"*Previously on ...*" From Recaps to Story Summarization

SUPPLEMENTARY MATERIAL

Aditya Kumar Singh Dhruv Srivastava Makarand Tapaswi
CVIT, IIIT Hyderabad, India

https://katha-ai.github.io/projects/recap-story-summ/

We provide additional information and experimental results to complement the main paper submission. In this work, we propose a dataset *PlotSnap* consisting of two crime thriller TV shows with rich recaps and a new hierarchical model *TaleSumm* that selects important local story groups to form a representative story summary. We present details of PlotSnap in Appendix A. Appendix B extends on additional experimentation on PlotSnap as well as SumMe [8] and TVSum [29] followed by an extensive qualitative study. Finally, we conclude this supplement with a discussion of future research directions in Appendix C.

A. Dataset Details

We start with the essential step of creating soft labels for each shot and dialog of the episode, presented in Appendix A.1. We describe how recaps are used to generate binary labels using our novel shot-matching algorithm. Next, Appendix A.2 describes how label smoothing is performed. This is an essential step towards capturing the local sub-story in a better way. Further, Appendix A.3 comments on the different data split creation strategies used to evaluate baselines and our model's generalizability. Finally, to conclude Appendix A.4 presents detailed episode-level label reliability scores.

A.1. Shot Matching

We propose a novel shot-matching algorithm whose working principle involves frame-level similarities to obtain matches.

First, we compute frame-level embeddings using DenseNet169 [13], which were found to work better than models such as ResNet pre-trained on ImageNet [10, 24] based on a qualitative analysis. An example is shown in Fig. 1 where we see higher similarity between retrieved episode frames and the query frame from the recap.

Second, we discuss how these embeddings are used to obtain matches is detailed below.

Matching. For a given recap shot s in $\mathcal{R}_{n+1}$ we compare it against multiple frames in the episode $\mathcal{E}_n$, and compute a matrix dot-product with appropriate normalization (*cosine similarity*) between respective frame representations of the recap and episode as illustrated in the toy-example of Fig. 2. We remove very dark or very bright frames, typical in poor lighting conditions or glares, to avoid spurious matches and noisy labels.

Figure 1. Retrieval results for Recap from Episode Frames with DenseNet (Top) v/s ResNet (bottom). We observe qualitatively that DenseNet is able to match to the correct frames from the episode more often.

Next, we choose a high threshold to identify matching frames (0.85 in our case after analysis) and fetch all the top matching frames along with their shot indices (the shot where the frame is sourced from). We compute a set union over all matched shots obtained by scoring similarities between the recap frame of shot s and denote this set as $\mathbb{S}_s$. In the example, we match 3 frames of a recap shot and identify several episode shots shown in the blue box with $\mathbb{S}_s$.

Weeding out spurious shot matches. The set $\mathbb{S}_s$ may contain shots beyond a typical shot thread pattern due to spurious matches. These need to be removed to prevent wrong importance scores from being obtained from the recap. To do this, we first find the best matching shot in the episode. We observe that taking top three matched frames for every recap frame results in strong matches. Subsequently, we pick the maximum similarity score for each unique shot matched to a recap frame. This allows us to accumulate the score for an episode shot if multiple frames of the episode shot match with frames of the recap shot. The shot that scores the highest (after summing up the scores) is considered the best-matched episode shot for recap shot s.

Next, we choose a window size of 21 (10 on either side) and include all shots in $\mathbb{S}_s$ that fall within this window to a new matched set, $\mathcal{N}_s$. This is motivated by the typical duration of a scene in a movie or TV episode (40-60 seconds) and an average shot duration of 2-3 seconds. We repeat this process until no more shots are added to the set $\mathcal{N}_s$ and discard the rest in $\mathbb{S}_s$. Thus, for a given shot from the recap, we obtain all matching shots in the episode that are localized to a certain region of high-scoring similarity (see Fig. 2). We repeat this process for all frames and shots from the recap.

