Video in 10 Bits: Few-Bit VideoQA for Efficiency and Privacy

Shiyuan Huang[1], Robinson Piramuthu[2], Shih-Fu Chang[1], and Gunnar A. Sigurdsson[2]

[1] Columbia University
[2] Amazon Alexa AI

Abstract. In Video Question Answering (VideoQA), answering general questions about a video requires its visual information. Yet, video often contains redundant information irrelevant to the VideoQA task. For example, if the task is only to answer questions similar to "Is someone laughing in the video?", then all other information can be discarded. This paper investigates how many bits are really needed from the video in order to do VideoQA by introducing a novel *Few-Bit VideoQA* problem, where the goal is to accomplish VideoQA with few bits of video information (e.g., 10 bits). We propose a simple yet effective task-specific feature compression approach to solve this problem. Specifically, we insert a lightweight **Feat**ure **Comp**ression Module (**FeatComp**) into a VideoQA model which learns to extract task-specific tiny features as little as 10 bits, which are optimal for answering certain types of questions. We demonstrate more than 100,000-fold storage efficiency over MPEG4-encoded videos and 1,000-fold over regular floating point features, with just 2.0−6.6% absolute loss in accuracy, which is a surprising and novel finding. Finally, we analyze what the learned tiny features capture and demonstrate that they have eliminated most of the non-task-specific information, and introduce a Bit Activation Map to visualize what information is being stored. This decreases the privacy risk of data by providing k-anonymity and robustness to feature-inversion techniques, which can influence the machine learning community, allowing us to store data with privacy guarantees while still performing the task effectively.

Keywords: Video Question Answering, Feature Compression, Privacy, Applications.

1 Introduction

Video data exemplifies various challenges with machine learning data: It is large and has privacy issues (*e.g.* faces and license plates). For example, an autonomous robot driving around may collect gigabytes of video data every hour, quickly filling up all available storage with potentially privacy-sensitive information. Yet, video is useful for a variety of computer vision applications, *e.g.* action detection [25], object tracking [31], and video question answering (VideoQA) [44,

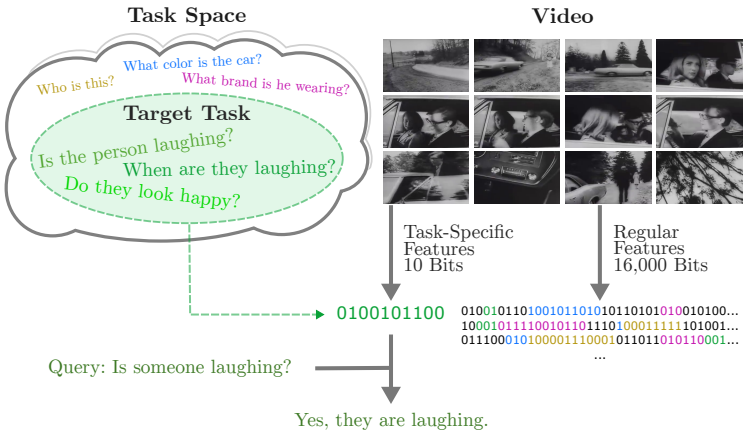


Fig. 1: We introduce the problem of *Few-Bit VideoQA* where only few bits of video information is allowed to do VideoQA tasks. Our proposed method compresses the features inside a neural network to an extreme degree: A video input of 1 MB can be compressed down to 10 bits, and still solve common question-answering tasks with a high degree of accuracy, such as "Is someone laughing?". This has both storage and privacy implications. Here the *Target Task* is all questions related to *laughing*, a subset of all possible tasks in the *Task Space*. *Regular Features* have additional task-irrelevant information visualized with questions and bits of corresponding color. *Regular Features* corresponds to 512 floating point features from a state-of-the-art video network, such as [40].

19]. With increasing adoption of computer vision applications, and mobile devices, efficiently storing video data becomes an increasingly important problem.

VideoQA [44, 19] is a general machine learning task that requires analyzing video content to answer an general question. As such, it contains elements of multiple video problems, such as classification, retrieval, and localization. The questions in VideoQA are not all available beforehand, which means that the model needs to extract general semantic information that is still useful enough to answer a diverse set of questions, such as any question about the task "Human Activities", for example. Concretely, the system may have been trained on queries including "Is the person laughing?", and at inference time, the system needs to record and store a video, and may later receive the query "Do they look happy?" which was not seen at training time. In contrast with classification problems where the system could potentially store the answer to all possible queries, a VideoQA system needs to keep enough information to answer any query about the video.

