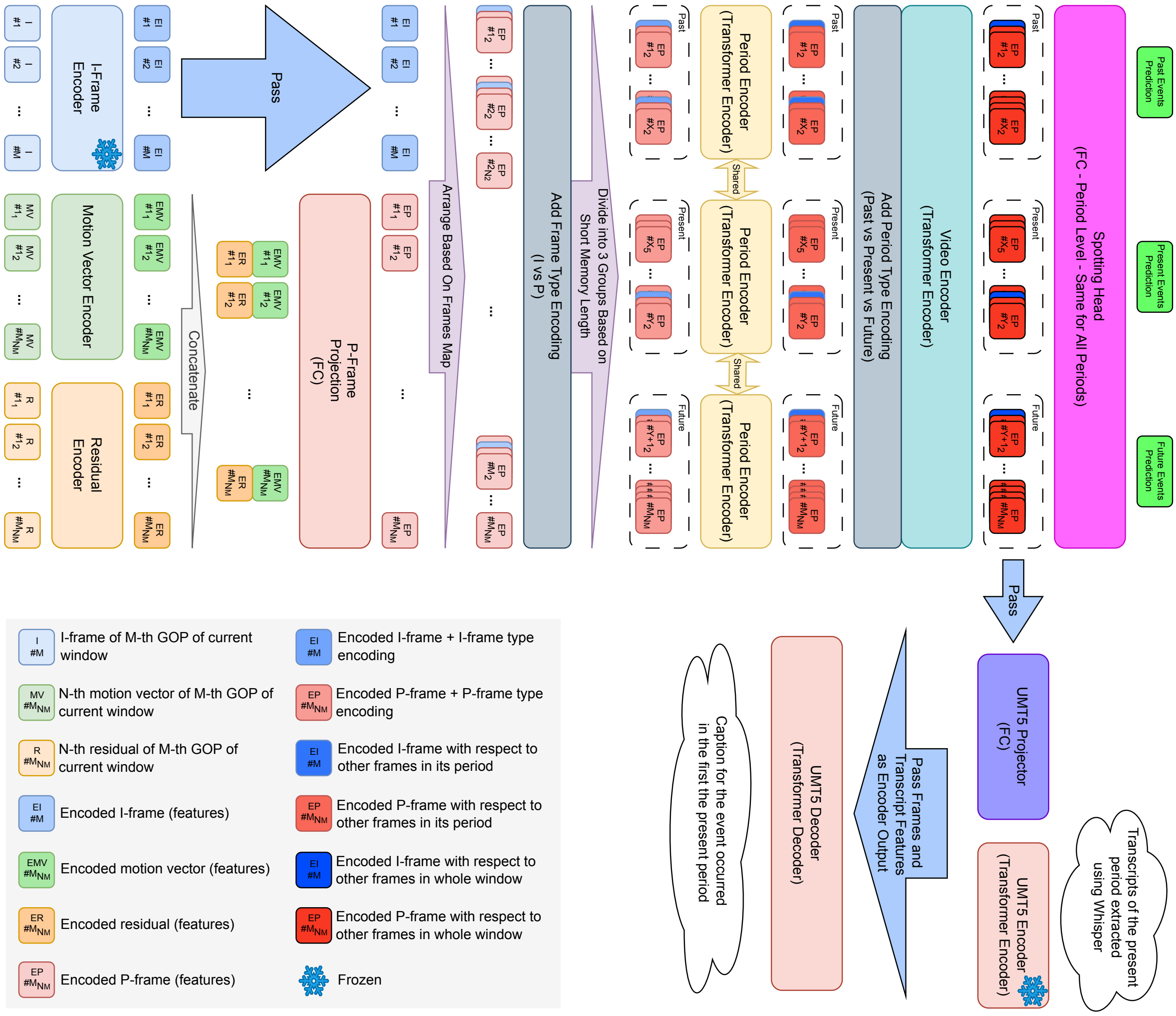
I-Frame Encoder
Motion Vector Encoder
Residual Encoder
Pass
Concatenate
P-Frame Projection (FC)
Arrange Based On Frames Map
Add Frame Type Encoding (I vs P)
Divide into 3 Groups Based on Short Memory Length
Past
Present
Future
Period Encoder (Transformer Encoder)
Shared
Add Period Type Encoding (Past vs Present vs Future)
Video Encoder (Transformer Encoder)
Spotting Head (FC - Period Level - Same for All Periods)
Past Events Prediction
Present Events Prediction
Future Events Prediction
Pass
UMT5 Projector (FC)
UMT5 Encoder (Transformer Encoder)
Transcripts of the present period extracted using Whisper
Pass Frames and Transcript Features as Encoder Output
UMT5 Decoder (Transformer Decoder)
Caption for the event occurred in the first the present period
I-frame of M-th GOP of current window
N-th motion vector of M-th GOP of current window
N-th residual of M-th GOP of current window
Encoded I-frame (features)
Encoded motion vector (features)
Encoded residual (features)
Encoded P-frame (features)
Encoded I-frame + I-frame type encoding
Encoded P-frame + P-frame type encoding
Encoded I-frame with respect to other frames in its period
Encoded P-frame with respect to other frames in its period
Encoded I-frame with respect to other frames in whole window
Encoded P-frame with respect to other frames in whole window
Frozen

Figure 3: Architecture overview. The process involves multiple stages: Encoding I-frames, motion vectors, and residuals separately; Encoding the past, present, and future periods using a common transformer encoder; Encoding the entire chunk and predicting events individually; and, if any event occurs, generating captions using the encoded frames of the entire chunk and transcript features from the present period.

motion vectors are more sensitive to noise, which can be damaged through .jpg compression.

3.2.3 Extract I-frames Features

We use the pooler output of the CLIP-vit-base-patch32 model (Radford et al., 2021), available on HuggingFace (Wolf et al., 2020). Given that soccer game frames share many common elements, such as the ball, players, and pitch, we selected the CLIP model over ResNet (He et al., 2015). CLIP is more likely to provide well-distinguishable features for different frames of a soccer game due to its training on image-caption pairs, which consider the relative positions of entities within an image—important for generating captions. In contrast, ResNet is trained solely to predict the presence of a class in an image. To expedite the training process, we save the extracted features to disk.

3.2.4 Extract Transcripts Features

Since the videos are multilingual, we extract their transcripts using the Whisper-large-v3 model (Radford et al., 2022), which can detect the language of the input audio and perform automatic speech recognition (ASR). We utilize the 're-turn_timestamps' argument of this model to obtain the timestamps for transcript chunks. These timestamps are then used to align the transcript chunks with the corresponding video frames. To extract features from the transcripts, we first divide them into chunks and then use the UMT5 encoder (Chung et al., 2023) , a multilingual version of T5. As with the video features, we save the extracted transcript features to expedite the training process.

