Joint Moment Retrieval and Highlight Detection Via Natural Language Queries

Richard Luo rluo47@gatech.edu
Austin Peng apeng39@gatech.edu
Koby Beard kbeard8@gatech.edu
Heidi Yap hyap7@gatech.edu

Georgia Institute of Technology
Atlanta, GA 30332

Abstract

Video summarization has become an increasingly important task in the field of computer vision due to the vast amount of video content available on the internet. In this project, we propose a new method for natural language query based joint video summarization and highlight detection using multi-modal transformers. This approach will use both visual and audio cues to match a user's natural language query to retrieve the most relevant and interesting moments from a video. Our approach employs multiple recent techniques used in Vision Transformers (ViTs) to create a transformer-like encoder-decoder model. We evaluated our approach on multiple datasets such as YouTube Highlights and TVSum to demonstrate the flexibility of our proposed method. Our code and raw results is publicly available at https://github.com/Skyline-9/Visionary-Vids

1. Introduction/Background/Motivation

1.1. Introduction/Background

Video content has proliferated in the last few years, now taking up the majority of all internet traffic. However, video is an inherently dense data medium, with a single video encoding large amounts of information about visual, audio, and linguistic elements. With increasingly large amounts of video available, there has been an increasing focus on two important needs: finding a specific, relevant moment in a video and quickly skimming over the a video to summarize its content. This reflects a shift towards summarizing or extracting just the highlights of a video. Twitch, for example, has clips where users can splice just an interesting section of hours-long live streams. Similarly, TikTok, Instagram Reels, and YouTube Shorts all move towards short-form content.

1.2. Related Works

Currently, video highlighting/summarization is largely done by hand. Vloggers or other content creators need to sift through long streams of video and cut them out to reveal only the most relevant parts. There are many online platforms that crowdsource the process of generating highlights, such as the subreddit r/nba, where users will clip a stream and post it on the forum [1]. Highlight generation is also done via machine learning, although many of the approaches are domain specific. For example, various deep learning schemes have been approached to perform automatic highlight detection in soccer [7] [13]. More recently, there have been attempts to generalize video summarization and highlight detection [22]. Another example is video thumbnail generation, which focuses on the downstream task of video summarization by selecting a short video clip to serve as the thumbnail [32]. However, video is an inherently data rich medium, as it contains information about a myriad of things such as objects, language, movement, sound, and time. This brings about a need for multimodal models. There have been some attempts at multimodal models, but they lack the critical ability to query specific moments [27]. In video thumbnail generation, which relies on video highlight summarization, early approaches used graph convolutional networks to model interactions between the word and text queries [21]. Other approaches that have the ability to query moments and generate highlights lack the information gained in a multi-modal model [17] [16].

1.3. Motivation

Video summarization and highlight detection have become increasingly important tasks in the field of computer vision, with significant potential to benefit a broad range of stakeholders, including content creators, marketers, researchers, and end-users. Content creators and marketers can use video summarization and highlight detection to increase engagement and retention for their target audience. By identifying and highlighting the most relevant and interesting parts of their video content, they can keep viewers engaged and motivated to watch more, leading to higher viewership, increased brand exposure, and potentially, higher revenues. On the other hand, researchers can use video

Method	YT	TVSum	QVHighlights	
	mAP	Top-5 mAP	MR (mAP)	HD (mAP)
MINI-Net [12]	64.36	73.24	–	–
Joint-VA [2]	71.80	76.30	–	–
Moment-DETR [16]	–	–	34.05	37.67
UMT [18]	74.93	83.14	38.59*	39.85*
VisionaryVid (Ours)	**76.15**	**85.50**	38.28*	39.12*
* denotes close performance to state of the art ($\triangle$ mAP < 1)				

Table 1. Effectiveness of multimodal learning on YouTube Highlights, TVSum, and QVHighlights

Method	Charades-STA			
	MR (mAP) R1@0.5	MR (mAP) R1@0.7	HD (mAP) R5@0.5	HD (mAP) R5@0.7
SAP [5]	27.42	13.36	66.37	38.15
SM-RL [31]	24.36	11.17	61.25	32.08
UMT† [18]	**48.31**	**29.25**	**88.79**	**56.08**
VisionaryVid† (Ours)	25.97	13.25	84.87	40.38
† denotes trained with video + audio				

Table 2. Effectiveness of multimodal learning on Charades-STA

summarization and highlight detection to improve the efficiency and effectiveness of data analysis, enabling them to quickly identify the most relevant and interesting parts of their videos for further investigation. This can help accelerate research progress and lead to new insights in various fields, such as social sciences, psychology, and medicine.

End-users can benefit from video summarization and highlight detection by being able to quickly find the most relevant and interesting parts of a video, saving them time and effort. This is especially important with the vast amount of video content available on the internet nowadays, where users are usually short on time and attention.

If successful, our proposed method using multi-modal transformers for natural language query-based joint video summarization and highlight detection has the potential to significantly improve the efficiency and effectiveness of video summarization and highlight detection, benefiting all of these stakeholders. This has the opportunity to ultimately lead to a more engaging, informative, and enjoyable video experience for all users.