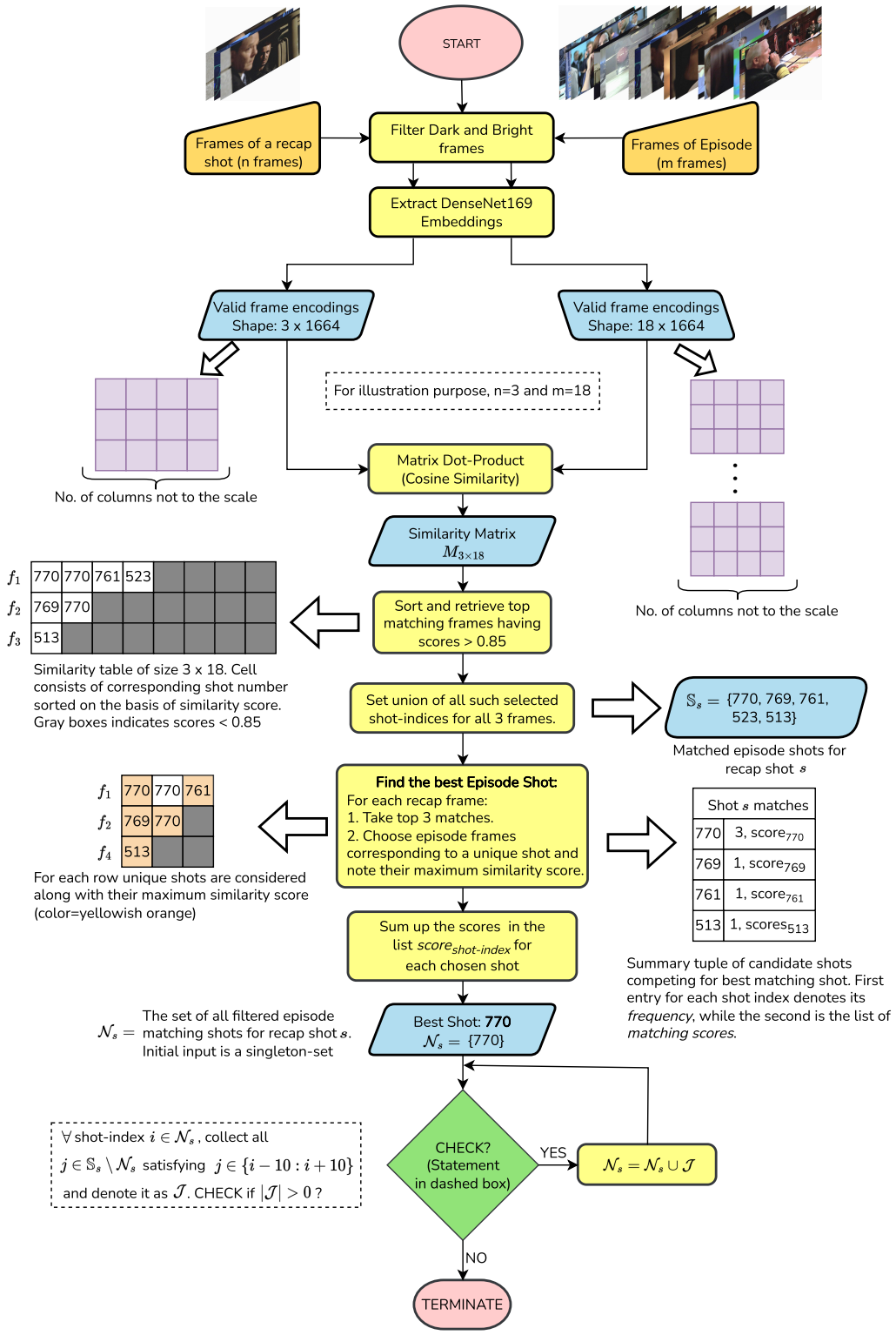

Figure 2. Flowchart for identifying shots from the episode that appear in a recap and can be used as weak labels for story summarization. The process involves identifying the list of high-scoring matching frames, indexing the shots, and then preventing spurious matches by looking for high-scoring matches within a bounded duration. The flowchart presents an example of the process used to identify the set of shots $\mathcal{N}_s$ from the episode that match to the recap shot.

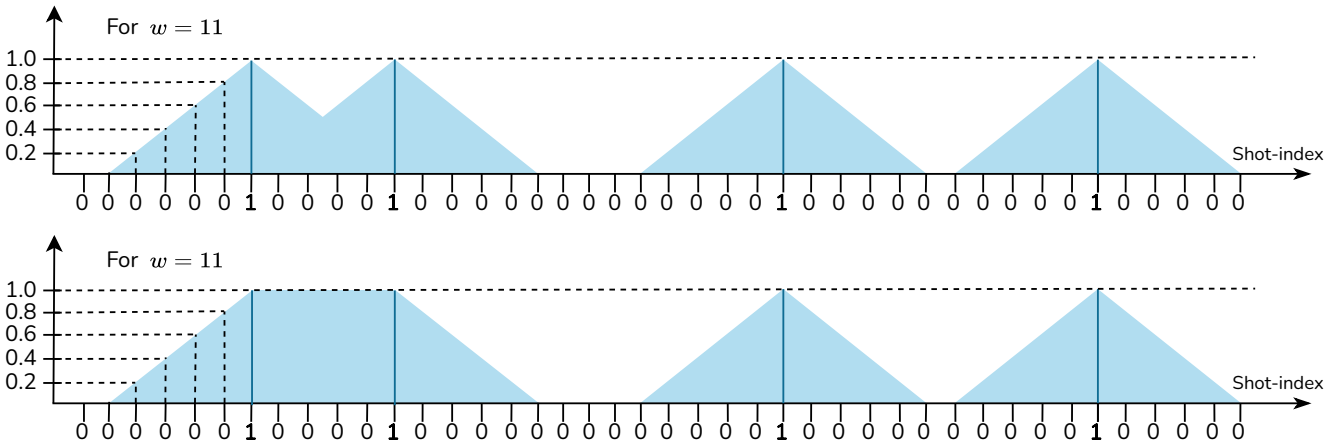

Figure 3. Triangle smoothing process. Here *x-axis* denotes binary labels derived from the *shot matching* process, while *y-axis* shows soft *importance scores* used to train our model. **Top:** The triangular filter is applied at each shot selected from the matching process. Scores of shots falling within the window are updated. **Bottom:** In the second step, we add shot importance derived potentially from multiple overlapping triangle filters. This typically happens when episode shots in close proximity are matched to the recap.

A.2. Label Smoothing

The intuition behind extending the recap matched shots obtained in the previous section is to include chunks of the sub-story that are important to the storyline. While a short recap (intended to bring back memories) only selects a few shots, a story summary should present the larger sub-story. Selecting only one shot in a thread [31] also adversely affects model training due to conflicting signals, as multiple shots with similar appearance can have opposite labels. Label smoothing solves both these issues.

Triangle Smoother. We hypothesize that the importance of shots neighboring a matched shot are usually quite high and use a simple *triangle smoother* to re-label the importance of shots. In particular, we slide a window of size w centering at positive labels and set the importance of neighboring shots according to height of the triangle. The above process is illustrated in Fig. 3 as the first step. In the second step, we add and clip the soft labels of strongly overlapping regions to prevent any score from going higher than 1. For the sake of simplicity, we used *triangle* smoothing, however, one could also use other filters.

We choose $w = 17$ by analyzing the spread of the shots and their importance scores and comparing them against a few episodes for which we manually annotated the story summaries.

Dialog labels. The above smoothing procedure generates soft labels for video shots. For dialog utterances, we simply import the score of the shot that encompasses the mid-timestamp of the dialog. The key assumption here is that the dialog utterance associated with the matched shot is also important.

A.3. Data Splits

For evaluation of our approach, PlotSnap is split into 4 types of splits as described below:

1. IntraCVT (Intra-Season 5 fold cross-val-test) represents 5 different non-overlapping splits from 7 seasons of *24* (Season 2 to 8). *Intra-season* means episodes from each season are present in the train/val/test splits. For example, split-1 uses 5 episodes from the end of each season for val and test (2, 3 respectively). Likewise, Split-5 (the fifth fold) uses episodes from the beginning of the season in val/test. We observe that Split-5 is harder.
2. X-Season (Cross-Season 7 fold cross-val-test) involves 7 non-overlapping splits with one season entirely kept for testing (from Season 2 to 8), while the train and val use 18 and 5 episodes respectively from each season. This strategy is used to test for the generalization of our model on different seasons of the same series, *24*.
3. X-Series (Cross-Series) split includes 8 seasons (seasons 2 to 9) from *24*in train/val (19/4) and 2 seasons (seasons 2 and 3) from *PB* for the test. This split is designed to check the effectiveness of our model across TV series.
4. *Multiple labels split* consists of a single non-overlapping train/val/test split with 126/18/17 episodes from *24*, respectively. We collect story summary annotations from 3 sources for the test set (recap: GT, fan website: F, and human annotation: H). This split is used for comparison against labels from *Fandom* or *Human* annotations.

