mmFilter: Language-Guided Video Analytics at the Edge

Zhiming Hu
Samsung AI Center
Toronto, Canada
zhiming.hu@samsung.com

Ning Ye
Samsung AI Center
Toronto, Canada
angela.ye@partner.samsung.com

Caleb Phillips
Samsung AI Center
Toronto, Canada
caleb.p@samsung.com

Tim Capes
Samsung AI Center
Toronto, Canada
t.capes@samsung.com

Iqbal Mohomed
Samsung AI Center
Toronto, Canada
i.mohomed@samsung.com

Abstract

Advances in deep learning have shown promising potential in scalable video analytics in the cloud. However, in constrained settings, it is not feasible to send every video frame from cameras at the edge. Being selective about which frames should be sent to the cloud reduces bandwidth consumption, conserves energy, and protects user privacy.

Existing solutions to edge-based filtering primarily focus on object detection and train binary classifiers to suppress transmission of irrelevant frames. This is both hard to scale when the queries of interest can change rapidly and infeasible for complex queries specified with natural language. In this paper, we propose mmFilter, a multimodal approach for filtering video streams matching predefined events (defined via natural language queries) at the edge. mmFilter learns compact representations for video and text data and automatically matches semantically similar pairs in a joint embedding space. Our model generalizes to various unseen queries, allowing new, complex queries to be added to the system in real time without retraining. We have implemented and evaluated our system on popular video datasets such as ActivityNet Captions and MSRVTT. mmFilter has shown a 1.5× improvement in event detection accuracy comparing with the state-of-the-art filtering approach.

CCS Concepts • **Computing methodologies → Machine learning approaches**; • **Computer systems organization → Distributed architectures**;

Middleware '20 Industrial Track, December 7–11, 2020, Delft, Netherlands

ACM ISBN 978-1-4503-8201-4/20/12…$15.00
https://doi.org/10.1145/3429357.3430518

Keywords edge computing, video processing

ACM Reference Format:
Zhiming Hu, Ning Ye, Caleb Phillips, Tim Capes, and Iqbal Mohomed. 2020. mmFilter: Language-Guided Video Analytics at the Edge. In *21st International Middleware Conference Industrial Track (Middleware '20 Industrial Track), December 7–11, 2020, Delft, Netherlands.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3429357.3430518

1 Introduction

Video camera deployments on edge devices are becoming increasingly prevalent. It is estimated that the market size of home security cameras will triple in the next seven years [3]. With millions of frames generated per camera each day, efficient video analytics are important for various applications. For example, live video feeds from home security cameras need to be processed in near real-time to automatically identify intruders or abnormal events and immediately alert the user. Home robots need to constantly survey the environment and instantly notify the user of any problems, such as a cat scratching the sofa or a burglar breaking in. Augmented reality (AR) glasses could warn vision-impaired wearers of nearby dangers. All these applications require the system to understand the relationship between the incoming frames and events of interest to the user or application.

However, uploading all the video content to the cloud for accurate detection of such events would not be viable because most edge devices are bandwidth-limited and energy-constrained. mmFilter serves to filter out uninformative video segments and pass only relevant frames to the cloud. Existing approaches such as NoScope [11] and FilterForward [5] train binary classifiers to suppress frames unrelated to a desired object class. However, both works are constrained to detecting object-based queries and require annotated training data for each object, making it impractical to dynamically support new queries.

In this paper, we propose a lightweight multimodal framework, mmFilter, to analyze the incoming frames and match them to predefined user queries. Specifically, we encode a

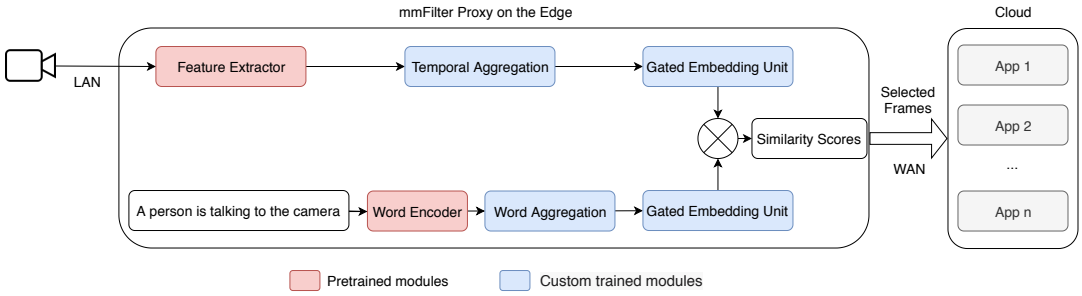


Figure 1. System design of mmFilter.