While early video analysis used hand-crafted visual descriptors to represent videos, recent advances in deep neural networks are able to learn highly abstracted features [3, 30, 32]. These features still contain more information than what is actually needed in a specific task. For example, a popular video encoder network ResNet3D [40] extracts a 512-dim floating number feature, which re-

Table 1: Overview of different levels of feature storage and privacy limitations.

	Data Amount	Can Identify User?	Can Discriminate User?
Original Data	1,000,000 bits (e.g. MPEG4 video)	Yes	Yes
Regular Feature	16,000 bits (e.g. 512 floats)	Yes (Can be inverted)	Yes
Regular Compressed Feature	100 bits	No	Yes
Task-Compressed Feature	10 bits	No	No (k-Anonymity [33])

quires 16,384 bits. If the task is to only answer questions about the happiness of a person, such as "do people look happy", we theoretically only need 1-bit of information to accomplish that task by encoding the existence of person + laughing. A single 16,384-bit feature encodes much more information than required to answer this question. Moreover, it is hard to interpret what information is captured by continuous features, and hence hard to guarantee that the features do not contain any sensitive information, which may carry privacy concerns.

Much of recent work on VideoQA focuses on learning stronger vision features, improved architectures, or designing better multi-modal interaction [8, 36, 22, 29, 20]. This paper instead investigates how many bits are really needed from video in current VideoQA tasks. To this end, we introduce a novel "*Few-Bit VideoQA*" problem, which aims to accomplish VideoQA where only few bits of information from video are allowed. To our knowledge, the study of few-bit features is an understudied problem, with applications to storing and cataloging large amounts of data for use by machine learning applications.

We provide a simple yet effective task-specific compression approach towards this problem. Our method is inspired by recent learning-based image and video coding [27, 24, 13, 7, 9, 14, 4, 38, 39, 43], which learns low-level compression with the goal of optimizing visual quality. In contrast, our method looks at compressing high-level features, as shown in Figure 1. Given a video understanding task, we compress the deep video features into few bits (e.g., 10 bits) to accomplish the task. Specifically, we utilize a generic **Feat**ure **Comp**ression module (**FeatComp**), which can be inserted into neural networks for end-to-end training. FeatComp learns compressed binarized features that are optimized towards the target task. In this way, our task-specific feature compression can achieve a high compression ratio and also address the issue of privacy. This approach can store large amounts of features on-device or in the cloud, limiting privacy issues for stored features, or transmitting only privacy-robust features from a device. Note that this work is orthogonal to improvements made by improved architectures, shows insights into analyzing how much video data is needed for a given VideoQA dataset, and provides a novel way to significantly optimize storage and privacy risks in machine learning applications.

10-bits is a concrete number we use throughout this paper to demonstrate the advantage of tiny features. While 10-bits seems like too little to do anything useful, we surprisingly find that predicting these bits can narrow down the solution space enough such that the model can correctly pick among different answers in VideoQA [44, 19]. Different tasks require different number of bits, and we see in

Section 4 that there are different losses of accuracy for different tasks at the same number of bits. Storing only 10 bits, we can assure that the stored data does not contain any classes of sensitive information that would require more than 10 bits to be stored. For example, we can use the threshold of 33 bits (8 billion unique values) as a rule-of-thumb threshold where the features stop being able to discriminate between people in the world. In Table 1 we show different levels of features and how they compare to this threshold. At the highest level, *Original Data*, we would have the full image or video with all their privacy limitations.[3] Even with feature extraction, various techniques exist to invert the model and reconstruct the original data [10, 28]. Using *Regular Compressed Features* (*i.e.* off-the-shelf compression on features in Section 5.1 or task-independent compression in Section 4.1) would still pose some privacy threats. Our *Task-Compressed Features*, provide privacy guarantees from their minimal size.