A.4. Label Consistency

As discussed in Tab. 8 (of the main paper), we show the agreement between story summary labels obtained from different sources. Consistency is evaluated on 3 kinds of labels: *From recap* (GT), *Fandom* (F) and *Human* (H). We

season	episode	Video			Dialog		
		Cα	PF_1	Fκ	Cα	PF_1	Fκ
S02	E21	0.978	0.51	0.337	0.964	0.51	0.335
S02	E23	0.937	0.407	0.135	0.966	0.472	0.182
S03	E20	0.959	0.684	0.595	0.939	0.715	0.629
S03	E22	0.907	0.568	0.422	0.861	0.553	0.409
S04	E20	0.954	0.671	0.47	0.949	0.716	0.568
S04	E21	0.981	0.618	0.405	0.983	0.504	0.286
S05	E21	0.886	0.521	0.27	0.872	0.497	0.249
S05	E22	0.994	0.573	0.282	0.991	0.611	0.334
S06	E20	0.986	0.639	0.432	0.975	0.689	0.534
S06	E21	0.979	0.645	0.437	0.96	0.618	0.439
S06	E22	0.965	0.557	0.297	0.929	0.632	0.406
S06	E23	0.981	0.496	0.263	0.986	0.475	0.206
S07	E20	0.942	0.684	0.451	0.968	0.67	0.382
S07	E22	0.849	0.531	0.349	0.892	0.54	0.334
S07	E23	0.993	0.525	0.215	0.988	0.541	0.235
S08	E21	0.233	0.68	0.527	0.715	0.667	0.518
S08	E22	0.968	0.685	0.507	0.948	0.728	0.538
Average		0.911	0.588	0.376	0.934	0.596	0.387

Table 1. Detailed overview of reliability scores for 17 episodes from *24* (test set of *multiple labels split*) with the last row showing the average across all episodes. *Si* and *Ej* stands for Season i Episode j, Cα for Cronbach's α, PF_1 for Pairwise F_1, and Fκ for Fleiss' κ.

assess the consistency of labels via Cronbach's α, Pairwise F_1, and Fleiss' κ statistics.

Tab. 1 expands on the individual scores obtained for each of the 17 episodes. We observe that Cronbach's α is consistently high, while Fleiss's κ varies typically between 0.2-0.5 indicating fair to moderate agreement.

B. Experiments and Results

In this section we expand our implementation details (Appendix B.1) and present additional details of the backbones used for feature extraction (Appendix B.2). Extensive ablation studies with regard to feature combinations and model hyperparameters are presented in Appendix B.3.

In Appendix B.4, we present details of the modifications that need to be made to adapt SoTA baselines such as MSVA [7], PGL-SUM [4], PreSumm [20], and A2Summ [9] for comparison with TaleSumm. Appendix B.5 details how we adapted our model for SumMe and TVSum, along with comprehensive hyperparameter particulars. To conclude, Appendix B.6 shows extended qualitative analysis, as in the main paper, on three other episodes (S06E20, S07E22, and S05E21).

B.1. Implementation Details

Visual features. We first segment the episode into shots using [34]. We adopt 3 specialized backbones (for their combined effective performance; shown in Tab. 2): (i) DenseNet169 [13] pre-trained on ImageNet [24], SVHN [35], and CIFAR [17] for object semantics; (ii) MViT [6] pre-trained on Kinetics-400 [14] for action information; and (iii) OpenAI CLIP [23], pre-trained on 4M image-text pairs, for semantic information.

Utterance features. We adapt RoBERTa-large [38] originally pretrained on the reunion of five datasets: (i) BookCorpus [37], a dataset consisting of 11,038 unpublished books; (ii) English Wikipedia [3] (excluding lists, tables and headers); (iii) CC-News [1], a dataset containing 63 millions English news articles crawled between September 2016 and February 2019; (iv) OpenWebText [2], an opensource recreation of the WebText dataset used to train GPT-2; and (v) Stories [32] a dataset containing a subset of CommonCrawl data filtered to match the story-like style of Winograd schemas [18]. Together these datasets weigh 160GB of text.

Given dialogs from the episode, our fine-tuning objective is to predict the important dialogs. We extract word/token-level representations ($\mathbf{w}$) from *finetuned* (but frozen) RoBERTa-large ($\phi_{\mathcal{U}}^{\text{FT}}$) for the task of dialog story summarization.

Frame sampling strategy. We *randomly* sample up to 25 frames per shot during training as a form of data augmentation. During inference, we use *uniform* sampling. We used fourier position embeddings $\mathbf{E}_j^{\mathcal{S}}$ for indexing video frames.

Architecture details. We experiment with the number of layers for ST, $H_{\mathsf{ST}} \in [1:3]$ and ET, $H_{\mathsf{ET}} \in [1:9]$, and find $H_{\mathsf{ST}}=1$ and $H_{\mathsf{ET}}=6$ to work best. Except the number of layers, ST and ET have the same configuration: 8 attention heads and $D=128$. Appropriate padding and masking is used to create batches. We compare multiple local story group sizes $n_g \in \{5:30:5\}$ and find $n_g=20$ to work best.

Training details. Our model is trained on 4 RTX-2080 Ti GPUs for a maximum of 65 epochs with a batch size of 4 (*i.e.* 4 entire episodes – each GPU handling one episode). We adopt the AdamW optimizer [22] with parameters: learning rate$=10^{-4}$, weight decay$=10^{-3}$. We use OneCycleLR [27] as learning rate scheduler with max lr$=10^{-3}$, and multiple dropouts [11]: 0.1 for projection to 128 dim inside video and utterance encoder; 0.2 for attention layers; and 0.2 for the classification head. The hyperparameters are tuned for best performance on validation set.

B.2. Feature Extraction

Prior to feature extraction, setting an appropriate fps for every video is important to trade off between capturing all aspects of the video while keeping computational load low. We find 8 fps to be a good balance between the two.

Visual Feature Backbones We present the details for three backbones capturing different aspects of a video.

DenseNet169 $\mathbf{f}^1$. Feature extraction of salient objects/person in each frame is of utmost priority and is achieved through DenseNet169 [13] pretrained on ImageNet [24], SVHN [35], and CIFAR [17]. We consider the frozen backbone without the linear classification head to obtain flattened features, $\mathbf{f}^1 \in \mathbb{R}^{1664}$. Before feeding the images, we apply a few preprocessing steps to sub-sample raw images.

1. Frames are resized to 256×256 resolution along with center cropping.
2. RGB pixel values are scaled to $[0, 1]$ followed by mean and standard deviation normalization.

We use the architecture as well as parameters from PyTorch Hub[1], version `pytorch/vision:v0.10.0`.

MVIT $\mathbf{f}^2$. Beyond objects/person, their actions too affect the importance of a shot, and hence having them serves the purpose of representing a shot from a different perspective. For this, we use MViT [6] pretrained on Kinetics-400 [14]. We feed the original video with some pre-processing as explained above to obtain feature embeddings, $\mathbf{f}^2 \in \mathbb{R}^{768}$. With a window-size=32 and stride=16, we extracted per-window encodings while padding zeros at the end (black-frames) to account for selecting window-size amount of frames.