1.4. Datasets

QVHighlights [16]:

QVHighlights is a large-scale video dataset that contains 4,000 online video clips from 35 popular sports categories, including basketball, soccer, tennis, and more. The dataset includes manually annotated highlight timestamps and corresponding importance scores from 3-5 annotators for each video clip. In addition, it also contains audio transcripts and visual frames extracted from each video clip. This dataset was introduced in the paper "QVHighlights: A Large-Scale Video Dataset for Highlight Detection in Sports" by Lei et al. in 2021.

Charades-STA [10]:

Charades-STA is a video captioning and temporal localization dataset that includes 9,848 short video clips of daily activities, such as brushing teeth, cooking, and more. The dataset provides 33,753 natural language sentences for describing the video content, as well as temporal annotations for the beginning and ending of the action segments within each video clip. Charades-STA also includes a challenge set of 500 videos with more complex activities and multiple action segments. This dataset was introduced in the paper "Charades-STA: A Benchmark Dataset for Natural Language Visual Reasoning and Scripting" by Gao et al. in 2017.

YouTube Highlights [26]:

YouTube Highlights is a video highlight dataset that contains 12,000 video clips from 20 popular sports categories, including basketball, soccer, and football. Each video clip is annotated with the start and end timestamps of the highlights and the corresponding importance score from 5 annotators. The dataset also includes audio transcripts and visual frames extracted from each video clip. This dataset was introduced in the paper "Ranking and Classifying Atypical Activities in Video" by Sun et al. in 2014.

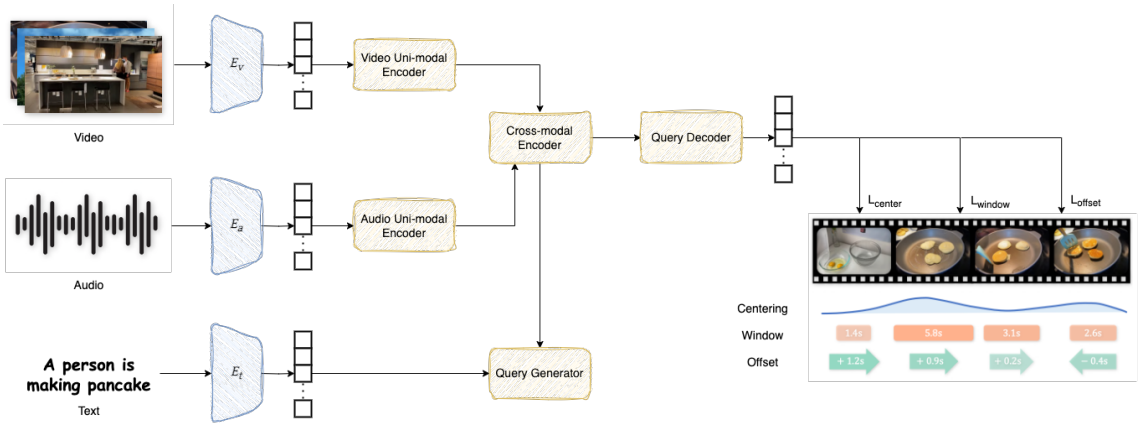

Figure 1. VisionaryVids Model Architecture

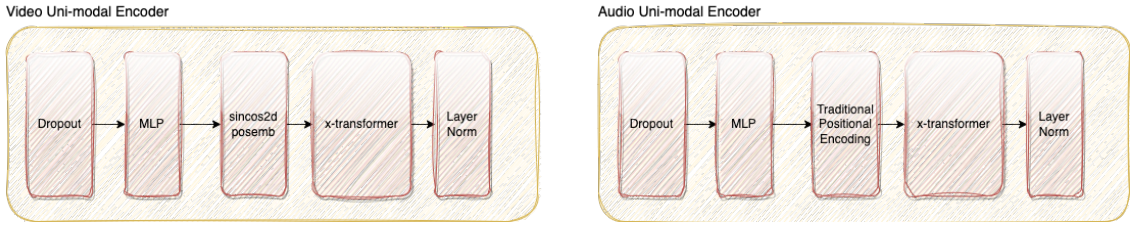

Figure 2. VisionaryVids Video and Audio Encoder

TVSum [24]:

TVSum is a video summarization dataset that contains 50 videos of various topics, such as news, documentaries, and sports. The dataset provides annotations of the keyshot frames and their corresponding importance scores, which are based on the amount of information presented in the video. In addition, TVSum also includes annotations of the human-created summaries for each video, as well as baseline summaries generated by different summarization methods. This dataset was introduced in the paper "TVSum: Summarizing web videos using titles" by Song et al. in 2015.

2. Approach

2.1. Overview

Our team was inspired by the work done by Unified Multi-modal Transformers [18], vision transformers (ViT) [8], and the improved vision transformers [3] and combined these state of the art efforts for our model's novelty. Our model is shown in Figure 1 and Figure 2.

In the UMT model, we decided to improve the uni-modal encoder for both video and audio. For video, we took inspiration from improvements made to the vanilla vision transformer, specifically in the positional encoding. The original transformer paper uses a sinusoidal positional embedding, which is used in the original UMT paper [28]. However, recent advances to ViTs found that using a sine-cosine 2D positional encoding achieved better results by capturing more fine-grained local details [3].