We experiment on public VideoQA datasets to analyze how many bits are needed for VideoQA tasks. In our experiments, the "**Task**" is typically question types in a specific dataset, such as TGIF-Action [17], but in a practical system this could be a small set of the most common queries, or even a single type of question such as "do people look happy in the video?". Note that while the "Task" of all questions in a single dataset might seem very diverse and difficult to compress, the task is much more narrow than any possible question about any information in the video, such as "What color is the car?", "What actor is in the video?", etc. In summary, our contributions can be summarized as:

1. We introduce a novel Few-Bit VideoQA problem, where only few bits of video information is used for VideoQA; and we propose a simple yet effective task-specific feature compression approach that learns to extract task-specific tiny features of very few bits.
2. Extensive study of how many bits of information are needed for VideoQA. We demonstrate that we lose just 2.0%−6.6% in accuracy using only 10 bits of data. It provides a new perspective of understanding how much visual information helps in VideoQA.
3. We validate and analyze the task specificity of the learned tiny features, and demonstrate their storage efficiency and privacy advantages.

The outline of the paper is as follows. In Section 2, we review related work in VideoQA, image/video coding and deep video representations. In Section 3 we introduce the Few-Bit VideoQA problem, and our simple solution with feature compression. In Section 4 we discuss several experiments to analyze and validate our approach. Finally, in Section 5 we demonstrate applications of this novel problem, including distributing tiny versions of popular datasets and privacy.

2 Related Work

Video Question Answering VideoQA is a challenging task that requires the system to output answers given a video and a related question [45, 17, 44, 21, 37].

[3] In the MSRVTT-QA dataset the videos are 630KB on average, which consists of 320×240-resolution videos, 15s on average.

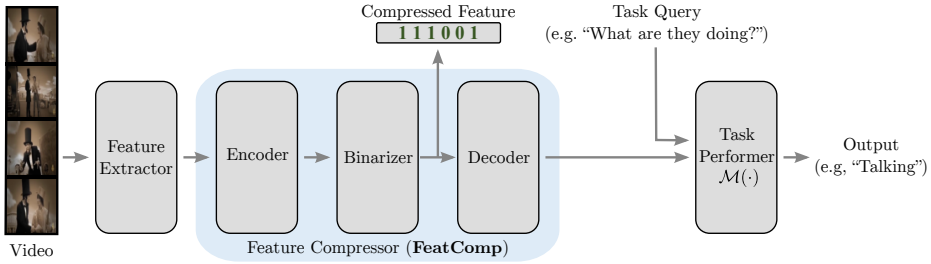


Fig. 2: Pipeline of our generic feature compression approach (**FeatComp**) towards Few-Bit VideoQA, which follows the procedure of encoding, binarization and decoding. It has learned to only encode information relevant to the questions of interest through task-specific training.

Recent approaches include multi-modal transformer models [22, 20] and graph convolutional networks [15]. We instead look at VideoQA under limited bits, which shares some philosophy with work that has looked at how much the visual content is needed for Visual Question Answering [11].

Image and Video Coding Image and video coding is a widely studied problem which aims to compress image/video data with minimum loss of human perceptual quality. In the past decades, standard video codecs like HEVC [35], AVC [42], image codes like JPEG [41] have been used for compressing image/video data. More recently, learning-based image/video compression [27, 24, 13, 7, 9, 14, 4, 6, 12] has been proposed to replace the codec components with deep neural networks that optimize the entire coding framework end-to-end, to achieve better compression ratios. All of these existing approaches compress with the goal of pixel-level reconstruction. Our approach is inspired by these works but is applied under a different context — we aim to solve a novel Few-bit VideoQA problem.

Deep Video Representation Deep neural networks have been shown effective to learn compact video representation, which is now a favored way to store video data for machine learning applications. With the emergence of large-scale video datasets like Kinetics [3] and HowTo100M [30], recent advances in representation learning [8, 36, 22, 29, 20] extract continuous video features which contain rich semantic information. Such pre-computed video features can be successfully applied to video understanding tasks like action detection, action segmentation, video question answering, etc [22, 29, 20]. However, deep video features could still contain more information than what's actually needed by a specific task. In this work, we instead focus on learning tiny video features, where we aim to use few bits of data to accomplish the target task.

3 Few-Bit VideoQA

In this section, we first establish the problem of Few-Bit VideoQA; then provide a simple and generic task-specific compression solution; finally we introduce our simple implementation based on a state-of-the-art VideoQA model.