1. Frame resizing to 256×256 followed by center-cropping to 224×224 resolution.
2. Pixel scaling from *8-bit* format to *float* format ($[0, 1]$). Following this, mean and standard deviation normalization is performed.
3. We chose MViT-Base 32×3 that ingests a chunk of video (32 frames) at once and produces an embedding vector.

We import the architecture and pretrained parameters from PytorchVideo[2].

CLIP $\mathbf{f}^3$. This is a multi-modal backbone that can produce representations corresponding to matching textual descriptions. We borrow the CLIP [23] pre-trained model from the huggingface [33] library and use their inbuilt image processor as well as feature extractor to obtain subsampled frame-level encodings, $\mathbf{f}^3 \in \mathbb{R}^{512}$.

1. Short-side is resized to 224 pixels followed by center-cropping (to 224×224).
2. Re-scaling 8-bit image to $[0, 1]$ interval.
3. Mean and standard deviation normalization.

Dialog Features We test three different dialog features and present the details as follows.

Fine-tuning Language Models. We fine-tune language models such as $\text{PEGASUS}_{\text{LARGE}}$ [36], RoBERTa-Large [38], and MPNet-base-v2 [28] for our task of extractive dialog summarization. To account for the small dataset sizes, for all fine-tuning, we use the Adapter modules [12] that add only a few trainable parameters in the form of down- and up-projection layers as illustrated in Fig. 4.

[1] https://pytorch.org/hub/

[2] https://pytorchvideo.org/

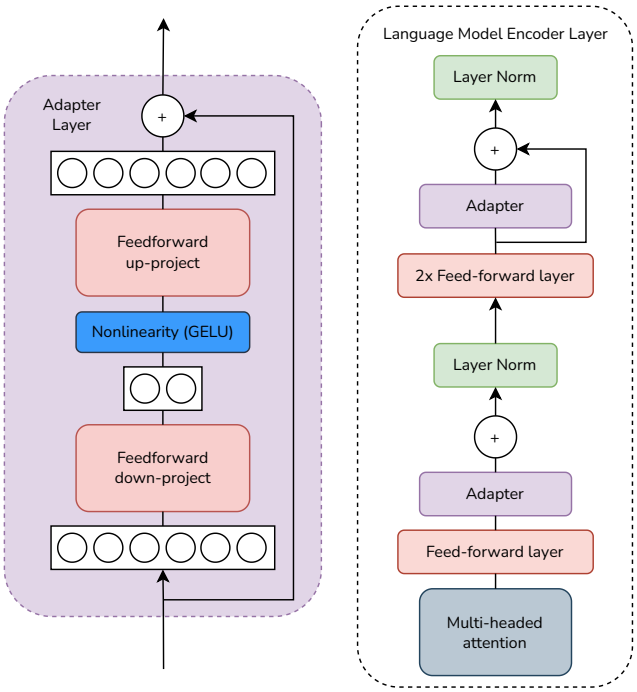

Figure 4. Architecture of the adapter module [12] and its integration with language model's encoder layer. **Left**: The adapter module has fewer parameters compared to the attention and feed-forward layers in a Transformer layer and consists of a bottleneck and skip-connection. **Right**: We add the adapter module twice to each encoder layer. Once after the multi-head attention and the other is placed after the two feed-forward layers typical of a Transformer architecture. For the Transformer *decoder*, in the case of $\text{PEGASUS}_{\text{LARGE}}$, we add one extra adapter after cross-attention as well. **Adaptation**: When adapting the model, the purple and green layers illustrated on the right module are trained on the downstream data and task, while the other blocks are frozen.

RoBERTa-Large [38] and MPNet-base-v2 [28] are fine-tuned for utterance-level binary classification to decide whether the dialog utterance is important or not.

$\text{PEGASUS}_{\text{LARGE}}$ is trained originally to generate abstractive summaries. Instead, we adapt it to generate summary dialogs. As the number of tokens accepted by the $\text{PEGASUS}_{\text{LARGE}}$ model does not allow feeding all the dialog from the episode, we break it into 6 chunks.

Word-level embeddings. We use *last hidden-state* of the encoder to obtain contextualized word-level embeddings, $\mathbf{w} \in \mathbb{R}^{1024}$ (768 for MPNet-base-v2). $\text{PEGASUS}_{\text{LARGE}}$, being a generative model (consisting of both encoder and decoder), we keep only the encoder portion for word-level feature extraction.

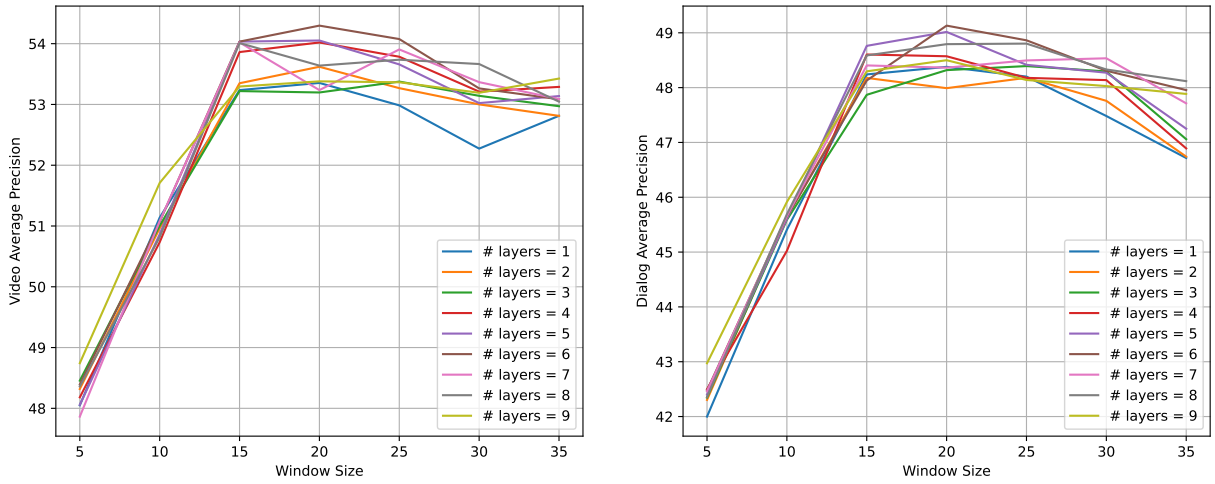

Figure 5. Performance of TaleSumm for varying local story group size (also referred to as window size) and number of layers in the episode-level Transformer ET. y-axis denotes the AP score for the **Left**: Video and **Right**: Dialog. We observe that a 6 layer model $H_{\mathrm{E}} = 6$ works well together with a local story group size of $n_g = 20$.

B.3. Feature and Architecture Ablations

All experiments are run on *IntraCVT* split, and we report mean and standard deviation.