Additionally, for both the video and audio uni-modal encoders, we improved the standard transformer self-attention mechanism by augmenting with persistent memory. The idea is to combine the self-attention layer and the feed-forward layer. This would simplify the model with no performance loss while also being capable of storing contextual information. However, empirical results have shown that including both the feed-forward layer and persistent memory yield better results [25].

Finally, we brought in a relatively new, simple optimizer called Lion (EvoLved Sign Momentum). Lion caught our attention as it brought ViT to the top performer on the ImageNet benchmark. It is memory-efficient and achieves stronger generalizations across architectures when compared to optimizers (ex. Adam, AdamW, Adafactor) used in recent state-of-the-art models.

2.2. Positional Encoder

The original ViT paper [8] uses standard 1d positional embeddings. The improved vision paper transformers paper [3] uses 2d positional embdeddings for better results.

Method	TVSum										
	HD (Top-5 mAP)										
	BK	BT	DS	FM	GA	MS	PK	PR	VT	VU	Avg.
MINI-Net [12]	75.0	80.2	65.5	57.8	78.2	81.8	78.1	65.8	80.6	68.3	73.1
Joint-VA [2]	73.0	**97.4**	67.5	70.0	78.5	86.1	80.1	69.2	83.7	57.3	76.3
TCG [33]	77.3	78.6	68.1	71.6	81.9	78.6	80.2	75.5	85.0	71.4	76.8
UMT [18]	86.9	84.4	79.6	76.0	88.2	78.8	81.4	**87.0**	**87.5**	81.5	83.1
VisionaryVids (Ours)	**92.4**	86.5	**80.3**	**79.4**	**91.7**	**87.3**	**82.9**	82.7	84.9	**87.1**	**85.5**

Table 3. Effectiveness of multimodal learning on TVSum. Comparison with representative highlight detection methods on TVSum dataset. Above are the methods using visual-audio features.

Method	YT						
	HD (mAP)						
	Dog	Gym.	Par.	Ska.	Ski.	Sur.	Avg.
MINI-Net [12]	58.2	61.7	70.2	72.2	58.7	65.1	64.4
Joint-VA [2]	55.4	62.7	70.9	69.1	60.1	59.8	63.0
TCG [33]	64.5	71.9	80.8*	62.0	**73.2**	78.3	71.8
UMT [18]	65.9	75.2*	**81.6**	71.8	72.3*	**82.7***	74.9
VisionaryVids (Ours)	**70.0**	**75.2***	78.2	**79.4**	72.0	82.1*	**76.2**
* denotes close performance to state of the art ($\triangle$ mAP < 1)							

Table 4. Experimental results on YouTube Highlights dataset. Above are the methods using visual-audio features.

Specifically, a 2d sin cos function is used for positional embedding. The 2d function allows for improved representation of the spatial information of the input. We use this same 2d sin cos positional embedding.

2.3. Persistent Memory

The idea of persistent memory was first introduced as a replacement for the feed-forward layer [25]. The persistent memory block allows the self-attention layer and feedforward layer to be combined into one, simplifying the model. Persistent memory works by concatenating the weights of the feedforward vectors onto the key and value vectors of the self-attention layer. Empirical findings suggest that keeping the feedforward layer while adding persistent memory leads to even better performance [30]. Our model leverages this optimization, and uses the self-attention layer with persistent memory and a feedforward layer.

2.4. Lion Optimizer

The Lion optimizer is a recently proposed optimizer [6] that was discovered from a program search for optmization algorithms for deep neural networks. To close the gap between proxy and target tasks, Lion introduces a program selection and simplification strategy using evolved sign momentum.

Some benefits of Lion are that is is more memory efficient than Adam because it only tracks momentum. In addition, it uses the same magnitude for the update unlike adaptive optimizers. The main difference is that it uses the sign operation to calculate the magnitude.

The Adam optimizer is a widely used optimization algorithm that is based on adaptive moment estimation. The optimizer computes adaptive learning rates for each parameter based on the first and second moments of the gradients. This allows the optimizer to converge more quickly and efficiently than traditional gradient descent algorithms. It also incorporates momentum, which helps the optimizer to continue moving in the same direction when the gradients are consistent. This can help to speed up convergence and avoid getting stuck in local optima.

One paper [6] presents Lion and evaluates its performance against widely used optimizers such as Adam and Adafactor on various deep learning models and tasks. The experiments show that Lion outperforms Adam and Adafactor in several scenarios; Lion boosts the accuracy of ViT (Vision Transformer) by up to 2% on ImageNet and reduces pre-training compute by up to 5x on JFT (JFT-300M, a large-scale image-text dataset). In vision-language contrastive learning, Lion achieves 88.3% and 91.1% accuracy on ImageNet, surpassing the previous state-of-the-art results by 2% and 0.1%, respectively. Lion also outperforms Adam in diffusion models by achieving a better FID

(Fréchet Inception Distance) score and reducing training compute by up to 2.3x. For autoregressive, masked language modeling, and fine-tuning, LION exhibits similar or better performance compared to Adam.

In this study, we investigate the performance of the Lion optimizer as an alternative to the widely used Adam optimizer for training the UMT model by implementing Lion and integrating it into the existing code.

2.5. Problems

Initially, we anticipated extremely long training times due to the large structure of the model and the many many transformers used. However, after using PyTorch 2's torch.compile optimization, we were able to significantly speed up our training speed. Additionally, we gained access to a multi-GPU server, which allowed us to train the larger datasets such as QVHighlights and Charades-STA.