3.1 Problem Formulation

In a standard VideoQA framework, a feature extractor (e.g., ResNet3D [40]) is applied to a video sequence to extract the video embedding x, which is a high-level and compact representation of the video, normally a vector composed of floating point numbers. We write $\mathcal{M}(\cdot)$ as the VideoQA task performer, and the output from $\mathcal{M}(x, q)$ is the predicted answer in text, where q refers to the text embedding of the question. In our context, we assume $\mathcal{M}(\cdot)$ is a neural network that can be trained with an associated task objective function $\mathcal{L}_{\text{task}}$.

Though the compact feature x already has a much smaller size compared to the original pixel data, it still raises storage and privacy concerns as discussed in the introduction. To this end, we introduce a novel problem of Few-Bit VideoQA, where the goal is to accomplish VideoQA tasks with only few bits of visual information, i.e., we want to perform $\mathcal{M}(x, q)$ with the size of x less than N bits, where N is small (e.g., $N = 10$).

3.2 Approach: Task-Specific Feature Compression

We propose a simple yet effective approach towards the problem of Few-Bit VideoQA. As shown in Figure 2, we insert a feature compression bottleneck (**FeatComp**) between the video feature extractor and task performer $\mathcal{M}$ to compress x. We borrow ideas from compression approaches in image/video coding [38, 39, 43]. But rather than learning pixel-level reconstruction, we train the compression module solely from a video task loss, which has not been explored before to our knowledge. In fact, FeatComp is a generic module that could be applied to other machine learning tasks as well.

Encoding and Decoding FeatComp follows the encode-binarize-decode procedure to transform floating-point features into binary, and then decode back to floating-point that can be fed into the task performer. In encoding, we first project x to the target dimension, $x' = f_{\text{enc}}(x)$, where $x' \in \mathbb{R}^N$ and N is the predefined bit level to compress into. Binarization is applied directly on feature values; so we map the feature values to a fixed range. We use batch normalization (BN) [16] to encourage bit variance, and then a hyperbolic function $tanh(\cdot)$ to convert all the values to $[-1, 1]$. Binarization is inherently a non-differentiable operation, in order to incorporate it in the learning process, we use stochastic binarization [38, 39, 43] during training. The final equation for the encode-binarize-decode procedure is:

$$x_{\text{dec}} = \text{FeatComp}(x) = \left(f_{\text{dec}} \circ \text{BIN} \circ \tanh \circ \text{BN} \circ f_{\text{enc}}\right)(x) \tag{1}$$

where $\circ$ denotes function composition. The output after the binarization step is $x_{\text{bin}} \in \{0,1\}^N$, which is the N-bit compressed feature to be stored.

Learning Task-Specific Compression Our FeatComp is generic and can be inserted between any usual feature extractor and task performer $\mathcal{M}(\cdot)$. The task performer will instead take the decoded feature to operate the task: $\mathcal{M}(x_{\text{dec}})$ or $\mathcal{M}(x_{\text{dec}}, q)$. To compress in a task-specific way, FeatComp is trained along with $\mathcal{M}(\cdot)$ with the objective $\mathcal{L} = \mathcal{L}_{\text{task}}$, where $\mathcal{L}_{\text{task}}$ is the target task objective.

Simple Implementation Our FeatComp can be easily implemented into any VideoQA models. As an instantiation, we choose a recent state-of-the-art model ClipBERT [20] as our baseline, and add our compression module to study the number of bits required for VideoQA. Specifically, ClipBERT follows the similar pipeline as in Figure 2. We then insert FeatComp after feature extractor. For encoding and decoding, we use a fully connected layer where $f_{\text{enc}}\colon \mathbb{R}^{(T\times h\times w\times D)} \mapsto \mathbb{R}^N$ and $f_{\text{dec}}\colon \{0,1\}^N \mapsto \mathbb{R}^{(T\times h\times w\times D)}$. x is flattened to a single vector to be encoded and binarized, and the decoded x_{dec} can be reshaped to the original size. Then the answer is predicted by $\mathcal{M}(x_{\text{dec}}, q)$. For FeatComp with N-bit compression, we write it as FeatComp-N. This implementation is simple and generic, and as we see in Section 4 already works surprisingly well. Various other architectures for encoding, decoding, and binarization could be explored in future research.