set of queries expressed by natural language and a window of incoming frames into fixed-sized embeddings, and use a trained multimodal model to map them into a joint embedding space. We can then compute cosine similarities between the video-text embeddings to determine the relevant frames to a query. mmFilter is able to generalize to unseen queries as similar queries would be in close proximity with one another in the shared embedding space. For purposes of illustration, suppose the feature extractors in mmFilter have been trained with instances of a cat and a couch, while the user expresses the query "[notify me if] my kitten scratches the sofa". Through consideration of semantic similarity between words (i.e. cat and kitten), a multimodal video-text matching approach can generalize much better to such an unseen query than an object-detection-based approach.

The main contributions of this paper are three-fold. First, we propose a lightweight architecture to detect and filter incoming frames based on user-specified events (in natural language) in camera-based streaming applications. Moreover, we demonstrate it is feasible to deploy the system on resource-constrained devices to analyze live video streams. Second, our system represents both the text and video data as compact embeddings, which allows highly complex queries to be used. We train the system only once to automatically generalize to new queries, whereas in existing edge-filtering solutions, a new classifier needs to be trained for each different query. Finally, we propose an efficient mapping algorithm to find the closest video and text embedding pair, which speeds up the conventional cosine similarity computation by 50% in some cases.

2 System Design

In this section, we introduce the system design of mmFilter, a novel approach for efficiently filtering camera-based video streams given a list of complex natural language user queries.

2.1 System Overview

An overview of the mmFilter architecture is described in Fig. 1. Video cameras stream their frames to the mmFilter proxy on the edge via a local area network (LAN) link. The mmFilter middleware internally comprises of four main components: a) encoding the inputs into high-dimensional feature vectors (also known as embeddings), b) aggregating the embeddings, c) mapping the features to a common embedding space, and d) performing efficient video-to-query matching. Given a live video input and a set of user-defined queries, mmFilter will output a set of scores relating each window of frames and query. One key difference to consider between the two inputs is that the frame features are extracted in real time while the queries are processed offline, as opposed to existing work on multimodal video retrieval [16, 19] in which both features are computed offline.

We train a model on large-scale video datasets offline to learn the joint embeddings for video and text such that once the features are projected into a joint space, frame and query embeddings that are semantically and contextually similar will be closer to one another. The distances between the embeddings can be computed using the cosine of the angle between them. If the similarity score meets an event detection threshold, the window of frames associated with the embeddings will be transmitted to the cloud for further processing (i.e., running a larger model for more accurate results). Note, we set the threshold for similarity scores empirically in our paper. How to dynamically tune the threshold for different videos will be explored in our future work. The objective of the system is to identify all the query-relevant frames on the edge node to upload, thereby reducing bandwidth used on the wide area link (WAN).

2.2 Video and Text Encoder

We consider two designs to process the live video feed. A simple approach would be to process each frame separately [5]. The features for each frame would be acquired and projected into the joint embedding space to compare with the query embeddings. Based on the similarity scores, or the embedding distances, we can detect precisely the frames that relate to the queries. To further reduce the noise, the result can be smoothed by adopting a k-voting algorithm over N consecutive frames. If the majority of the frames match a certain trigger, the entire window will be uploaded to the cloud.

Alternatively, we could compare the video frames with the queries using a per-window basis. Rather than computing a score of each frame with each event trigger, the features for a window of N consecutive frames are temporally aggregated using average pooling to produce a fixed length feature vector. As before, the feature vector will be projected into the joint space and compared with the queries. The cosine of the angle between the embeddings will decide whether to drop the entire window or send it to the cloud.

We investigate the effect of varying window sizes in frame-based and window-based approaches using event F1 [5] as a metric. Here, as both recall and precision are important for event detection, event F1 is used to balance the two metrics. As shown in Fig. 2, the latter method outperforms the former in all window sizes. Intuitively, this is because the aggregation module allows temporal correlations to be exploited. For example, consider a video frame of a hand covering a water bottle cap. An image alone raises ambiguity on whether the bottle is being tightened or loosened. Window-based approach also avoids redundant comparisons since individual scores do not need to be computed for consecutive frames of nearly identical features.