Our first approach was actually to use convolutions (e.g. ConvNeXt v2) as an encoder layer instead of the typical TransformerEncoderLayer. This idea was that convolutions are inherently translation invariant and locality sensitive, which allows them to perform better when not a lot of data is fed in. Because our datasets are relatively small compared to something like ImageNet (which has over 14 million images), the presupposition was that convolutions would outperform transformers. However, because we are using feature extractors before feeding in the code, the convolutions would not have the raw image data fed into them, which makes them lose many of their inherent values. As such, we pivoted towards improving the transformer architectures.

3. Experiments and Results

3.1. Experimental Settings

Evaluation Metrics Following existing works, we use the same evaluation metrics based off mean average precision (mAP) [18]. In the YouTube Highlights, mAP is used, and in the TV-Sum dataset, top 5 mAP is used. For QVHighlights moment retrieval, we adopt Recall@1 with IoU thresholds of 0.5 and 0.7, mAP with IoU thresholds 0.5 and 0.75, and average mAP over a series of IoU thresholds [0.5:0.05:0.95]. For QVHighlights highlight detection, we adopt mAP and HIT@1, where a clip prediction is treated as a true positive if it has the saliency score of Very Good. In the CharadesSTA dataset, we used Recall@1 and Recall@5 with IoU thresholds 0.5 and 0.7.

Feature Extraction

Given an untrimmed video and a query, our research aims to find all the video moments where visual-audio contents are relevant to the query. To simplify our task, we utilize pre-trained feature extractors to extract visual, audio, and textual features from video and clip datasets.

More specifically, with QVHighlights, we leveraged the extracted features using SlowFast [9] and CLIP [20]. For Charades-STA, we utilized optical flow features, VGG [23], and GloVe [19] embeddings.

For the remaining two datasets, TVSum and YouTube Highlights, we leveraged I3D [4] pre-trained on Kinetics 400 [14] to obtain visual features. For TVSum, we also extracted the titles using CLIP.

For all four datasets, we extracted audio features using a PANN [15] model pre-trained on AudioSet [11]. The corresponding visual and audio features of each short segment of the video (aka clip) was synchronized to occur at the same time interval using timestamps.

Note: The extracted video and audio features are fed into separate uni-modal encoders then fused by a cross-modal encoder, following the method from UMT [18].

Model Configuration

In YouTube Highlights and TVSum, we use 1 decoder layer each, and in QVHighlights and Charades-STA, we use 3 decoder layers due to the larger dataset size. In each of the experiments, we use only one cross-modal encoder and one uni-modal encoder each for visual features, audio features, and query. Following UMT [18], we use learning sincos2d positional encodings, pre-norm style layer normalization, 8 attention heads, and a dropout rate of 0.1 for all the transformers, with extra pre-dropouts with rate 0.5 for visual and audio inputs, and 0.3 for text inputs. As for the optimizer, we used the Lion optimizer [6, 29]. On YouTube Highlights and TVSum, we used learning rate 1e-4, 5e-4, and 1e-3 with weight decay 1e-4. We evaluated each dataset on the best combination of each of these and took the best result. For Charades and QVHighlights, the training time was much longer, so we only tested one combination of learning rate 1e-4 and weight decay 1e-4. The training of the model involved using various batch sizes and epoch numbers for different datasets. Specifically, a batch size of 32 was used for 200 epochs on QVHighlights, a batch size of 8 was used for 100 epochs on Charades-STA, a batch size of 4 was used for 100 epochs on YouTube Highlights, and a batch size of 1 was used for 500 epochs on TVSum.

3.2. VisionaryVids Joint Moment Retrieval and Highlight Detection Results

We first evaluate our model on the YouTube Highlights, TVSum, and QVHighlights dataset shown in Table 1, in comparison to all the other performances reported before. On highlight detection, we outperform previous state-of-the-art method UMT in the YouTube Highlights and TVSum dataset. However, on QVHighlights, we perform near state-of-the-art in both moment retrieval and highlight detection. This is likely due to the lack of hyperparameter tuning for QVHighlights, as the larger dataset and more

complicated data required more detail. Specifically, it's important to note that both our approach with VisionaryVids and UMT use more decoder layers in QVHighlights than in YouTube and TVSum due to the larger scale of the dataset, and this may be a factor to tune as a hyperparameter. Since the loss curve for QVHighlights in Figure 3 ends up going back up, we believe that our hyperparameteres were faulty. With sufficient time and resources to perform hyperparameter tuning on QVHighlights, we can improve the performance of VisionaryVids.

Next, we evaluated our model on Charades-STA against UMT and other pre-existing methods using video and audio. While UMT also used the optical flow features of Charades instead of audio, we did not have time to test the optical flow features [18]. These results are shown in Table 2. Surprisingly, our model performed terribly compared to UMT on moment retrieval, and slightly worse on highlight detection. Without more hyperparameter tuning, it's difficult to say whether our model architecture has large flaws or if the hyperparameters we chose as default performed extremely poorly. Another key characteristic to note is that Charades-STA is a large dataset, even larger than QVHighlights.