Intuition of FeatComp To learn a compression of the features, various methods could be explored. We could cluster the videos or the most common answers into 1024 clusters, and encode the cluster ID in 10 bits, etc. For a task such as video action classification, where only the top class prediction is needed, directly encoding the final answer may work. However, in a general video-language task such as VideoQA, the number of possible questions is much more than 1024, even for questions about a limited topic. Our method can be interpreted as learning 2^N clusters end-to-end, that are predictable, and useful to answering any questions related to the task.

4 Experiments

In this section, we show the experimental results on Few-Bit VideoQA, study how much visual information is needed for different VideoQA tasks, and analyze what the bits capture.

Datasets We consider two public VideoQA datasets: 1) TGIF-QA [17] consists 72K GIF videos, 3.0s on average, and 165K QA pairs. We experiment on 3 TGIF-QA tasks — Action (*e.g.* "What does the woman do 5 times?"), Transition (*e.g.* "What does the man do after talking?"), which are multiple-choice questions with 5 candidate answers; and FrameQA (*e.g.* What does an airplane drop which bursts into flames?"), which contains general questions with single-word answers. 2) MSRVTT-QA [44] consists of 10k videos of duration 10−30s each and 254K general QA pairs, *e.g.* "What are three people sitting on?".

Implementation Details We leverage the ClipBERT model pre-trained on COCO Captions [5] and Visual Genome Captions [18], and train on each VideoQA

dataset separately, following [20]. During training, we randomly initialize FeatComp, and finetune the rest of the network; we randomly sample T=1 clip for TGIF-QA and T=4 clips for MSRVTT-QA, where in each clip we only sample the middle frame. We fix the ResNet backbone and set the learning rate to 5×10^{-5} for FeatComp, and 10^{-6} for $\mathcal{M}$. We use the same VideoQA objective and AdamW [26] optimizer as in [20]. During inference, we uniformly sample T_{test} clips to predict answer, where T_{test}=T, unless noted otherwise. We set N=1, 2, 4, 10, 100, 1000 for different bit levels. Code will be made available.

Evaluation Metrics QA accuracy at different bit levels.

Baselines To our knowledge, there is no prior work directly comparable; hence we define the following baselines:

- *Floats*: the original floating-point-based ClipBERT; it provides performance upper bound using enough bits of information .[4]
- *Q-only*: answer prediction solely from question.
- *Random Guess*: randomly choose a candidate/English word for multiple-choice/single-word-answer QA.

Additionally, since our approach follows an autoencoder-style design, we also study whether an objective of feature reconstruction helps with Few-Bit VideoQA. We add the following two approaches for comparison:

- *R-only*: learn FeatComp solely with a reconstruction loss, $\mathcal{L}_R$=MSE(x, x_{dec}), then finetune $\mathcal{M}$ using learned compressed features with task objective.
- *FeatComp+R*: learn FeatComp from both task and reconstruction objectives, i.e., $\mathcal{L}$=$\mathcal{L}_{task}$+$\mathcal{L}_R$.

We also study how test-time sampling affects the results with:

- *4×FeatComp*: test-time sampling T_{test}=4T.

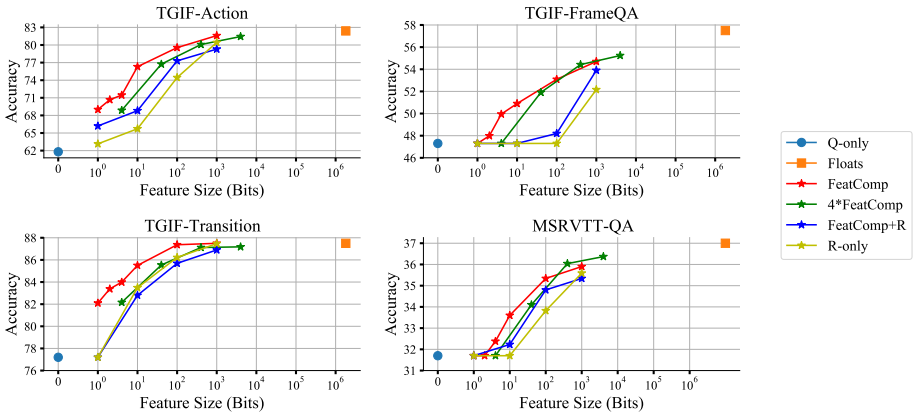
4.1 Few-Bit VideoQA Results

We demonstrate our Few-Bit VideoQA results in Figure 3 and compare to the baselines. As expected, more bits yields accuracy improvements. Notably, for our *FeatComp*, even a 1-bit video encoding yields a 7.2% improvement over *Q-only* on TGIF-Action. On all datasets, at 1000-bit, we maintain a performance drop <2.8% while compressing more than >1000 times. At 10-bit, the drop is within 2.0−6.6% while compressing over >100,000 times, demonstrating 10-bit visual information already provides significant aid in VideoQA tasks.