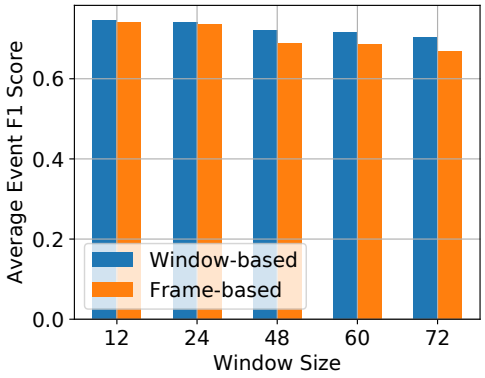


Figure 2. Window-based approach and frame-based approach on MSRVTT dataset (the higher the better).

For each query, we use a pretrained BERT model [7] to obtain the word embeddings for each word. The word vectors are then aggregated into a fixed-size feature vector using NetVLAD [4], whose weights are learned during training.

2.3 Gated Encoding Unit

Upon aggregating the video and text features into a single fixed-length vector, the output will be passed into a gated embedding unit (GEU). This unit maps the features into the joint embedding space and selects which raw features to be highlighted or suppressed.

The three-step process is shown in the following equations. Here $W_1 \in R^{d_2 x d_1}$, $W_2 \in R^{d_2 x d_2}$, $b_1 \in R^{d_2}$ and $b_2 \in R^{d_2}$ are learnable parameters given the dimension of the input features to be d_1 and dimension of the output to be d_2. σ is the element-wise sigmoid activation and $\circ$ is the element-wise multiplication.

$$E_1 = W_1 E_0 + b_1 \tag{1}$$

$$E_2 = E_1 \circ \sigma(W_2 E_1 + b_2) \tag{2}$$

$$E = \frac{E_2}{\|E_2\|_2} \tag{3}$$

Eq.(1) projects the features into the joint embedding space with a linear layer. The output is passed into a non-linear context gating layer [18, 19] shown in Eq.(2). Lastly, the result is L2-normalized as shown in Eq.(3). The motivation behind the gating function in Eq.(2) is two-fold. First, we wish to introduce non-linearities in our model. Second, we wish to reweight the strengths of the activations in E_1 to highlight and suppress information given the context of the video and query. Consider the query "a dog is biting the curtains" and a video clip of a dog doing so. Although the video frames may contain other objects such as furniture or outdoor scenery, the visual activations for these features are downweighted given the context of the query, while the features for the dog and the curtain are strengthened.

2.4 Approximate Nearest Neighbour Search

After the embeddings for a window of frames are mapped into the joint space, the last step is to search for the query with the highest cosine similarity score to the video segment. The simplest approach would be to return the highest score between the set of frames and every predefined trigger. However, this scales linearly with the number of queries and does not leverage the fact that the text embeddings are computed ahead of time.

Therefore, in mmFilter, we adopt the approximate nearest neighbour search system HNSW [17] to speed up the process of identifying the query with the highest cosine similarity score. The key idea in HNSW [17] is to build a hierarchical set of graphs (layers) with nested subsets of nodes in each layer. The links are separated into different layers by their distance scales. The algorithm has logarithmic time complexity, which reduces the overall computation cost and offers better scalability compared to the naive approach.

2.5 Loss Function

We train a lightweight model offline to jointly optimize the parameters in the video encoder and query encoder. With the normalized video and text embeddings obtained from the GEU, the cosine similarity score s_i^j between the i^{th} video v_i and j^{th} query q_j can be computed. Note, the ground truth query of the i^{th} video is the i^{th} query. Therefore, s_i^i denotes the similarity score for the i^{th} positive pair. We minimize a bidirectional max-margin ranking loss [21] to jointly learn the video-text representations:

$$L = \frac{1}{N} \sum_{i=1, j \neq i}^{N} \max(0, m + s_i^j - s_i^i) + \max(0, m + s_j^i - s_i^i) \,,$$

where N is the batch size and m is a tunable margin, which is set to 0.25 in this paper. The objective is to ensure that the similarity scores of positive pairs are larger than similarity scores of negative pairs by at least m.