3.2.5 Video Encoder

The video encoder has three stages:

1. Encode frames individually: I-frame features are pre-extracted. Motion vectors and residuals are encoded using two separate ResNet-18 models as they can be processed using lightweight neural networks. Each frame in the current chunk is encoded using its respective encoder. The motion vectors and residuals of a P-frame are concatenated and projected to match the I-frame feature size using a fully connected layer. Finally, frames are arranged according to their positions in the original video.

2. Encode periods: Frame type encodings, which are learnable and help the model differentiate between I-frames and P-frames, are added to the frames. The frames are then divided into three periods (past, present, future) of equal size (short memory length). A common period encoder, a transformer encoder, is used to encode the frames within each period concerning each other.

3. Encode the whole chunk: Period type encodings, which are learnable and help the model distinguish different periods, are added. A final transformer encoder encodes the frames of the entire chunk with respect to each other.

Note that all transformer encoders are standard, utilizing sinusoidal positional encoding.

3.2.6 Event Spotting

A fully connected layer with a sigmoid activation function is applied to the mean of the encoded frames of each period to predict whether an event has occurred in the current period. The present period is the primary focus, while the past and future periods aid in feature extraction. A softmax is not used because events can occur simultaneously (e.g., a penalty and a yellow card).

3.2.7 Decoder

The decoder uses the UMT5 architecture and pretrained weights to generate captions. The input consists of encoded video frames concatenated with transcript features from the present period in time dimension. The output is the caption corresponding to the present period. Multiple captions for different events within the same period are separated by the '@' symbol. The decoder is trained solely on positive samples.

4 Experiments

All experiments were conducted using a single NVIDIA A100-SXM4-80GB GPU.

4.1 Two-stage Training

We use a standard two-stage training process: first, training the encoder and the spotting head, then freezing them and fine-tuning the decoder. End-to-end training yielded worse results. In the first stage, a weighted random sampling strategy is used, and in the second stage, all positive samples are utilized.

4.2 Sampling Strategy

As mentioned earlier, the video is divided into chunks of length equal to the short memory length (a hyperparameter). The model is also provided with the previous and next chunks to utilize past and future information. The SoccerNet dataset is highly imbalanced, with far fewer chunks containing events than those without. To mitigate this, we assign higher weights to positive samples during sampling. The weights are calculated as follows:

$$ew_c = (E - E_c)/E \tag{2}$$

$$w_i = \sum_{c=0}^{C} y_{i_c} \times ew_c \tag{3}$$

where E is the total number of events in chunks, E_c is the number of events in class c, ew_c is the weight of event class c, y_{i_c} is the c-th class of the i-th sample's label, and w_i is i-th sample's weight.

And negative sample weights are calculated as follows:

$$nw = \sum_{i=0}^{N} w_i/NS \tag{4}$$

$$w_i = nw/NS \tag{5}$$

where nw is the total weight of the negative samples, which equals the total weight of positive samples, and NS is the number of negative samples. As a result, each epoch contains an equal number of positive and negative samples.

4.3 Loss Function

Despite this sampling strategy, the model still encounters more zeros than ones, as the number of ones in a label is significantly smaller. Therefore, we use the focal loss function to further address the dataset imbalance, experimenting with different values of alpha while keeping gamma fixed at 2.

4.4 Implementation Details

Our model is implemented using PyTorch. As mentioned, videos are resized to 224×224 resulting in motion vectors of size 56×56 and residuals of size 224×224. The residuals are resized to 56×56 to match the size of the motion vectors. For ablation studies, the model is trained on the SoccerNet training set and evaluated on the validation set. Each training epoch in this phase contains 18,000 samples. For comparison with state-of-the-art models, the model is trained on the combined training, validation, and test sets and evaluated on the challenge set using the Eval.ai platform (EvalAI). Each training epoch in this part has 29000 samples for the first training stage. Each training epoch in this phase contains 29,000 samples. The batch size is fixed at 16 for all experiments. Both the period encoder and the video encoder are 2-layer transformer encoders with a hidden size of 1536 and 32 attention heads. We use the AdamW (Loshchilov and Hutter, 2017) optimizer with $\beta_1 = 0.9$ and $\beta_1 = 0.999$. The cosine learning rate scheduler starts from $1e - 4$ and ends at 0.35 of the cosine cycle, yielding a final learning rate of $2e - 5$.

4.5 Evaluation Metrics

We use the SoccerNet challenge metrics: METEOR, BLEU@1, BLEU@2, BLEU@3, BLEU@4, ROUGE_L, CIDEr, recall, and precision. BLEU@N measures n-gram precision, METEOR assesses semantic accuracy, ROUGE evaluates word order, and CIDEr measures the degree to which the caption conveys key information.

4.6 Ablation Study

We conducted multiple ablation studies to evaluate the impact of various changes on model performance. The best model from each part was selected for further evaluation. In the tables, **bold = best**, <u>underline = second best</u>, TS = 1 stage training or 2 stage training, SML = short memory length, GOP = size of each group of pictures, Res = Use residuals, Trans = Use transcripts.

4.6.1 Residuals and Transcripts

The first experiment assessed the usefulness of residuals and transcripts. We tested all four possible combinations, and Table 1 shows the results.

As shown, using residuals degrades the model accuracy a lot, possibly because this reason: as residuals have visual structure as I-frames and the differences between successive frames in soccer videos are negligible, the residuals cannot add any information.

Using transcripts, however, can be beneficial, as indicated by the higher CIDEr score, which reflects better key point capture in generated captions. Other metrics can be ignored for comparison, as they are nearly identical and do not follow a consistent pattern.

4.6.2 Short Memory Length

We cannot use a short memory length greater than 60 (30s) as the challenge evaluation uses a 30s window around the ground truth. Table 2 shows the results for two different short memory lengths.

Most metrics improve with a short memory length of 60 as a larger chunk size provides the model with more information for caption generation.

4.6.3 GOP Size

As mentioned earlier, the H.264 codec typically uses a dynamic GOP (Group of Pictures) size, which adjusts according to the frequency of changes between successive frames. A higher frequency of changes results in a smaller GOP size, leading to more I-frames in areas with significant changes, and vice versa. To evaluate the impact of this, we conducted an experiment where we forced the FFMPEG package to set the GOP size to a short memory length. Table 3 presents the results.