3.3. Strengths/Weaknesses of Our Approach

First, looking at the TVSum results in Table 3, our model has a higher ceiling than previous state-of-the-art UMT. On the BK dataset in TVSum, our model performed over 5% better as measured by mAP. Similarly, we outperform UMT again by over 5% on the VU dataset. This trend holds again for the YouTube Highlights results in Table 4. We outperform previous state-of-the-art UMT again on Dog by over 4% and over 7% on Skate.

However, we perform similarly to state-of-the-art on QVHighlights and worse on Charades-STA. Looking at the loss curves in Figure 3, we can see that we are drastically overfitting the training set. Quantitatively, our validation loss is twice as high as our training loss! Future research can be done investigating how to reduce overfitting on the QVHighlights dataset, either using increased weight decay, different batch sizes, increasing dropout in the transformers, or changing the number of decoder layers. Looking at the loss curves in Figure 4, we see that loss is not decreasing after just a few epochs, and even worse, the training loss is very erratic and high. This likely suggests underfitting, and possibly also a bad set of hyperparameters, as we are stuck in a local optima.

4. Work Division

Our work division is shown in Table 5.

References

[1] r/nba subreddit. Accessed: 2023-04-26. 1

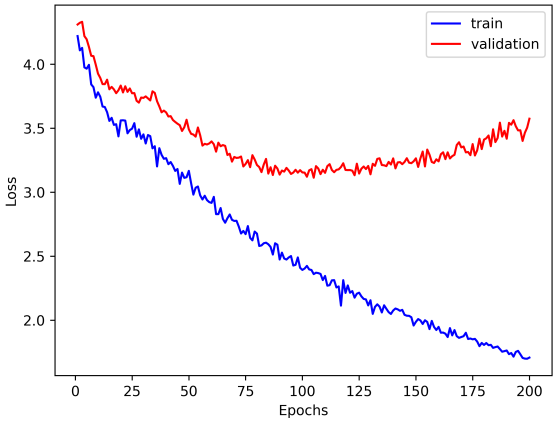

Figure 3. Loss Curve for QVHighlights

[2] Taivanbat Badamdorj, Mrigank Rochan, Yang Wang, and Li Cheng. Joint visual and audio learning for video highlight detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8127–8137, 2021. 2, 4

[3] Lucas Beyer, Xiaohua Zhai, and Alexander Kolesnikov. Better plain vit baselines for imagenet-1k, 2022. 3

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset, 2018. 5

[5] Shaoxiang Chen and Yu-Gang Jiang. Semantic proposal for activity localization in videos via sentence query. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 8199–8206, 2019. 2

[6] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifend Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms, 2023. 4, 5

[7] Narayana Darapaneni, PrashanthAnand Kumar, Nikhil Malhotra, Vigneswaran Sundaramurthy, Abhaya Thakur, Shivam Chauhan, Krishna Chaitanya Thangeda, and Anwesh Reddy Paduri. Detecting key soccer match events to create highlights using computer vision. *ArXiv*, abs/2204.02573, 2022. 1

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 3

[9] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 5

[10] Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, page 5267–5275, 2021. 2

Student Name	Contributed Aspects	Details
Richard Luo	Data Collection, Implementation, Training	Wrote scripts to download all four datasets, implemented sincos2D positional embedding, implemented persistent memory in transformer layer, and trained the model.
Austin Peng	Implementation, Data Visualization	Trained/validated the model for the YouTube Highlights dataset. Compared model output with [18] through plot visualizations.
Heidi Yap	Lion Implementation and Data Visualization	Helped Koby with implementing the Lion optimizer. Then trained/validated model for TVSum dataset and wrote the script for visualizing the loss curve for all of the metrics JSON files.
Koby Beard	Lion Implementation and Training	Implemented the Lion optimizer. Experimented with Lion vs. Adam optimizer. Also trained/validated the model for the TVSum dataset.

Table 5. Contributions of team members.

[11] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780, 2017. 5

[12] Fa-Ting Hong, Xuanteng Huang, Wei-Hong Li, and Wei-Shi Zheng. Mini-net: Multiple instance ranking network for video highlight detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 345–360, 2020. 2, 4

[13] Ali Karimi, Ramin Toosi, and Mohammad Ali Akhaee. Soccer event detection using deep learning, 2021. 1

[14] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017. 5

[15] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D. Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition, 2020. 5

[16] Jie Lei, Tamara L. Berg, and Mohit Bansal. Qvhighlights: Detecting moments and highlights in videos via natural language queries, 2021. 1, 2

[17] David Chuan-En Lin, Fabian Caba Heilbron, Joon-Young Lee, Oliver Wang, and Nikolas Martelaro. Videogenic: Video highlights via photogenic moments, 2022. 1

[18] Ye Liu, Siyuan Li, Yang Wu, Chang Wen Chen, Ying Shan, and Xiaohu Qie. Umt: Unified multi-modal transformers for joint video moment retrieval and highlight detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3042–3051, 2022. 2, 3, 4, 5, 6, 7

[19] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. 5

[20] 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. Learning transferable visual models from natural language supervision, 2021. 5

[21] Mrigank Rochan, Mahesh Kumar Krishna Reddy, and Yang Wang. Sentence guided temporal modulation for dynamic video thumbnail generation. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2020. 1