Visual features for Story summarization. Table 2 shows the results for all combinations of the three chosen visual feature backbones in a multimodal setup. We draw two main conclusions: (i) The Shot Transformer encoder (ST) shows improvements when compared to simple average or max pooling. (ii) DMC (DenseNet + MViT + CLIP), the feature combination that uses all backbones, performs well, while MC shows on-par performance. Importantly, the feature combination is better than using any feature alone.

Language backbones for Story summarization. Different from the previous experiment, Table 3 shows results for different dialog features with different word-to-utterance pooling approaches in a multimodal setting. RoBERTa-Large outperforms all other language models. Nevertheless, the other models are not too far behind.

Number of ET layers and local story group size. Two important hyperparameters for our model are the number of layers H_{E} in the ET and the local story group size n_g. Fig. 5 shows the performance on video AP (left) and dialog AP (right) with a clear indication that 6 layers and a story group size of 20 shots are appropriate.

B.4. Adapting SoTA Approaches for PlotSnap

In this section, we discuss how we adapt different video- and dialog-only state-of-the-art baselines for our task.

MSVA [7] considers frame-level features from multiple sources and applies aperture-guided attention across all such feature streams independently, followed by intermediate fusion and a linear classification head that selects frames based on predicted importance scores. Since we are modeling at the level of the entire episode, we feed condensed shot features (after Avg or Max pooling or ST) of each backbone: DenseNet169 (X_o), MViT (X_r), and CLIP (X_f) through *three* different input streams and output shot-level importance scores.

Our model is different from MSVA as MSVA treats each feature separately, while we perform early fusion and concatenate representations from multiple backbones even before obtaining a compact video shot representation. TaleSumm is also developed for encoding and making predictions on multiple modalities, while MSVA is not.

PGL-SUM [4] splits the video in small equal-sized group of frames. Similar to our work, contextualization is performed within local multi-headed self-attention on small groups, while another is at global level using global multi-headed attention for the entire video. Later, both are merged via addition along the feature axis and subsequently passed through an MLP classification head to obtain frame-level scores. To adapt PGL-SUM to our work, we again think of shots as the basic unit and concatenate visual features from all streams ($\mathbf{f}^1$, $\mathbf{f}^2$, and $\mathbf{f}^3$), followed by pooling (ST or Max or Avg pooling) and then PGL-SUM to generate shot-level importance scores.

PGL-SUM has some similarities to our approach as both involve local groups. Interestingly, PGL-SUM creates groups of frames to perform summarization for short videos of a few minutes, while TaleSumm creates groups of shots and dialog utterances to generate summaries for 40 minute long episodes. Among technical contributions, we also explore different attention mechanisms such as across the full-

Pooling		Avg	Max	Cat	Tok	Stack	⊞
Concatenate		✓	✓	✓	NC	✓	✓
DMC	Vid AP	54.1 ± 4.5	54.1 ± 4.6	*54.2* ± 4.1	54.1 ± 4.0	53.9 ± 4.7	**54.2** ± 3.3
	Dlg AP	48.7 ± 4.1	48.8 ± 4.4	48.9 ± 4.7	*49.0* ± 4.8	*49.1* ± 4.8	**49.0** ± 4.9
DM	Vid AP	54.0 ± 3.8	54.1 ± 3.3	54.0 ± 4.1	53.9 ± 3.2	53.6 ± 4.3	53.7 ± 3.0
	Dlg AP	48.6 ± 4.4	48.8 ± 4.4	48.9 ± 4.4	48.6 ± 3.6	48.4 ± 4.8	48.6 ± 4.3
MC	Vid AP	53.8 ± 3.6	53.9 ± 4.0	53.8 ± 4.5	54.0 ± 4.9	53.8 ± 4.2	**54.2** ± 4.3
	Dlg AP	48.5 ± 4.6	48.8 ± 4.1	48.7 ± 4.5	*49.0* ± 4.1	48.5 ± 4.7	**49.1** ± 4.2
DC	Vid AP	54.0 ± 4.5	54.1 ± 4.3	54.1 ± 4.0	54.0 ± 4.0	54.1 ± 3.9	54.1 ± 4.2
	Dlg AP	48.2 ± 4.9	48.7 ± 4.5	48.9 ± 4.7	*49.0* ± 4.6	*49.0* ± 4.8	*49.0* ± 4.9
D	Vid AP	53.5 ± 4.2	53.4 ± 4.2	54.0 ± 3.8	-	-	-
	Dlg AP	48.1 ± 4.2	48.6 ± 4.3	48.9 ± 4.4	-	-	-
M	Vid AP	52.9 ± 3.2	53.6 ± 3.6	53.5 ± 3.4	-	-	-
	Dlg AP	48.2 ± 4.0	48.4 ± 3.9	48.7 ± 4.6	-	-	-
C	Vid AP	53.9 ± 3.7	53.9 ± 3.9	54.1 ± 3.7	-	-	-
	Dlg AP	48.6 ± 4.2	48.7 ± 4.2	48.7 ± 4.0	-	-	-

Table 2. TaleSumm ablations for different feature combination strategies, using **both video and dialog modalities**. Feature ablations are performed over the visual modality. Columns describe the *Pooling* approaches used to form shot-level from frame-level representations; *Concatenate* corresponds to how backbone feature are combined (concatenate ✓ or as separate tokens (NC)). *Stack* pooling is an alternative approach to ⊞, where instead of condensing the aggregated (concatenated) visual features via a linear layer $\mathbf{W}_{\mathcal{P}} \in \mathbb{R}^{3\times 3D}$, we obtain individual feature importance score (with $\mathbf{W}_{\mathcal{P}} \in \mathbb{R}^{1\times D}$ followed by $\tanh$ and $\mathsf{softmax}$). **D**: DenseNet169, **M**: MViT, and **C**: CLIP. All results are for TaleSumm that captures *Episode* level interactions.

Pooling		Max	Avg	wCLS
PEGASUS$_{\text{LARGE}}$ [36]	Vid AP	52.7 ± 4.3	53.1 ± 4.2	53.1 ± 4.7
	Dlg AP	47.9 ± 3.6	48.0 ± 5.1	47.9 ± 4.9
MPNet-Base [28]	Vid AP	53.2 ± 3.9	53.1 ± 3.7	53.5 ± 3.4
	Dlg AP	48.0 ± 4.4	47.2 ± 3.2	48.6 ± 5.0
RoBERTa-Large [38]	Vid AP	54.1 ± 3.8	**54.2** ± 3.3	54.1 ± 4.2
	Dlg AP	*49.0* ± 4.6	**49.0** ± 4.9	*49.0* ± 4.3

Table 3. TaleSumm ablations for various dialog utterance feature backbones. Visual features are fixed to DMC.

episode (FE) or within a local story group (SG). Different from PGL-SUM, we introduce a story group token that allows to capture the essence of a local story group.