The results demonstrate that forcing a short GOP size significantly degrades performance, particularly in the CIDEr metric, which measures the alignment of key concepts. Thus, allowing FFMPEG to use a dynamic GOP size is beneficial for the model's performance.

4.6.4 Two Stage Training

We also investigated whether a two-stage training approach is more effective than training the model all at once. As shown in Table 4, two-stage training yields better results. This is because negative samples can negatively impact the decoder component of the model when trained simultaneously.

TS	SML	GOP	Res	Trans	α	B@1	B@2	B@3	B@4	M	R@L	C	recall	precision
2	60	auto	✗	✗	0.9	33.29	**27.50**	**24.11**	**21.63**	19.21	**26.40**	18.38	82.95	**62.11**
2	60	auto	✗	✓	0.9	**33.30**	27.17	23.63	21.05	**19.87**	26.31	**19.23**	82.92	**62.11**
2	60	auto	✓	✗	0.9	30.24	25.13	22.28	20.09	18.00	22.48	13.61	**91.86**	60.30
2	60	auto	✓	✓	0.9	30.01	25.48	22.83	20.78	17.71	23.61	15.98	**91.86**	60.29

Table 1: Ablation study about impact of using residuals and transcripts.

TS	SML	GOP	Res	Trans	α	B@1	B@2	B@3	B@4	M	R@L	C	recall	precision
2	60	auto	✗	✓	0.9	**33.30**	27.17	23.63	21.05	**19.87**	**26.31**	**19.23**	82.92	**62.11**
2	40	auto	✗	✓	0.9	32.33	**27.22**	**24.17**	**21.85**	19.11	25.16	17.03	**91.61**	59.44

Table 2: Ablation study about impact of different length of short memory.

4.6.5 Focal Loss Alpha

We experimented with three different values of the alpha parameter in focal loss. The results indicate that a higher alpha increases recall at the cost of lower precision, and vice versa. According to Table 5 the optimal alpha value is 0.6. The experiments reveal a strong correlation between precision and generation metrics.

4.6.6 Generation Method

Initially, we observed that samples with multiple events often resulted in empty captions. We examined whether filtering these out would be beneficial. Additionally, we experimented with different generation strategies, including beam search and top-k+top-p sampling. Table 6 shows that the best configuration is to use greedy generation while ignoring blank captions.

4.7 Comparison with RGB

To assess the utility of using compressed videos, we compared our proposed model with an RGB variant where original RGB frames were used instead of I/P-frames. Results in Table 7 show that our proposed method achieves competitive results compared to its RGB variant.

Moreover, to understand the impact of using compressed videos on speed and GPU memory consumption, we conducted another comparison against the RGB variant. The results in Table 8 indicate approximately $5.4\times$ faster training speed and $5.1\times$ lower GPU memory usage, as well as $4.7\times$ faster inference speed and $7.8\times$ lower GPU memory usage compared to the RGB-based approach, which is significant. These results were obtained using the PyTorch profiler.

4.8 Comparison with SOTA methods

We evaluated our proposed model against state-of-the-art (SOTA) models on the challenge set of the SoccerNet dataset. The evaluation was conducted using the best configuration, which includes two-stage training, a short memory length of 60, auto GOP size, no residuals, inclusion of transcripts, a focal loss alpha of 0.6, and greedy generation while ignoring blank captions.

Table 9 presents the results, where our model ranks second in almost all metrics and exhibits performance comparable to SOTA models. Once again, a positive correlation between precision and generation metrics is evident. These results suggest that compressed video retains most of the necessary information with minimal redundancy.

4.9 Qualitative Results

As discussed in the appendix, Table 10 provides examples comparing the model's output to the ground truth, highlighting the model's bias towards corner events and events with no comments. Similar results are shown in Figure 4, which presents the confusion matrix. Further examples, along with frames from the time intervals used for predictions, are shown in Figures 5, 6, 7, and 8.

5 Conclusion

In this paper, we introduce a CNN-Transformer architecture for dense video captioning using compressed videos for the first time. The experiments demonstrate that our proposed model not only achieves competitive performance with SOTA models but also significantly reduces GPU memory usage and improves processing speed. This suggests that compressed videos could potentially become the standard for video processing, replacing traditional RGB frames processing.

TS	SML	GOP	Res	Trans	α	B@1	B@2	B@3	B@4	M	R@L	C	recall	precision
2	60	auto	✗	✓	0.9	**33.30**	**27.17**	23.63	21.05	**19.87**	**26.31**	**19.23**	82.92	**62.11**
2	60	60	✗	✓	0.9	29.66	26.22	**24.13**	**22.43**	18.21	24.33	11.91	**91.77**	56.86

Table 3: Ablation study about impact of different different GOP sizes.

TS	SML	GOP	Res	Trans	α	B@1	B@2	B@3	B@4	M	R@L	C	recall	precision
2	60	auto	✗	✓	0.9	**33.30**	**27.17**	**23.63**	**21.05**	19.87	**26.31**	**19.23**	82.92	**62.11**
1	60	auto	✗	✓	0.9	32.17	26.31	23.02	20.57	**20.24**	25.20	17.02	**93.60**	56.86

Table 4: Ablation study about impact of different training strategies.

TS	SML	GOP	Res	Trans	α	B@1	B@2	B@3	B@4	M	R@L	C	recall	precision
2	60	auto	✗	✓	0.9	33.30	27.17	23.63	21.05	19.87	26.31	19.23	**82.92**	62.11
2	60	auto	✗	✓	0.6	**36.70**	**30.12**	**26.24**	**23.32**	21.94	**32.48**	**28.77**	32.44	72.46
2	60	auto	✗	✓	0.4	34.66	28.33	24.60	21.80	**22.54**	32.25	28.54	23.99	**74.81**

Table 5: Ablation study about impact of different values of alpha in focal loss.

IB	NB	TOP_K	TOP_P	α	B@1	B@2	B@3	B@4	M	R@L	C	recall	precision
✗	1	0	0	0.9	33.30	27.17	23.63	21.05	19.87	26.31	19.23	82.92	**62.11**
✓	1	0	0	0.9	**33.89**	**27.65**	**24.06**	**21.43**	**21.53**	**28.49**	**21.33**	**82.92**	61.12
✓	5	0	0	0.9	28.40	22.38	19.32	17.12	20.18	25.99	15.64	82.92	61.10
✓	1	50	0.95	0.9	30.27	24.34	21.19	18.87	19.81	25.90	13.24	82.92	61.18

Table 6: Ablation study about impact of different generation strategies. (IB = Ignore blanks, NB = Number of beams)

RGB	TS	SML	GOP	Res	Trans	α	B@1	B@2	B@3	B@4	M	R@L	C	recall	precision
-	2	60	auto	✗	✓	0.9	33.30	27.17	23.63	21.05	19.87	26.31	19.23	82.92	**62.11**
✓	2	60	auto	-	-	0.9	**34.05**	**28.60**	**25.32**	**22.78**	**20.39**	**26.84**	**22.04**	**86.26**	60.79

Table 7: Accuracy comparison with RGB variant of the same architecture.