[22] Mrigank Rochan, Mahesh Kumar Krishna Reddy, Linwei Ye, and Yang Wang. Adaptive video highlight detection by learning from user history, 2020. 1

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 5

[24] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. Tvsum: Summarizing web videos using titles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5179–5187, 2015. 3

[25] Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. Augmenting self-attention with persistent memory, 2019. 3, 4

[26] Min Sun, Ali Farhadi, and Steve Seitz. Ranking domain-specific highlights by analyzing edited videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 787–802, 2014. 2

[27] Naoya Takahashi, Michael Gygli, and Luc Van Gool. Aenet: Learning deep audio features for video analysis, 2017. 1

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 3

[29] Phil Wang. Github: lion-pytorch. 5

[30] Phil Wang. Github: x-transformers. 4

[31] Weining Wang, Yan Huang, and Liang Wang. Language-driven temporal activity localization: A semantic matching reinforcement learning model. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 334–343, 2019. 2

[32] Yi Xu, Fan Bai, Yingxuan Shi, Qiuyu Chen, Longwen Gao, Kai Tian, Shuigeng Zhou, and Huyang Sun. Gif thumbnails: Attract more clicks to your videos. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(4):3074–3082, May 2021. 1

[33] Qinghao Ye, Xiyue Shen, Yuan Gao, Zirui Wang, Qi Bi, Ping Li, and Guang Yang. Temporal cue guided video highlight detection with low-rank audio-visual fusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7950–7959, 2021. 4