3 Evaluation

This section evaluates the performance of mmFilter.

3.1 Experimental Setup

We use a lightweight MobileNetV2 [2] network pretrained on ImageNet [6] to extract the visual features from the incoming video frames. On the other hand, the embeddings for the predefined queries are encoded offline by BERT Large [7].

We evaluate mmFilter on an edge server and an NVIDIA Jetson Nano. The server has a 1080Ti GPU with 8GB of GPU memory, and an i9-7900X CPU @ 3.30GHz with ten cores.

3.1.1 Datasets

We trained our models on two widely used video datasets, ActivityNet Captions [13] and MSRVTT [23]. We followed the ActivityNet training splits proposed by [13] but with the validation set further divided in half to allocate samples for the test set. While ActivityNet has mostly distinct captions for each video, MSRVTT [23] contains multiple video clips associated with the same query. The abundance of data related to a specific caption enables us to evaluate against the existing filtering approach FilterForward [5], which detects relevant frames by using a separate "microclassifier" for each event. For our experiments, we specifically selected five events from MSRVTT and trained a new binary classifier for each query. We created 140 testing videos each comprising of one positive clip and four randomly selected video segments irrelevant to the queries.

3.1.2 Baselines

We evaluate mmFilter against two baselines.

FilterForward [5]. FilterForward [5] is the current state-of-the-art approach for detecting event matching frames to transmit to the cloud. For purposes of comparison, we attempt to recreate their setup by training MobileNet classifiers on the selected queries of MSRVTT to obtain a baseline.

mmFilter with other features. We compare mmFilter, which uses a lightweight, expressive MobileNet feature extractor, with motion, scene, and RGB features from a heavier, more complex model.

3.1.3 Metrics

The three metrics we explore in the paper are event F1 score, bandwidth consumption and runtime efficiency. The event F1 score proposed by [5] measures the accuracy of event detection in a video. Recall is redefined to adapt to an event i using the following parameters:

$$\text{Existence}_i = \begin{cases} 1 & \text{if detect any frame in event i} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Overlap}_i = \sum \frac{|Intersect(G_i, P_i)|}{|G_i|}$$

$$\text{EventRecall}_i = \alpha \times Existence_i + \beta \times Overlap_i$$

G_i and P_i are the ground truth frames and predicted frames for $event_i$. We also use the same settings as in FilterForward where α is 0.9 and β is 0.1. The parameters will highlight the importance of not missing an event, which is a crucial property for a filtering proxy.

For a given event, precision represents the fraction of correctly detected frames out of all detected frames. Finally, the event F1 score can be computed from the event-based recall and precision.

3.2 Experimental Results

We discuss the performance of mmFilter compared with the baseline approaches in terms of the event F1 score and bandwidth consumption. Furthermore, we measure the runtime of different components on an edge server and an NVIDIA Jetson Nano. Lastly, we show that our system has a minimal drop in accuracy with the addition of new queries.