Video Encoder	Train Time (s)	Train GPU Mem. (TB)	Inference Time (s)	Inference GPU Mem. (TB)
Compressed Video	**15.13**	**1.5**	**8.67**	**0.29**
RGB	81.77	7.63	40.90	2.26

Table 8: Speed and GPU memory footprint comparison with RGB variant of the same architecture. The values shows the total time and total amount of GPU memory spent to encode the frames of all of the samples of a match (Encoder part only).

Team	B@1	B@2	B@3	B@4	M	R@L	C	recall	precision
OPPO	35.55	**31.03**	**28.13**	**25.65**	**26.66**	**32.33**	**69.73**	24.59	68.59
HZC	29.73	24.52	21.44	19.13	21.30	24.56	24.76	98.68	51.19
Baseline 2	30.01	24.80	21.74	19.44	21.25	24.65	25.68	98.68	51.21
justplay	29.83	24.68	21.66	19.38	21.20	24.34	25.89	98.68	50.99
aisoccer	29.53	24.42	21.42	19.15	21.02	24.31	23.72	98.63	50.83
Baseline 1	11.91	9.97	8.83	7.97	15.24	10.69	16.33	**98.97**	23.92
CVT5 (Ours)	**36.64**	29.60	25.55	22.59	22.17	32.02	26.84	42.16	**72.97**

Table 9: Comparison with state-of-the-art models (Leaderboard 2023) (Cioppa et al., 2023).

6 Limitations

The primary limitation of this work is that it was only evaluated on a single dataset. The results may vary significantly on other benchmarks, as events in a soccer game differ greatly from, for example, cooking events. It would be beneficial to evaluate the model on additional datasets.

Another limitation is that the quality of generated captions is heavily influenced by the low precision in event detection. This issue arises for several reasons: 1. Low precision means the model may predict an event where there is none, leading to incorrect captions. 2. Low precision also makes it difficult for the encoder to distinguish between events, complicating the task of generating distinct captions for different events. 3. The SoccerNet dataset has a highly imbalanced distribution of events, where the model's outputs are often biased toward the majority class. As shown in Table 9 on the 2023 leaderboard, achieving a certain accuracy level is easy, but improving beyond that is challenging due to the difficulty in accurately predicting minority class events. Future work could focus on strategies to better learn and predict minority classes.

References

Ali Abdari, Pouria Amirjan, and Azadeh Mansouri. 2019. Action recognition in compressed domain using residual information. In *2019 4th International Conference on Pattern Recognition and Image Analysis (IPRIA)*, pages 130–134.

Barak Battash, Haim Barad, Hanlin Tang, and Amit Bleiweiss. 2020. Mimic the raw domain: Accelerating action recognition in the compressed domain. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2926–2934.

Aaron Chadha, Alhabib Abbas, and Yiannis Andreopoulos. 2017. Compressed-domain video classification with deep neural networks: "there's way too much information to decode the matrix". In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1832–1836.

Hyung Won Chung, Xavier Garcia, Adam Roberts, Yi Tay, Orhan Firat, Sharan Narang, and Noah Constant. 2023. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining. In *The Eleventh International Conference on Learning Representations*.

Anthony Cioppa, Silvio Giancola, Vladimir Somers, Floriane Magera, Xin Zhou, Hassan Mkhallati, Adrien Deliège, Jan Held, Carlos Hinojosa, Amir M. Mansourian, Pierre Miralles, Olivier Barnich, Christophe De Vleeschouwer, Alexandre Alahi, Bernard Ghanem, Marc Van Droogenbroeck, Abdullah Kamal, Adrien Maglo, Albert Clapés, Amr Abdelaziz, Artur Xarles, Astrid Orcesi, Atom Scott, Bin Liu, Byoungkwon Lim, Chen Chen, Fabian Deuser, Feng Yan, Fufu Yu, Gal Shitrit, Guanshuo Wang, Gyusik Choi, Hankyul Kim, Hao Guo, Hasby Fahrudin, Hidenari Koguchi, Håkan Ardö, Ibrahim Salah, Ido Yerushalmy, Iftikar Muhammad, Ikuma Uchida, Ishay Be'ery, Jaonary Rabarisoa, Jeongae Lee, Jiajun Fu, Jianqin Yin, Jinghang Xu, Jongho Nang, Julien Denize, Junjie Li, Junpei Zhang, Juntae Kim, Kamil Synowiec, Kenji Kobayashi, Kexin Zhang, Konrad Habel, Kota Nakajima, Licheng Jiao, Lin Ma, Lizhi Wang, Luping Wang, Menglong Li, Mengying Zhou, Mohamed Nasr, Mohamed Abdelwahed, Mykola Liashuha, Nikolay Falaleev, Norbert Oswald, Qiong Jia, Quoc-Cuong Pham, Ran Song, Romain Hérault, Rui Peng, Ruilong Chen, Ruixuan Liu, Ruslan Baikulov, Ryuto Fukushima, Sergio Escalera, Seungcheon Lee, Shimin Chen, Shouhong Ding, Taiga Someya, Thomas B. Moeslund, Tianjiao Li, Wei Shen, Wei Zhang, Wei Li, Wei Dai, Weixin Luo, Wending Zhao, Wenjie Zhang, Xinquan Yang, Yanbiao Ma, Yeeun Joo, Yingsen Zeng, Yiyang Gan, Yongqiang Zhu, Yujie Zhong, Zheng Ruan, Zhiheng Li, Zhijian Huang, and Ziyu Meng. 2023. Soccernet 2023 challenges results. *Preprint*, arXiv:2309.06006.

EvalAI. Evalai: Evaluating state of the art in ai.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Lijun He, Miao Zhang, Sijin Zhang, Liejun Wang, and Fan Li. 2022. Mtrfn: Multiscale temporal receptive field network for compressed video action recognition at edge servers. *IEEE Internet of Things Journal*, 9(15):13965–13977.

Gabriel Huang, Bo Pang, Zhenhai Zhu, Clara Rivera, and Radu Soricut. 2020. Multimodal pretraining for dense video captioning. *CoRR*, abs/2011.11760.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. *CoRR*, abs/1705.00754.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. *CoRR*, abs/1906.03327.