Appendix

4.1. Loss Curves

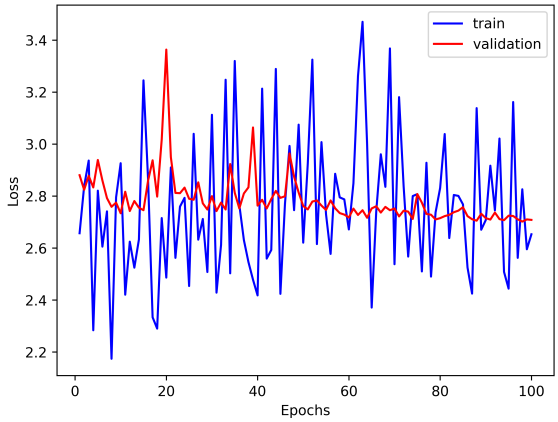

Figure 4. Loss Curve for Charades-VA

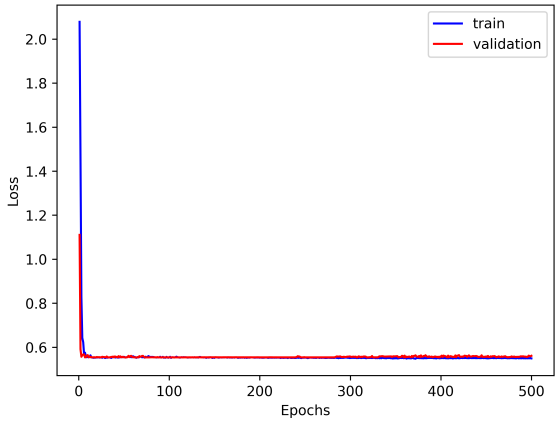

Figure 5. Loss Curve for TVSum-BK

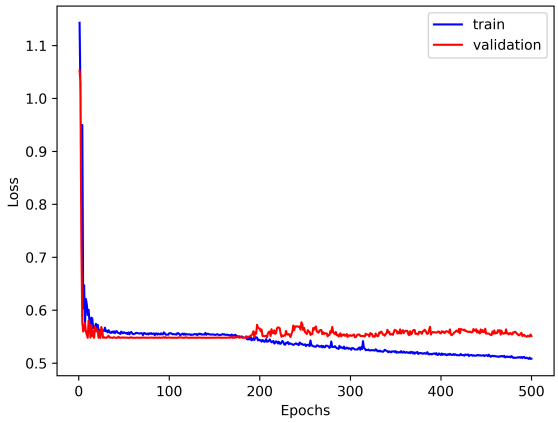

Figure 6. Loss Curve for TVSum-BT

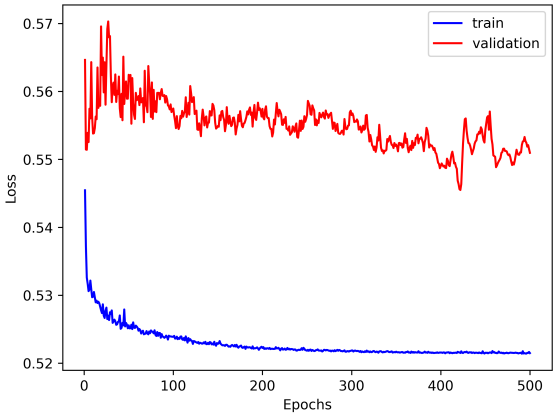

Figure 7. Loss Curve for TVSum-DS

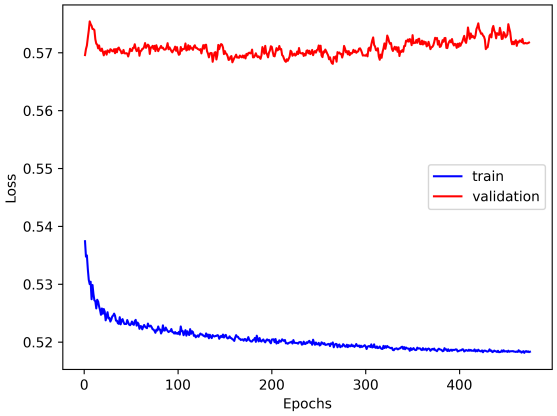

Figure 8. Loss Curve for TVSum-FM

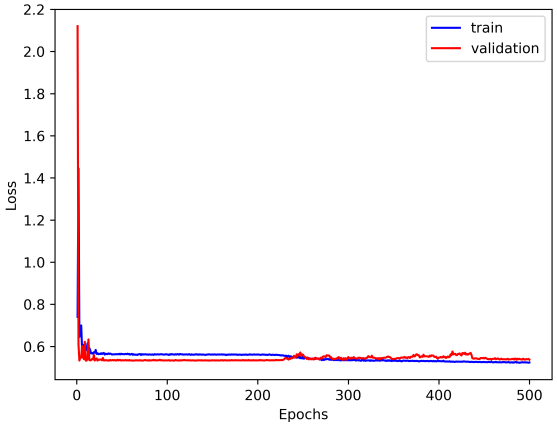

Figure 9. Loss Curve for TVSum-GA

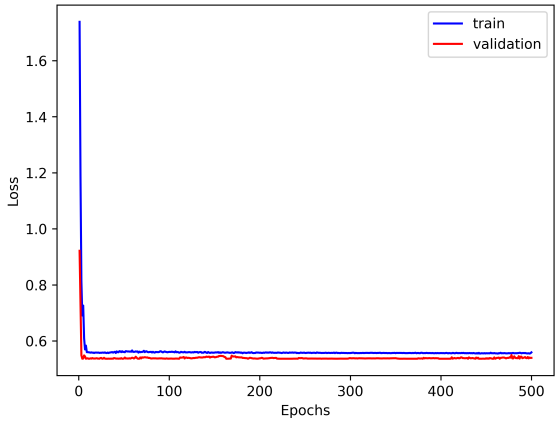

Figure 10. Loss Curve for TVSum-MS

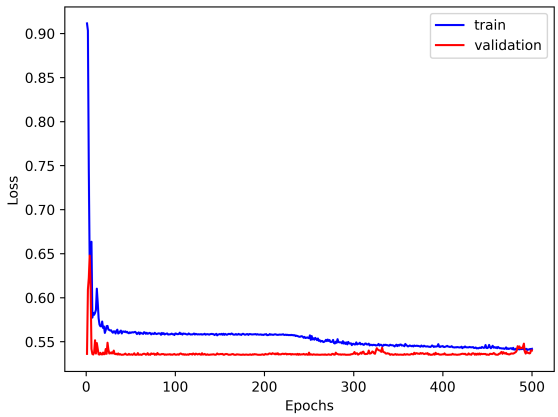

Figure 11. Loss Curve for TVSum-PK