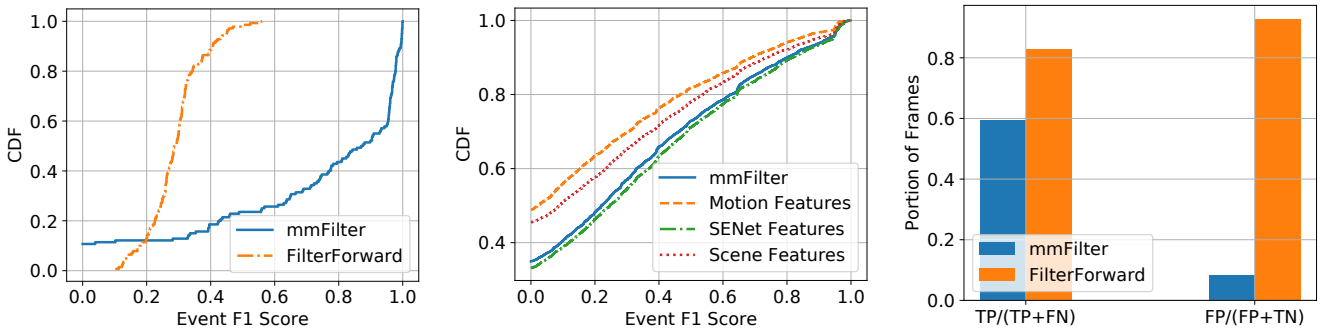
3.2.1 Event Detection Accuracy

Fig. 3(a) compares the event detection accuracy of mmFilter to the FilterForward design on the MSRVTT dataset. The average event F1 score for mmFilter is 0.72, significantly higher than the baseline score of 0.29 in FilterForward. Adopting FilterForward's binary classifier approach, 89% of events have an event F1 score of under 0.4 with no event scoring over 0.6. Conversely, in mmFilter, over 70% of the events have higher event F1 scores than 0.6.

The improvement in filtering ability is because mmFilter encodes the video and text data into high dimensional vector representations to learn semantically similar pairs in search for the best match. On the other hand, the binary classifiers could only output discrete labels to recognize simple objects, which is problematic even in the case of simple queries such as "a person is cooking".

3.2.2 Feature Extractor Selection

We assess the effectiveness of mmFilter by replacing the MobileNetV2 feature extractor with models from other modalities. We explore three other encoders: motion features from a 34-layer R(2+1)D model [22], scene features from a DenseNet-161 model [1] pretrained on Places365 [24], and RGB features from a more heavyweight SENet-154 model [10] pretrained



(a) mmFilter vs. FilterForward (the lower the better).

(b) mmFilter vs. other features (the lower the better). The motion, SENet and scene features are extracted offline.

(c) The percentage of frames that are sent to the cloud. (TP+FN) and (FP+TN) are the total number of positive frames and negative frames in the ground truth, respectively.

Figure 3. The performance of mmFilter comparing with the baselines.

on ImageNet [6]. The window size and threshold are set to be 48 and 0.2.

Fig. 3(b) shows the event F1 score on ActivityNet Captions for various encoders, with object features outperforming other modalities by a significant margin. Notably, 49% and 46% of events are missed by action and scene features, whereas only 35% and 33% of events are not captured by mmFilter and SENet-154. While a more powerful RGB model SENet-154 shows a 0.016 improvement in average F1 score compared to mmFilter, it takes 4.7 seconds to extract features for a batch size of 48 frames on the edge server's CPU. This is more than 20x slower than MobileNetV2, making it infeasible to deploy in live video settings.

3.2.3 Bandwidth Consumption

We compare the proportion of positive frames and negative frames sent to the cloud in mmFilter and FilterForward, where a positive frame is one that is related to a user query. Fig. 3(c) shows the result averaged over the 140 test videos from MSRVTT. For positive frames, mmFilter is able to detect 60% correctly while FilterForward successfully detects over 80%. However, the binary classifiers adopted in FilterForward are unable to learn simple queries such as "a man is talking" and mistakenly identify nearly all the frames as positive. Hence, the false positive rate in FilterForward is 11× higher than mmFilter. Overall, our approach reduces bandwidth consumption by around 80% compared to FilterForward.

3.2.4 Approximate Nearest Neighbor Search

To further improve efficiency, we adopt an approximate nearest neighbour search to return the most related query to a window of frames. We compare the results with a conventional cosine similarity approach. Evaluation is performed by varying the number of predefined queries and comparing the runtime between the two methods. The results shown are averaged over 1,000 runs and processed on the CPU on a Jetson Nano. Note that both the visual and query embeddings are normalized after the gated embedding layer; hence, cosine similarity simplifies to a matrix multiplication calculation between the two vectors. A linear search is performed to obtain the match with the highest score.

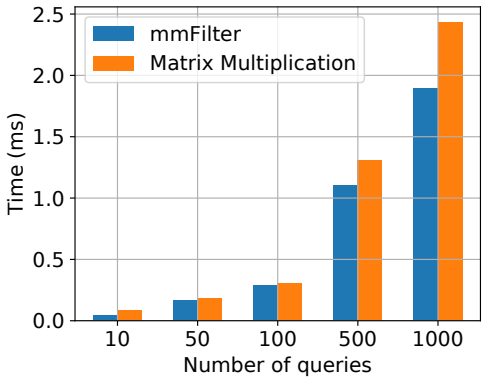


Figure 4. Matching time for mmFilter on Jetson Nano.