Yue Ming, Lu Xiong, Xia Jia, Qingfang Zheng, Jiangwan Zhou, Fan Feng, and Nannan Hu. 2023. Frequency enhancement network for efficient compressed video action recognition. In *2023 IEEE In-*

ternational Conference on Image Processing (ICIP)*, pages 825–829.

Yue Ming, Jiangwan Zhou, Nannan Hu, Fan Feng, Panzi Zhao, Boyang Lyu, and Hui Yu. 2024. Action recognition in compressed domains: A survey. *Neurocomputing*, 577:127389.

Hassan Mkhallati, Anthony Cioppa, Silvio Giancola, Bernard Ghanem, and Marc Van Droogenbroeck. 2023. SoccerNet-caption: Dense video captioning for soccer broadcasts commentaries. abs/2304.04565.

Yuting Mou, Xinghao Jiang, Ke Xu, Tanfeng Sun, and Zepeng Wang. 2024. Compressed video action recognition with dual-stream and dual-modal transformer. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(5):3299–3312.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *arXiv preprint*.

Yaojie Shen. 2023. Github - acherstyx/compressed-video-reader: A video reader for extracting motion vectors and residuals from encoded h.264 videos.

Suramya Tomar. 2006. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Jue Wang and Lorenzo Torresani. 2022. Deformable video transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14033–14042.

Teng Wang, Jinrui Zhang, Feng Zheng, Wenhao Jiang, Ran Cheng, and Ping Luo. 2023. Learning grounded vision-language representation for versatile understanding in untrimmed videos. *CoRR*, abs/2303.06378.

Teng Wang, Ruimao Zhang, Zhichao Lu, Feng Zheng, Ran Cheng, and Ping Luo. 2021. End-to-end dense video captioning with parallel decoding. *CoRR*, abs/2108.07781.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. pages 38–45. Association for Computational Linguistics.

Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. 2017. Compressed video action recognition. *CoRR*, abs/1712.00636.

A. Yang, A. Nagrani, P. Seo, A. Miech, J. Pont-Tuset, I. Laptev, J. Sivic, and C. Schmid. 2023. Vid2seq: Large-scale pretraining of a visual language model for dense video captioning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10714–10726, Los Alamitos, CA, USA. IEEE Computer Society.

Rowan Zellers, Jiasen Lu, Ximing Lu, Youngjae Yu, Yanpeng Zhao, Mohammadreza Salehi, Aditya Kusupati, Jack Hessel, Ali Farhadi, and Yejin Choi. 2022. MERLOT reserve: Neural script knowledge through vision and language and sound. *CoRR*, abs/2201.02639.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2020. Deformable DETR: deformable transformers for end-to-end object detection. *CoRR*, abs/2010.04159.

A Qualitative Results

Table 10 shows examples of the model's output compared to the ground truth. The model's bias towards corner events and events with no comments is evident in these examples. Similar results are shown in Figure 4 which presents the confusion matrix of the model's output and ground truth. Additionally, when the spotting head is uncertain about the occurrence of an event, the captioner module often generates incorrect captions. Occasional language modeling errors, such as repetitive tokens, are also observed.

Examples of the model's output, along with the frames from the time intervals used for predictions, are shown in Figures 5, 6, 7, and 8. The frame rate is 4 times lower than the model's input sampling rate, meaning one frame is shown every 2 seconds. The images start from the top left and end at the bottom right. As observed in these examples, even when the model makes a mistake, the images closely resemble the generated descriptions, meaning that these examples are challenging. However, it can also be seen that the model does not pay enough attention to small details within the frames.