PreSumm [20] is used for text-only extractive summarization which takes word-level inputs and produces sentence-level probability scores. To represent each episode, the utterances are concatenated, lower-cased, and separated by CLS and SEP tokens into a single line input. The PreSumm model leverages word embeddings from pre-trained BERT-base [5] language model. Considering the long inputs in our case, we extend the existing positional embeddings of BERT from 512 to 10000 by keeping the original embeddings and replicating the last embeddings for the remainder. At the sentence level, the corresponding CLS token is fed into two transformer encoder layers for contextualization, followed by a small MLP and sigmoid operation to generate per-sentence scores. The model is trained using the Adam [16] optimizer with Binary Cross-Entropy loss.

PreSumm is very different from our work as it operates directly on tokens, while our model develops a hierarchical approach going from words to dialogs to local story groups.

A2Summ [9] is a contemporary multimodal summarization (MSMO) baseline, primarily designed to align temporal correspondence between video and text signals through a dual-contrastive loss approach. They also introduce a dataset, BLiSS [9], comprising 13,303 pairs of livestream videos and transcripts, each with an average duration of 5 minutes, along with multimodal (VT2VT) summaries. A2Summ exploits cross-modality correlations within and between videos through dual contrastive losses. These include: (a) inter-sample contrastive loss (operates across different sample pairs within a batch, leveraging the intrinsic correlations between video-text pairs), and (b) an intra-sample contrastive loss (works within each sample pair, emphasizing the similarities between ground-truth video and text summaries while contrasting positive features with hard-negative features).

We adapt A2Summ for PlotSnap, where we work at the episodic level, by using max-pooling with an MLP for

video features and average-pooling for dialog (text) features to derive shot- and utterance-level representations. We create explicit intra-sample attention masks encouraging temporal alignment, allowing video shots to attend to their corresponding utterances and permitting video and utterance tokens to fully attend to their respective counterparts. Considering memory constraints, we maintain a batch size of 4 (four entire episodes - inter-sample contrasting) on a single NVIDIA GeForce RTX-2080 Ti GPU. We adopt CyclicLR [26] with a maximum learning rate of 10^{-4} and the 'triangular2' mode. A2Summ model comprises 6 encoder layers and incorporates multiple dropout layers [11]: (a) `dropout_video` $=0.1$, (b) `dropout_text` $=0.2$, (c) `dropout_attn` $=0.3$, and (d) `dropout_fc` $=0.5$, while keeping rest the of the hyperparameters the same. Training extends to a maximum of 50 epochs, with the AdamW [22] optimizer utilized, featuring a learning rate of 10^{-5} and a weight decay [21] $=0.01$.

In PlotSnap, where episodes show related content, the application of inter-episode contrastive learning negatively impacts the model's performance. This is due to the variability in the importance of related story segments across episodes, which depends on the specific context.

B.5. Details for SumMe and TVSum

In this section, we will discuss how we adapted our model for SumMe [8] and TVSum [29], some experimentation details, and corresponding evaluation metrics.

Adaptation. We used our video-based model on both datasets, inheriting features from MSVA[3] [7]. To capture shot-level details, we stacked 15 contiguous frame-embeddings (as previous methods utilized ground-truth labels indexed at every 15th frame), and for group-level, we used `n_frame_per_seg` attribute of the dataset. We used continuous-index-based time embeddings for shot-frames. Essentially, we assume that a shot consists of 15 frames as SumMe and TVSum require predictions at every 15 frames.

Hyperparameter configuration. We determine the configuration based on the best validation score obtained over five random splits (5-RCV). This time our model is trained on a single NVIDIA GeForce RTX-2080 Ti GPU for a maximum of 300 epochs for SumMe and 100 epochs for TVSum, with a batch size of 1. Additional dataset-specific hyperparameters are detailed in Table 4. In common, we have AdamW optimizer [22] with parameters: learning rate $=5\times10^{-5}$, weight decay [21] $=10^{-3}$. We use CyclicLR [26] for learning rate scheduling with max lr $=5\times10^{-4}$ and *triangular2* mode. ReLU is used for classification head and GELU for projection and attention modules.

[3] https://github.com/TIBHannover/MSVA/tree/master

B.6. Extended Qualitative Analysis

In this section, we analyze 3 more episodes and compare the model's prediction against all three labels (GT, F, and H). Recall, the labels denoted *F* (Fandom) are based on summarized plot synopses from the *24* fan site[4] that includes the key events in the story (as a text description). Plot synopses are short textual descriptions of the key story events of an episode. We ask annotators to use the plot synopses and tag the start-end duration for story sequences corresponding to the description. We refer to these labels as *Fandom* (F) and use them for qualitative evaluation. While the *H*-labels are annotations from a human, based on what they feel is relevant summary as per the narrative.

Qualitative evaluation. We present importance scores for three episodes: S06E20[5], S07E22[6], and S05E21[7], in Fig. 6, Fig. 7, and Fig. 8, respectively. We observe that the model predictions are quite good and not only match the ground-truth labels (on which the model is trained) but also the fandom and human annotations. Please refer to the figure captions for additional comments on episode-specific remarks.

C. Future Work

Ingesting ∼40 minute long videos is a challenging problem. Aligning different modalities, such as processing a 40-minute video containing 2,500 frames (at 1fps) and 4,500 dialogue words, totaling approximately 8000 tokens, may require a larger GPU memory. And most of the L-VLMs [15, 19, 25] struggle to handle contexts exceeding 8,000 tokens, and even if they do, consumer-grade GPUs may lack the capacity to accommodate them.

Our approach considers coarse-grained visual information, which we demonstrate is beneficial for story-summarization. Considering more fine-grained visual info, such as person and face tracks across frames, and their emotions, would be useful. Our idea of using recap-inspired story summary labels or modeling approach are not specific to thrillers and can be easily extended to other genres and shows with recaps. Having speaker information in dialog utterances with mapping to character faces would probably improve performance on the summarization task. The local story groups are a proxy to scene segments of an episode. Replacing them with actual scene segments may improve our model's performance for summarization.

Finally, further analysis and experiments are required to determine the quality of these methods [30], particularly

[4] https://24.fandom.com/wiki/Wiki_24

[5] https://24.fandom.com/wiki/Day_6:_2:00am-3:00am talks about the key story events in S06E20 in a *Previously on 24* section (see Fig. 6).

[6] https://24.fandom.com/wiki/Day_7:_6:00am-7:00am talks about the key story events in S07E22 in a *Previously on 24* section (see Fig. 7).

[7] https://24.fandom.com/wiki/Day_5:_4:00am-5:00am talks about the key story events in S05E21 in a *Previously on 24* section (see Fig. 8).