As shown in Fig. 4, mmFilter reduces runtime costs by 50% and 25% for 10 and 1000 queries, respectively. mmFilter creates a graph based on the distances of the query embedding when they are first instantiated. Instead of conducting a linear search between the video embedding and all the queries, the time complexity is reduced to logarithmic.

3.2.5 Runtime Characteristics

Table 1 presents the runtime characteristics of different components in our system. We consider two CPU platforms: the edge server and Jetson Nano. On the edge server, the execution times are averaged over the ActivityNet Captions test set and evaluated over a batch of 10 queries. The time spent on video processing modules are for 48 frames, yielding a throughput of as high as 220 frames per second (fps).

On the Jetson Nano, visual feature extraction for a batch of 48 frames finished in 3.236 seconds, or around 15 fps. The

Table 1. Runtime for the components in mmFilter (ms). GEU is short for Gated Embedding Unit. TA is for Temporal Aggregation.

Platforms	Feature Extractor	TA	GEU (video)	Word Encoder	Word Aggregation	GEU (text)
Edge Server	218.267	0.223	0.252	1399.972	4.779	10.005
Jetson Nano	3236.323	1.279	2.666	5884.325	30.702	98.525

temporal aggregation and gated embedding unit completed in the order of milliseconds. The time for text-related modules is taken for a batch of 3 queries. Recall that these could all be run offline and do not impact live video processing time, therefore we selected a large BERT model to optimize for performance.

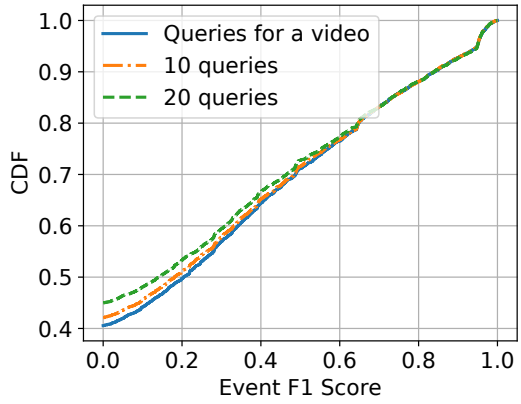


Figure 5. Event F1 score with different number of queries.

3.2.6 Performance with More Queries

As the number of predefined queries grows, so does the difficulty in identifying the correct match to the incoming frames. Fig. 5 compares the performance of mmFilter with different number of queries. The baseline uses only queries associated with a video, in which each video has approximately 3 captions on average. In the 10- and 20-query settings, we randomly inject irrelevant queries to the predefined list. We see a slight performance drop with the addition of new queries, but overall our model can achieve similar performance compared to the baseline. Queries irrelevant to the video frames will be farther away in the embedding space, and hence will not affect the similarity search results. This is critical in practical settings such as AR glasses applications, where a user may request multiple queries to be identified, but only a few or none are present in an incoming video feed.

4 Related Work

This section discusses works related to filtering live video streams and matching queries to images/videos.

Frame Filtering. NoScope [11] and Filterforward [5] are two recent filtering-based approaches for dropping video frames that do not contain objects of interest. They both operate on training distinct binary classifiers for each individual query. In this paper, we seek to directly map the frame features and text features into a joint transformed space using only one feature extractor for each component. By learning a fixed-vector representation for both video and text, we are able to generalize to a multitude of diverse, novel queries without the need for retraining.

Reducto [15] aims to filter frames based on cheap and low level vision features such as edges and corners. However, it only focuses on three kinds of queries: tagging objects, counting queries and bounding box detection. Our approach, mmFilter, focuses on detecting events specified by queries with rich, natural language descriptions.

Image-Text Matching. Image-text matching is the problem of measuring the semantic similarity between visual data and a sentence descriptor. VSE++ [8] is based on image features and proposes a new loss emphasizing hard negative samples. Conversely, SCAN [14] and RankMI [12] propose a region-based attention mechanism where region features are extracted from object detectors. In mmFilter, we focus on image features to reduce the computational overhead.

Video-Text Matching. Video-text matching is an extension of image-text matching that has also been actively researched in recent years. In MoEE [19], the model calculates the similarity scores between each expert in different modalities and the query and aggregates the weighted similarity scores. Collaborative experts [16] proposes a collaborative gating module to highlight or suppress the raw expert features from different modalities. Both approaches need to extract the video features offline and use a computationally more expensive RGB feature extractor. Finally, Rekall [9] and Lookout [20] adopt scripts to define the events of interests instead of using queries expressed by natural language.