Confidence	GT Event	G Event	GT Caption	G Caption
0.88	corner	corner	[PLAYER] ([TEAM]) attempts to find a teammate with the corner, but the effort is snuffed out by the goalkeeper.	[PLAYER] ([TEAM]) takes the corner, but it's intercepted by the defender.
0.52	-	corner	[PLAYER] ([TEAM]) smashes the ball towards goal from the edge of the penalty box, but it fails to bother the goalkeeper as it hits the defensive wall.	[PLAYER] ([TEAM]) takes the corner, but it's intercepted by the defender.
0.59	soccer-ball	soccer-ball	Goal! [PLAYER] feeds [PLAYER] ([TEAM]), who taps the ball into an empty net. He makes it 1:0.	[PLAYER] ([TEAM]) is unable to feed a low pass into the path of one of his teammates. The ball goes out of play and [TEAM] will have a goal kick.
0.81	corner	corner	[PLAYER] ([TEAM]) launches a powerful cross from the corner into the box, but the ball is intercepted by the defender. The linesman makes the right call and [TEAM] will have a corner.	[PLAYER] ([TEAM]) takes the corner, but it's intercepted by the defender.
0.89	substitution	substitution	The referee allows time for a substitution. [PLAYER] will be replaced by [PLAYER] ([TEAM]).	[COACH] has decided to introduce fresh legs, with [PLAYER] ([TEAM]) replacing [PLAYER].
0.66	-	-	[PLAYER] ([TEAM]) is having a lively performance. He is causing problems and wanting the ball at every opportunity.	[PLAYER] ([TEAM]) sends a cross into the box, [TEAM]]]] ([TEAM]]), [TEAM]] will have a chance to score from a free kick. [TEAM] will have a chance
0.54	-	-	[PLAYER] ([TEAM]) whips the ball in from the long-range free kick, but the first man gets it clear.	[PLAYER] ([TEAM]) sends a pass into the box, [PLAYER] ([TEAM]) is REFEREE]REFEREE] REFEREE]REFEREE] REFEREE]REFEREE] REFEREE]REFEREE]
0.68	-	-	[PLAYER] ([TEAM]) picks up a rebound inside the penalty area and drills a shot to the bottom right corner, but is denied by a reflex save from [PLAYER]. [TEAM] have been awarded a corner kick. The referee and one of his assistants both point at the corner flag.	[PLAYER] ([TEAM]) sends a cross into the box, but the opposition's defence clears the ball away to eliminate the danger. The referee blows his whistle, [TEAM] are awarded a corner kick
0.58	-	-	[PLAYER] ([TEAM]) receives a pass and decides to smash the ball from long range, but his poor effort sails high over the bar.	[PLAYER] ([TEAM]) sends a cross into the box, but [PLAYER] comes off his line to gather the ball.
0.86	substitution	substitution	The manager makes a subsitution with [PLAYER] ([TEAM]) coming on for [PLAYER].	[COACH] has decided to introduce fresh legs, with [PLAYER] ([TEAM]) replacing [PLAYER].
0.67	-	-	[PLAYER] ([TEAM]) was trying to get to the ball but clattered into the legs of the opponent as well. [REFEREE] blows his whistle for an infringement. [TEAM] are awarded a free kick. Let's see what they create from this.	[PLAYER] ([TEAM]) sends a cross into the box, but the opposition's defence clears the ball away to eliminate the danger.
0.82	corner	corner	[PLAYER] ([TEAM]) takes the resulting corner which is well defended.	[PLAYER] ([TEAM]) sends a cross into the box, but the opposition's defence clears the ball away to eliminate the danger. The referee and his assistant both point at the corner flag. [TEAM] will
0.69	-	corner	[PLAYER] ([TEAM]) whips the ball in, but it fails to reach any of his teammates as the opposition's defence averts the threat. The linesman points to the corner flag, [TEAM] are going to take it.	[PLAYER] ([TEAM]) sends a cross into the box, but the opposition's defence clears the ball away to eliminate the danger. The referee and his assistant both point at the corner flag. [TEAM] will
0.85	corner	corner	[PLAYER] ([TEAM]) swings in a cross from the corner, but [PLAYER] reads it well and gathers the ball.	[PLAYER] ([TEAM]) sends a cross into the box, but the opposition's defence clears the ball to safety. The ball goes out of play and [TEAM] have been awarded a corner kick.
0.53	-	-	He should have done better. [PLAYER] ([TEAM]) is afforded space to connect with a [PLAYER] cross, but his header from the centre of the box flies well wide of the left post.	[PLAYER] ([TEAM]) sends a cross into the box, but the opposition's defence clears the ball to safety.
0.68	corner	-	[PLAYER] ([TEAM]) goes over to take the corner kick and it looks like he will send the ball into the penalty box.	[PLAYER] ([TEAM]) sends a cross into the box, but the opposition's defence clears the ball away to eliminate the danger. The referee and his assistant both point at the corner flag. [TEAM] will
0.54	-	-	What a goal-scoring opportunity! [PLAYER] ([TEAM]) finds some space inside the box and gets in a strike, but the shot is brilliantly blocked by one of the defending players sliding in.	[PLAYER] ([TEAM]) sends a pass into the box, [PLAYER] ([TEAM]) is REFEREE]REFEREE] REFEREE]REFEREE] REFEREE]REFEREE] REFEREE]REFEREE]
0.50	-	-	[PLAYER] ([TEAM]) produces a lovely ball into the penalty area but the defender manages to intercept and comfortably averts the danger.	[PLAYER] ([TEAM]) sends a pass into the box, [PLAYER] ([TEAM]) is REFEREE]REFEREE] REFEREE]REFEREE] REFEREE]REFEREE] REFEREE]REFEREE]
0.50	-	-	[PLAYER] ([TEAM]) attempts to slip the ball through the defence, but is unable to find any of his teammates.	[PLAYER] ([TEAM]) sends a cross into the box, but [PLAYER] comes off his line to gather the ball.
0.58	corner	corner	[PLAYER] ([TEAM]) swings in the corner kick, but one of the defenders leaps highest to head the ball away.	[PLAYER] ([TEAM]) takes the corner kick and sends a lovely ball into the penalty area, but the opposition's defence is ready and knocks the ball to safety.
	penalty		Poor challenge! [PLAYER] ([TEAM]) is penalised for tripping and [REFEREE] blows his whistle. PENALTY to [TEAM]! Great chance to score.	
0.58	y-card	-	[PLAYER] ([TEAM]) commits a foul and is shown a yellow card without any hesitation from the referee.	[PLAYER] ([TEAM]) is penalised for holding. [REFEREE] signals a set piece.
	soccer-ball		[PLAYER] ([TEAM]) sends [PLAYER] the wrong way and fires the penalty into the middle of the goal!	