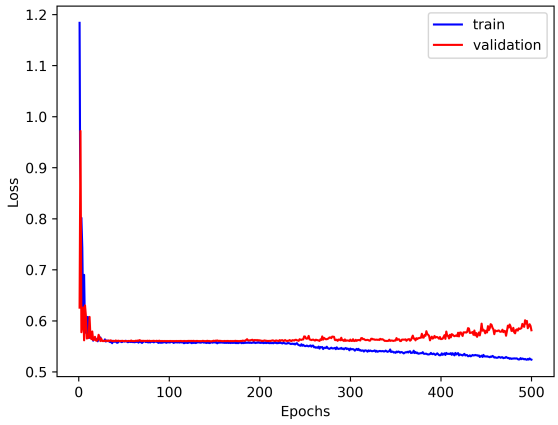

Figure 12. Loss Curve for TVSum-PR

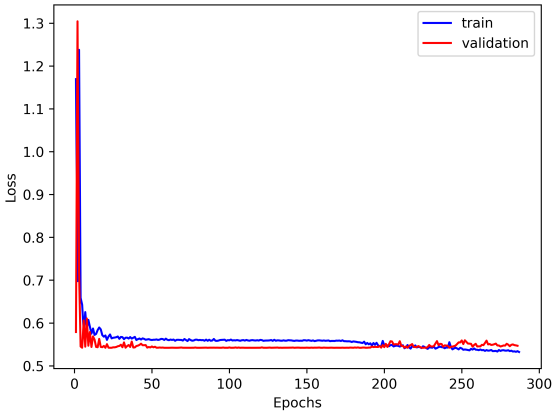

Figure 13. Loss Curve for TVSum-VT

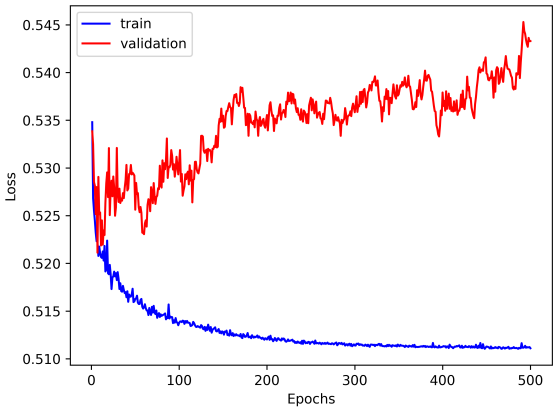

Figure 14. Loss Curve for TVSum-VU

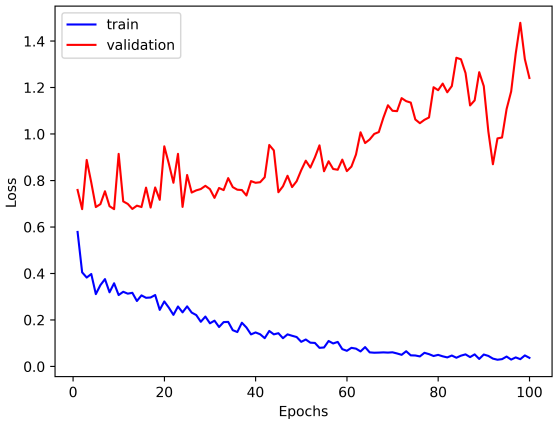

Figure 15. Loss Curve for YT-Dog

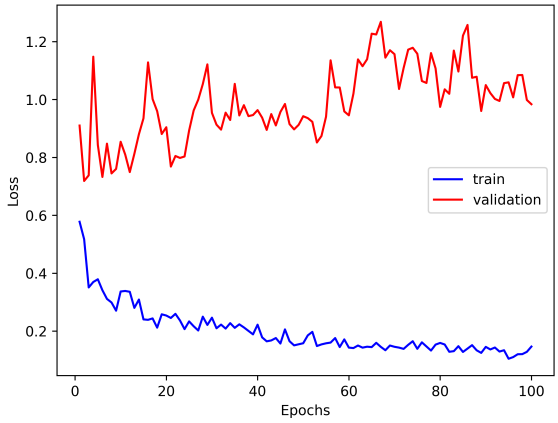

Figure 16. Loss Curve for YT-Gym

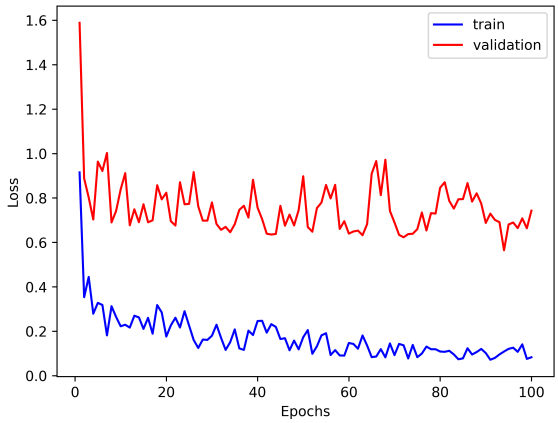

Figure 17. Loss Curve for YT-Par

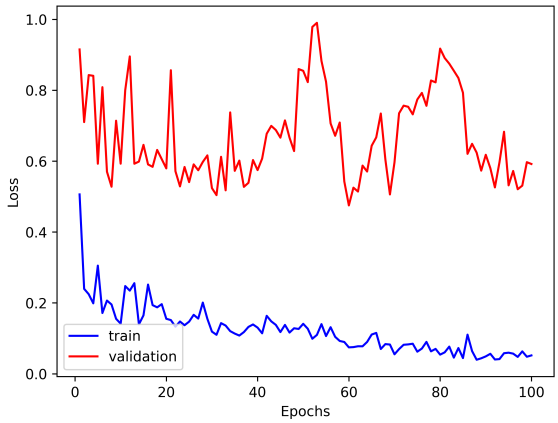

Figure 18. Loss Curve for YT-Ska

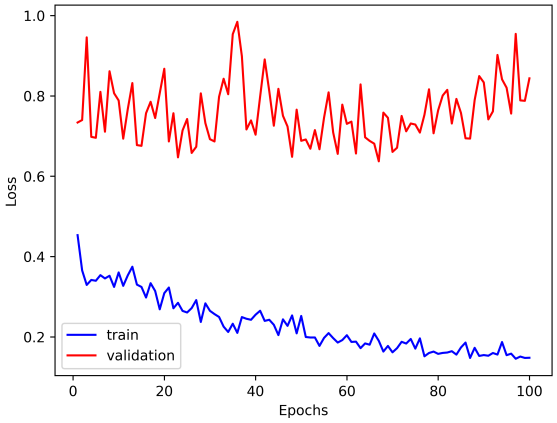

Figure 19. Loss Curve for YT-Ski

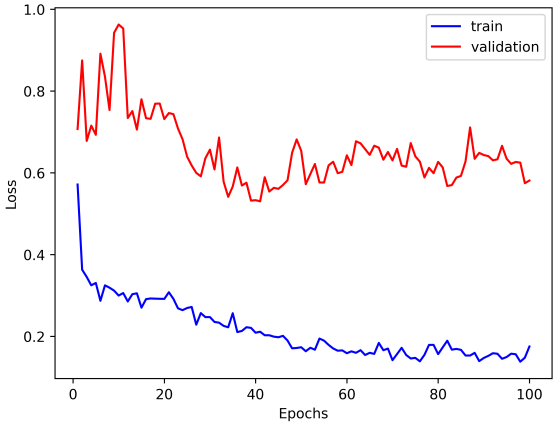

Figure 20. Loss Curve for YT-Sur