5 Concluding Remarks

Limited bandwidth between edge and cloud makes it infeasible to offload all the incoming frames from the camera to the cloud. In this paper, we propose mmFilter, a multimodal frame filtering approach that efficiently detects frames relevant to natural language queries by jointly mapping them to a common space. Adopting this technique, we are not limited to simple object detection queries, and we can support unseen user queries without retraining the model or annotating sample data for each newly introduced query. Comparing with the state of the art approach, we have shown an improvement of 1.5× in the average event F1 score. mmFilter opens up the possibility of performing large-scale video analytics on edge devices involving complex and dynamic queries.

References

[1] 2020. Dense Convolutional Network (DenseNet). http://shorturl.at/oJW58 Accessed: 2020-09-11.

[2] 2020. MobileNetV2 Feature Extrator. https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/4 Accessed: 2020-09-01.

[3] 2020. Smart Home Security Cameras Market Size, Share & Trends Analysis Report. http://shorturl.at/grAY7 Accessed: 2020-09-11.

[4] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. 2016. NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5297–5307.

[5] Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David G Andersen, Michael Kaminsky, and Subramanya R Dulloor. 2019. Scaling Video Analytics on Constrained Edge Nodes. *arXiv preprint arXiv:1905.13536* (2019).

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A Large-Scale Hierarchical Image Database. In *Proc. of IEEE CVPR*. 248–255.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* (2018).

[8] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2017. VSE++: Improving Visual-Semantic Embeddings with Hard Negatives. *arXiv preprint arXiv:1707.05612* (2017).

[9] Daniel Y Fu, Will Crichton, James Hong, Xinwei Yao, Haotian Zhang, Anh Truong, Avanika Narayan, Maneesh Agrawala, Christopher Ré, and Kayvon Fatahalian. 2019. Rekall: Specifying Video Events using Compositions of Spatiotemporal Labels. *arXiv preprint arXiv:1910.02993* (2019).

[10] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-Excitation Networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[11] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. Noscope: Optimizing Neural Network Queries over Video at Scale. *arXiv preprint arXiv:1703.02529* (2017).

[12] Kemertas, Mete and Pishdad, Leila and Derpanis, Konstantinos G. and Fazly, Afsaneh. 2020. RankMI: A Mutual Information Maximizing Ranking Loss. In *Proc. of the IEEE/CVF CVPR*.

[13] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-Captioning Events in Videos. In *Proceedings of the IEEE International Conference on Computer Vision*. 706–715.

[14] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. 2018. Stacked Cross Attention for Image-Text Matching. In *Proc. of ECCV*. 201–216.

[15] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. 2020. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. In *Proc. of ACM SIGCOMM*. 359–376.

[16] Y. Liu, S. Albanie, A. Nagrani, and A. Zisserman. [n. d.]. Use What You Have: Video Retrieval using Representations from Collaborative Experts. In *arXiv preprint arxiv:1907.13487* (2019).

[17] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2014. Approximate Nearest Neighbor Algorithm based on Navigable Small World Graphs. *Information Systems* 45 (2014), 61–68.

[18] Antoine Miech, Ivan Laptev, and Josef Sivic. 2017. Learnable Pooling with Context Gating for Video Classification. *arXiv preprint arXiv:1706.06905* (2017).

[19] Antoine Miech, Ivan Laptev, and Josef Sivic. 2018. Learning a Text-Video Embedding from Incomplete and Heterogeneous Data. *arXiv:1804.02516* (2018).

[20] Lenin Ravindranath, Matthai Philipose, Peter Bodik, and Paramvir Bahl. 2017. Demo: Live Video Stream Triggers. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '17)*. Association for Computing Machinery, New York, NY, USA, 195. https://doi.org/10.1145/3081333.3089340

[21] Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded Compositional Semantics for Finding and Describing Images with Sentences. *Transactions of the Association for Computational Linguistics* 2 (2014), 207–218.

[22] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 6450–6459.

[23] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. MSRVTT: A Large Video Description Dataset for Bridging Video and Language. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5288–5296.

[24] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 Million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).