Table 10: Some examples of the model's output compared to the ground truth

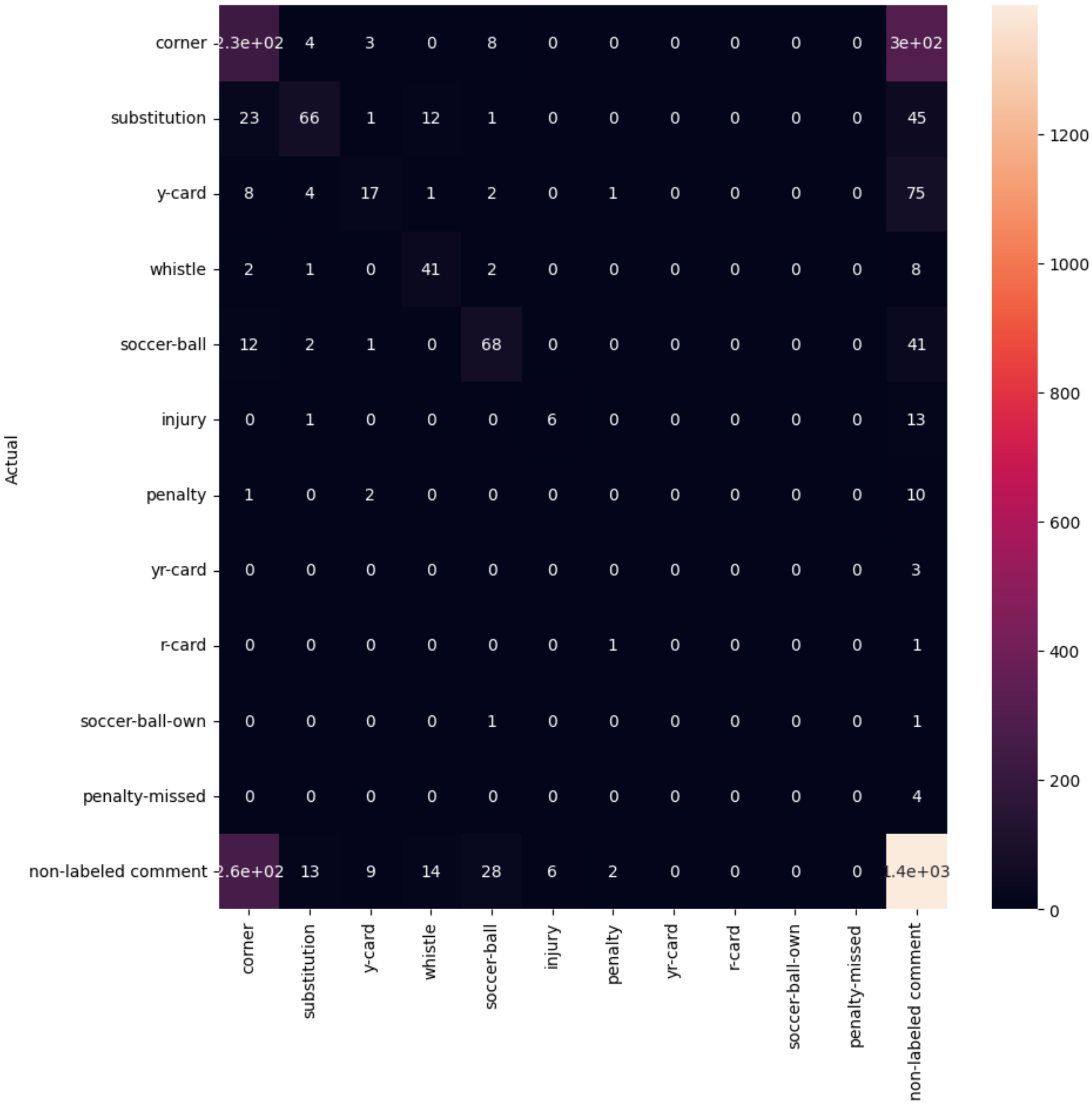

Figure 4: Confusion matrix on the SoccerNet validation set

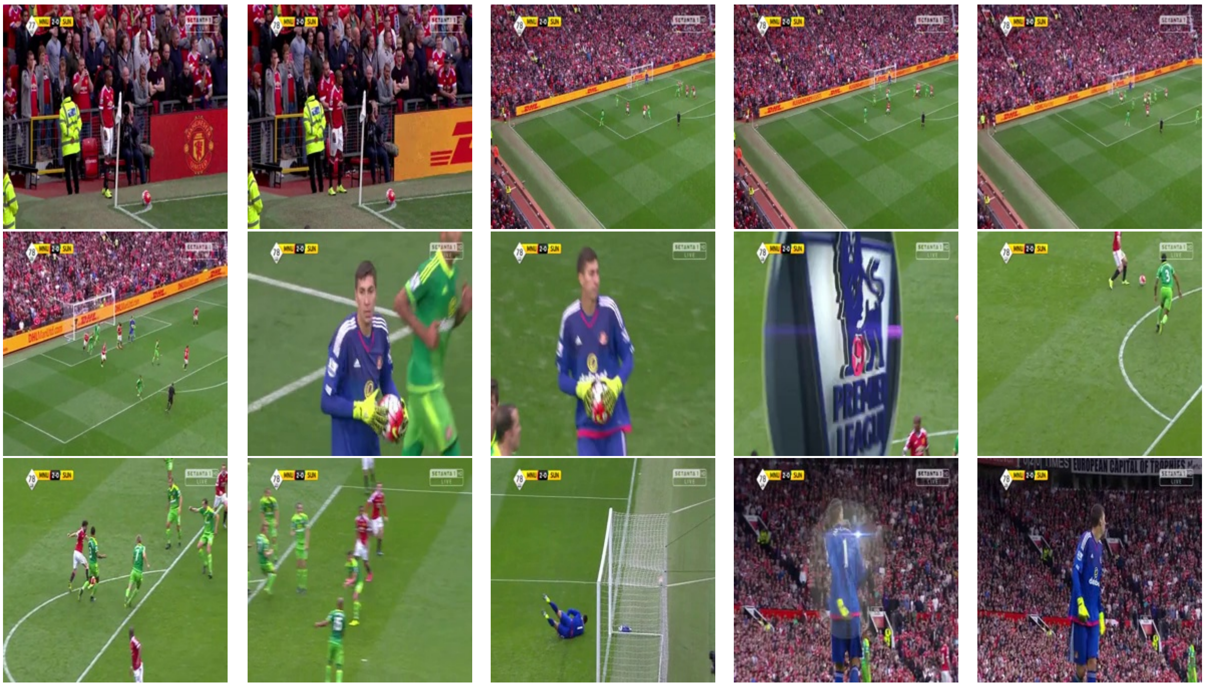

Figure 5: Model output: [PLAYER] ([TEAM]) takes the corner, but it's intercepted by the defender. Ground truth: [PLAYER] ([TEAM]) attempts to find a teammate with the corner, but the effort is snuffed out by the goalkeeper.

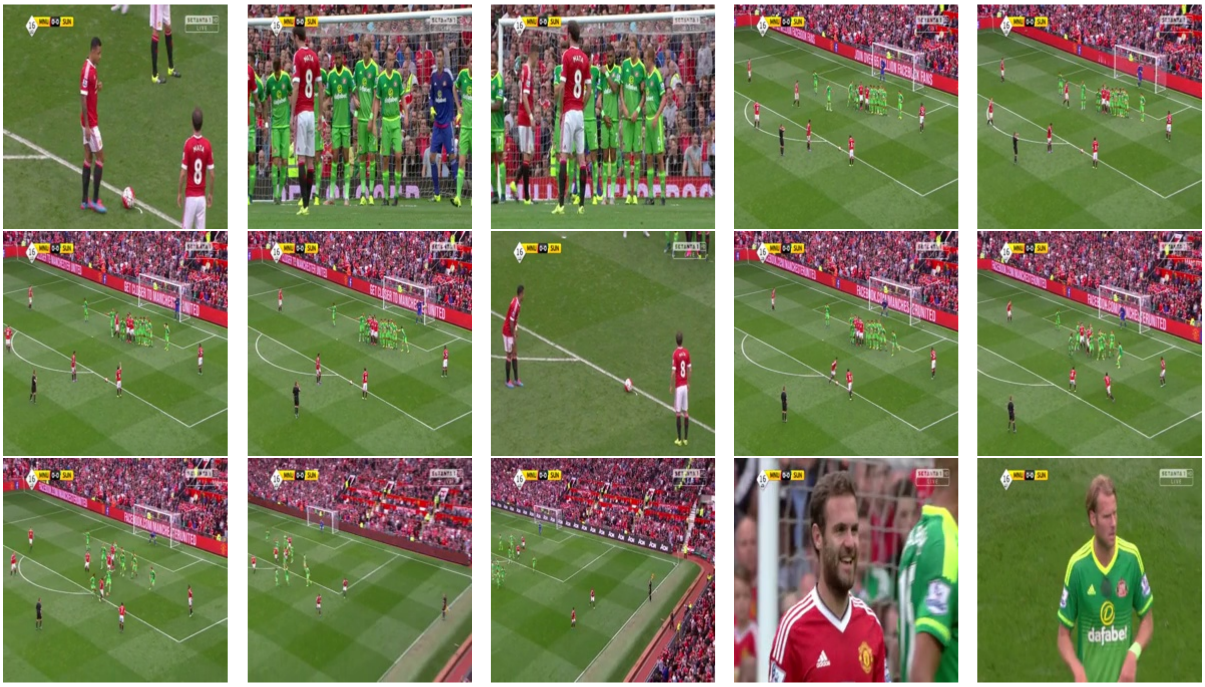

Figure 6: Model output: [PLAYER] ([TEAM]) takes the corner, but it's intercepted by the defender. Ground truth: [PLAYER] ([TEAM]) smashes the ball towards goal from the edge of the penalty box, but it fails to bother the goalkeeper as it hits the defensive wall.