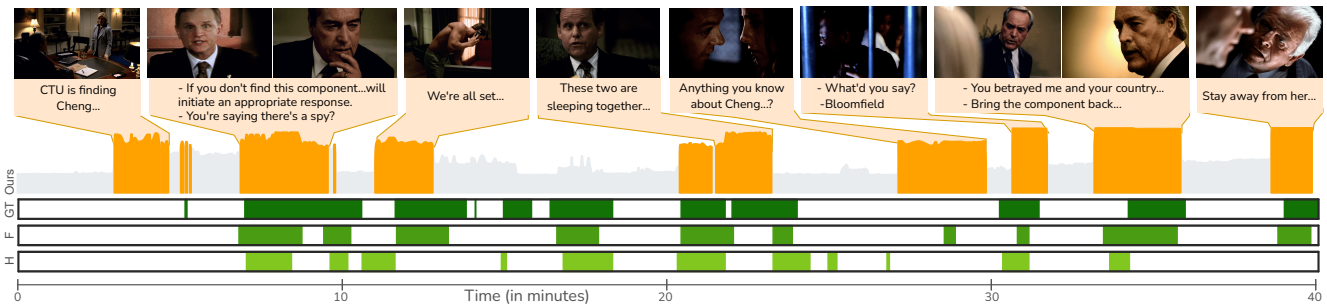

Figure 6. TaleSumm predictions on S06E20 of *24* (test set). "Ours" filled-plot illustrates the importance score profile over time, where orange patches indicate story segments selected for summarization. Annotations are shown below: ground-truth (GT), fandom (F), and human annotated (H). We number the grouped frames representing the predicted contiguous orange chunks as shot groups (SG-n), *e.g.* this episode has 8 SGs. **The story**: The White House directs CTU to locate Cheng, as depicted in SG-1, who possesses a Russian sub-circuit board that threatens national security. In SG-2, President Suvarov warns of military consequences if the Chinese agent with the circuit board isn't intercepted. SG-3,4,7 shows how Lennox suspects a spy within the administration and uncovers Lisa's treason. President Noah Daniels instructs Lisa to bring the component back by misleading her partner, Mark Bishop. In SG-5,6, Jack questions Audrey about Cheng, leading to a standoff with Doyle. Audrey mentions "Bloomfield," prompting research by Chloe. In his holding room, Heller warns Jack to stay away from Audrey due to the deadly consequences associated with him (SG-8). Intricate relationships and the imminent threat of international conflict mark the overall content of this episode.

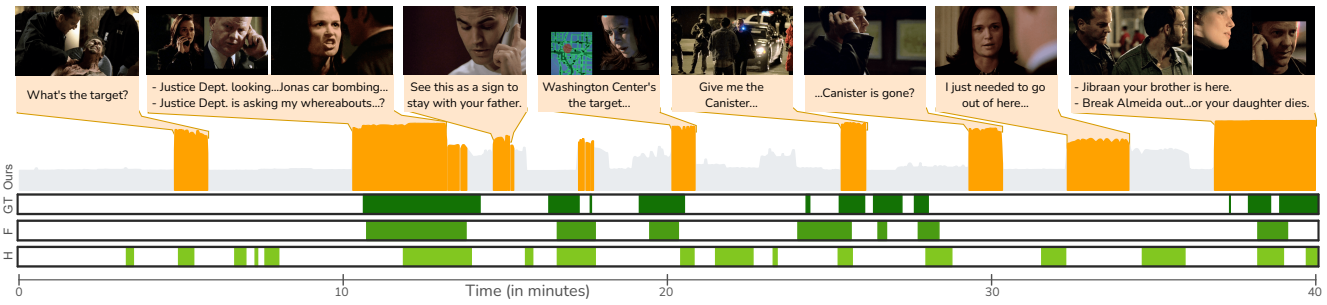

Figure 7. TaleSumm predictions on S07E22 of *24* (test set). "Ours" filled-plot illustrates the importance score profile over time, where orange patches indicate story segments selected for summarization. Annotations are shown below: ground-truth (GT), fandom (F) and human annotated (H). We number the grouped frames representing the predicted contiguous orange chunks as shot groups (SG-n), *e.g.* this episode has 8 SGs. **The story**: In a tense sequence, as shown in SG-1, Jack resorts to torture to extract information from Harbinson about the impending attack but is left empty-handed. In SG-2, following the murder of Jonas Hodges, Olivia Taylor faces scrutiny from the Justice Department. Meeting with Martin Collier, she denies transferring funds, revealing a sinister plot. Meanwhile, SG-3 shows Kim Bauer's plans are disrupted by a flight delay, leading to a strained father-daughter relationship. SG-4,5 displays how Jack, aided by Chloe O'Brian and Renee Walker, captures Tony Almeida and interrogates him about a dangerous canister, followed by Renee uncovering Jibraan's location, and a high-stakes exchange ensues at the Washington Center station. Jack detonates the canister, succumbing to its effects. As a consequence (SG-6), Cara Bowden reports Tony's failure to Alan Wilson, adding tension to the unfolding crisis. Olivia returns to the White House, explaining her absence to Aaron Pierce (SG-7), which beautifully connects back to the SG-2. The narrative takes a dire turn as Cara blackmails Jack for the safety of Kim (SG-8), introducing a new layer of suspense and complexity to the unfolding events. *The presence of SG-1,6,7 (absent in GT) clearly highlights our model's ability to complete the overall story arc.*

because evaluating long video summarization using human judgment is very time-consuming.

However, we believe that this work provides a window into this challenging problem and can help facilitate further research in this area.

References

[1] Common Crawl - News. https://commoncrawl.org/blog/news-dataset-available, 2016. 4

[2] OpenWebText. https://github.com/jcpeterson/openwebtext?tab=readme-ov-file, 2019. 4

[3] English Wikipedia. https://en.wikipedia.org/wiki/English_Wikipedia, 2021 – Present. 4

[4] Evlampios Apostolidis, Georgios Balaouras, Vasileios Mezaris, and Ioannis Patras. Combining Global and Local Attention with Positional Encoding for Video Summarization. In *IEEE International Symposium on Multimedia (ISM)*, 2021. 4, 6

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina

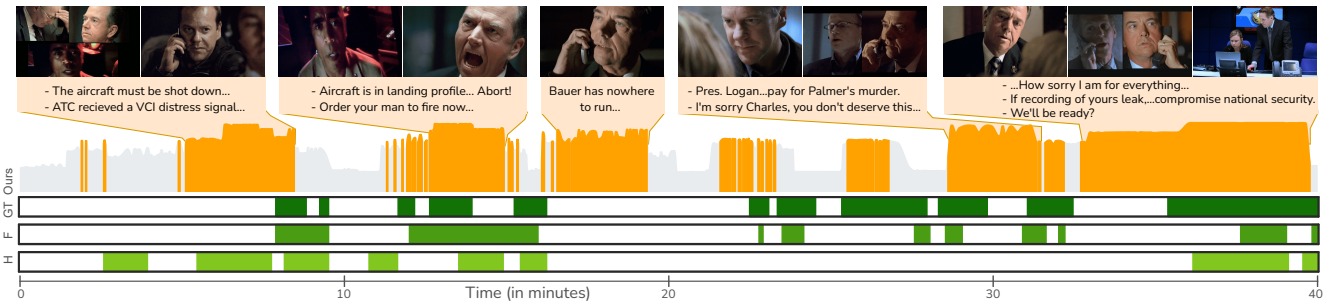

Figure 8. TaleSumm predictions on S05E21 of *24* (test set). "Ours" filled-plot illustrates the importance score profile over time, where orange patches indicate story segments selected for summarization. Annotations are shown below: ground-truth (GT), fandom (F), and human annotated (H). We number the grouped frames representing the predicted contiguous orange chunks as shot groups (SG-n), *e.g.* this episode has 5 SGs. This episode stands out due to its rapid and significant story advancements, where each sub-story holds apparent importance. Also, human annotations are a bit off in comparison to the ground-truth (can be verified from reliability scores shown in Tab. 1). Importantly, our model considers opinions from all the sources. **The story**: In the SG-1,2, President Logan, pretending surprised, learns from Admiral Kirkland that Flight 520, now under Jack's control, is a potential threat. Despite Mike's doubts about Jack's intentions, Kirkland urges immediate action, advocating for shooting down the plane. Logan, pretending shock, reluctantly authorizes the attack. Karen alerts Jack to the order, leading to a tense situation. As the plane assumes a landing profile, Kirkland suggests calling off the strike, but Pres. Logan insists on taking it down. Further, in SG-3, Graem criticizes Logan's decision, emphasizing the importance of capturing Jack, but Logan assures Graem of recapturing him. Meanwhile, in SG-4, Jack, having secured incriminating evidence, vows to make Logan pay for President Palmer's assassination. In a surprising turn, as shown in SG-5, President Logan contemplates suicide, but an unexpected call from Miles Papazian presents an alternative – the destruction of the recording. Encouraging Miles to act, Logan faces a critical juncture in the unfolding crisis. Overall the entire episode sets the stage for a series of dramatic events, stressing the depth of deceit and the potential consequences for key characters.

	amsgrad	d_model	drop_fc/trm/proj	dec_l	enc_l	wd	act_clf/mlp/trm	ffs
SumMe [8]	True	512	0.5/0.2/0.2	3	1	0.0001	r/g/g	Stack
TVSum [29]	False	768	0.7/0.4/0.2	6	1	0.01	r/g/g	⊞

Table 4. Hyperparameter configuration for SumMe and TVSum. `d_model` specify the dimension for the transformer module's internal representation, while `dec_l` and `enc_l` denote # of decoder and encoder layers, respectively. Other hyperparameters include `drop_fc/trm/proj` for dropout at classification head, attention module, and projection module, `wd` stands for weight decay parameter (used inside AdamW [22]), `act_clf/mlp/trm` for activation function used in classification head, projection, and attention module, with `r` for ReLU and `g` for GELU. Lastly, `ffs` stands for feature fusion style, with Stack and ⊞ depicting the pooling strategy showed in Table 2.

Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805*, 2018. 7

[6] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *International Conference on Computer Vision (ICCV)*, 2021. 4, 5

[7] Junaid Ahmed Ghauri, Sherzod Hakimov, and Ralph Ewerth. Supervised Video Summarization Via Multiple Feature Sets with Parallel Attention. *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6s, 2021. 4, 6, 8

[8] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. Creating summaries from user videos. In *European Conference on Computer Vision Workshops (ECCVW)*, 2014. 1, 8, 10

[9] Bo He, Jun Wang, Jielin Qiu, Trung Bui, Abhinav Shrivastava, and Zhaowen Wang. Align and Attend: Multimodal Summarization with Dual Contrastive Losses. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 4, 7

[10] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1

[11] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, abs/1207.0580, 2012. 4, 8

[12] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning (ICML)*, 2019. 5

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 4, 5

[14] Will Kay, João Carreira, Karen Simonyan, Brian Zhang,

Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Apostol Natsev, Mustafa Suleyman, and Andrew Zisserman. The Kinetics Human Action Video Dataset. *CoRR*, abs/1705.06950, 2017. 4, 5

[15] Wonjae Kim, Bokyung Son, and Ildoo Kim. ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision. In *International Conference on Machine Learning*, 2021. 8

[16] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, 2014. 7

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *cs.toronto.edu*, 2009. 4, 5

[18] Hector J. Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2012. 4

[19] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. In *International Conference on Machine Learning*, 2022. 8

[20] Yang Liu and Mirella Lapata. Text Summarization with Pretrained Encoders. In *Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019. 4, 7

[21] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*, 2017. 8

[22] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*, 2017. 4, 8, 10

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*. PMLR, 2021. 4, 5

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 211–252, 2015. 1, 4, 5

[25] Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. FLAVA: A foundational language and vision alignment model. In *CVPR*, 2022. 8

[26] Leslie N. Smith. Cyclical Learning Rates for Training Neural Networks. In *Winter Conference on Applications of Computer Vision (WACV)*, 2015. 8

[27] Leslie N. Smith and Nicholay Topin. Super-convergence: very fast training of neural networks using large learning rates. In *Defense + Commercial Sensing*, 2018. 4

[28] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and Permuted Pre-Training for Language Understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 5, 7

[29] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. TVSum: Summarizing web videos using titles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 8, 10

[30] Liyan Tang, Tanya Goyal, Alex Fabbri, Philippe Laban, Jiacheng Xu, Semih Yavuz, Wojciech Kryscinski, Justin Rousseau, and Greg Durrett. Understanding Factual Errors in Summarization: Errors, Summarizers, Datasets, Error Detectors. In *Association of Computational Linguistics (ACL)*, 2023. 8

[31] Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen. StoryGraphs: Visualizing Character Interactions as a Timeline. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 3

[32] Trieu H. Trinh and Quoc V. Le. A Simple Method for Commonsense Reasoning. *ArXiv*, abs/1806.02847, 2018. 4

[33] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019. 5

[34] Yusseri Yusoff, William J. Christmas, and Josef Kittler. A Study on Automatic Shot Change Detection. In *European Conference on Multimedia Applications, Services and Techniques*, 1998. 4

[35] Netzer Yuval. Reading digits in Natural Images with Unsupervised Feature Learning. In *Advances in Neural Information Processing Systems-Workshop (NeurIPS-W)*, 2011. 4, 5

[36] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning (ICML)*. PMLR, 2020. 5, 7

[37] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. In *arXiv preprint arXiv:1506.06724*, 2015. 4

[38] Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. A Robustly Optimized BERT Pre-training Approach with Post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, 2021. 4, 5, 7