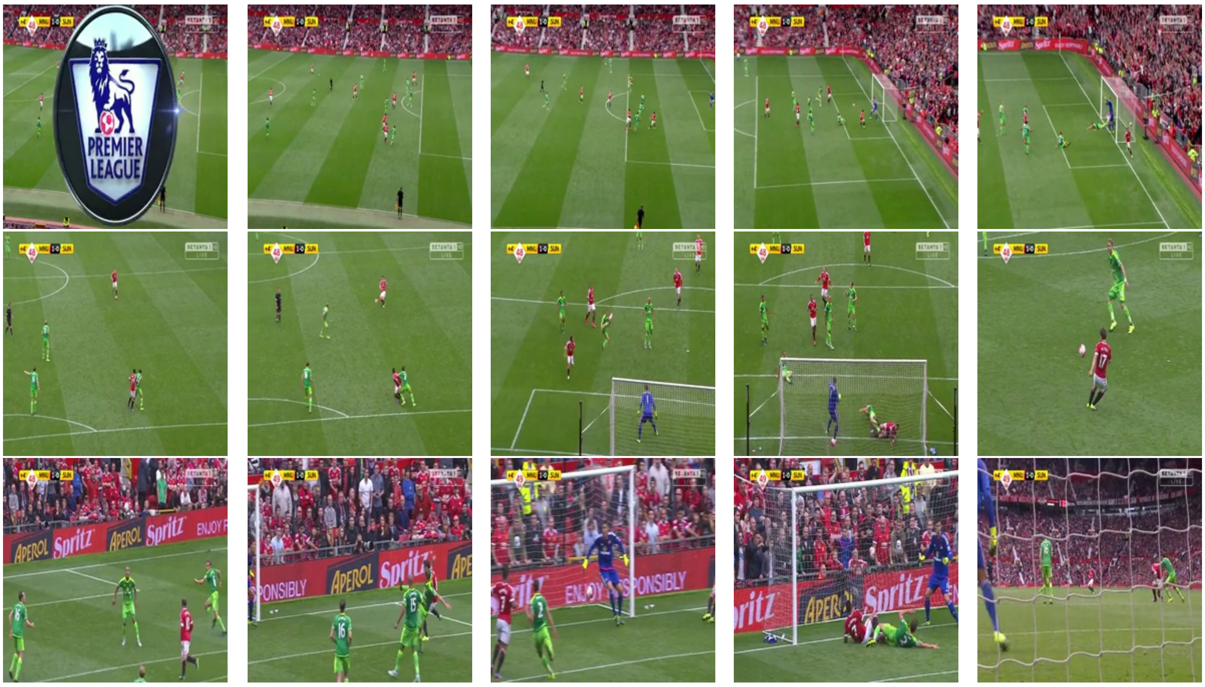

Figure 7: Model output: [PLAYER] ([TEAM]) is unable to feed a low pass into the path of one of his teammates. The ball goes out of play and [TEAM] will have a goal kick. Ground truth: Goal! [PLAYER] feeds [PLAYER] ([TEAM]), who taps the ball into an empty net. He makes it 1:0.

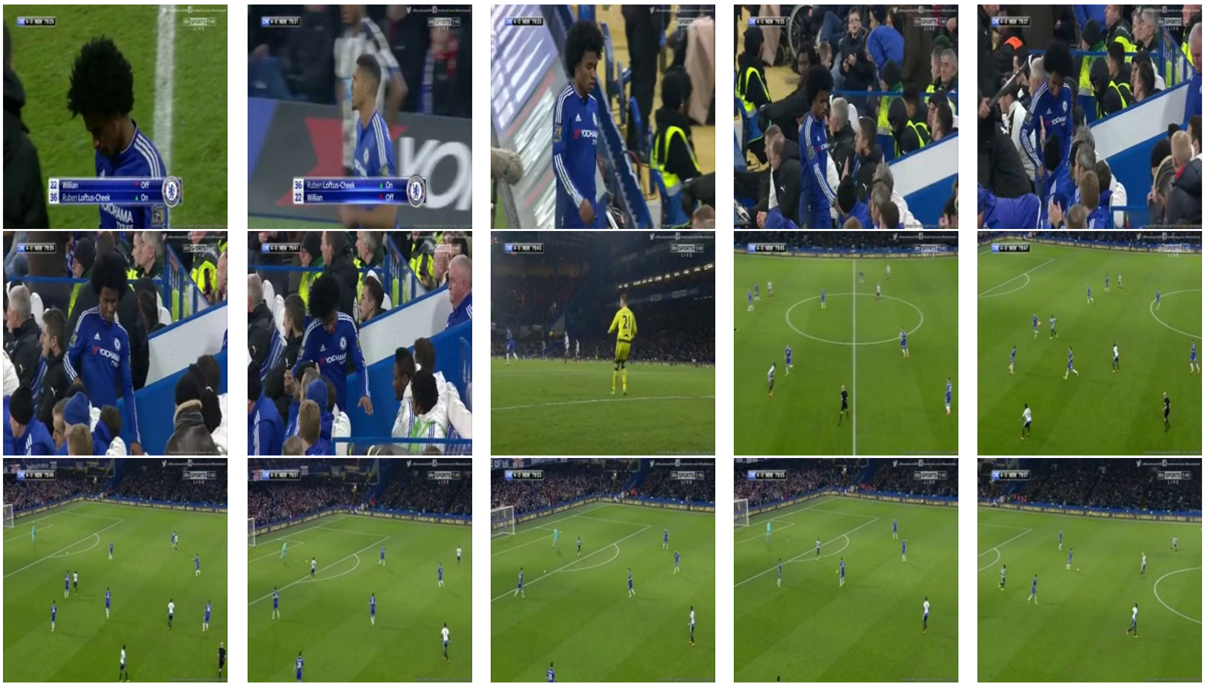

Figure 8: Model output: [COACH] has decided to introduce fresh legs, with [PLAYER] ([TEAM]) replacing [PLAYER]. Ground truth: The referee allows time for a substitution. [PLAYER] will be replaced by [PLAYER] ([TEAM]).