We also compute *supervision size* as a naive upper bound of bits required to accomplish the task. Theoretically, the compressed features should be able to differentiate the texts in all QA pairs. Hence, for each task, we use bzip2[5] to compress the text file for the training QA pairs, whose size is used as the upper bound. We observe correspondence between supervision size and compression difficulty. TGIF-Action and TGIF-Transition are easier to compress, requiring less bits to get substantial performance gains; and they also appear to require less

[4] ClipBERT uses 16-bit precision, we use this for calculation.

[5] https://www.sourceware.org/bzip2



Method	Bits	TGIF-Action	TGIF-FrameQA	TGIF-Transition	MSRVTT-QA
Random Guess	0	20.0	0.05	20.0	0.07
Q-only	0	61.8	47.3	77.2	31.7
R-only	10	63.7	47.3	83.5	31.7
FeatComp+R	10	75.8	47.4	82.8	32.2
FeatComp	10	76.3	50.9	85.5	33.6
Floats	1.8M - 9.8M	82.4 (82.9)	57.5 (59.4)	87.5 (87.5)	37.0 (37.0)
Supervision Size		30.9 Bits	59.2 Bits	57.7 Bits	59.5 Bits

Fig. 3: Analysis of how bit size affects VideoQA accuracy on MSRVTT-QA and TGIF-QA. *Floats* refers to the original ClipBERT model that serves as an upper bound of the performance without bit constraints. We both report the number we reproduced and cite paper results in parenthesis. With our simple approach *FeatComp*, we can reach high performance using only a few bits. Notably, at 10-bit level (see table), we get only 2.0−6.6% absolute loss in accuracy.

bits in supervision size. Instead TGIF-FrameQA and MSRVTT-QA are harder to compress, also reflected by their relatively larger supervision size.

Role of Reconstruction Loss Our approach FeatComp is learned in a task-specific way in order to remove any unnecessary information. On the other hand, it is also natural to compress with the goal of recovering full data information. Here we investigate whether direct feature supervision helps learn better compressed features from *FeatComp+R* and *R-only*. We can see that integrating feature reconstruction harms the accuracy at every bit level. In fact, R-only can be considered as a traditional lossy data compression that tries to recover full data values. It implies that recovering feature values requires more information than performing VideoQA task; hence feature reconstruction loss may bring task-irrelevant information that hurts the task performance.

Role of Temporal Context For longer videos, frame sampling is often crucial for task accuracy. We study the impact in Figure 3 (*FeatComp* vs. *4×FeatComp*). We can see denser sampling benefits compression ratio especially at higher bits on longer videos like MSRVTT-QA. For example, 400-bit compression with T_{test}=16

Table 2: Task-specificity of FeatComp-10. Compression learned from an irrelevant source task is less helpful for a target task, while compression from the same task gives the best performance.

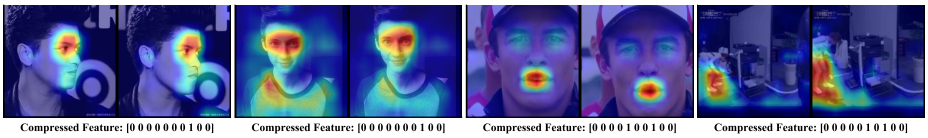
	Target Task		
Source Task	Action	FrameQA	Transition
Action	**76.9**	47.6	83.8
FrameQA	62.3	**51.7**	77.1
Transition	76.0	47.3	**85.0**
Q-Only	61.8	47.3	77.2

outperforms 1000-bit compression with T_{test}=4. However, on short video dataset TGIF-QA, it does not yield improvements, which implies TGIF-QA videos can be well understood with single frames.

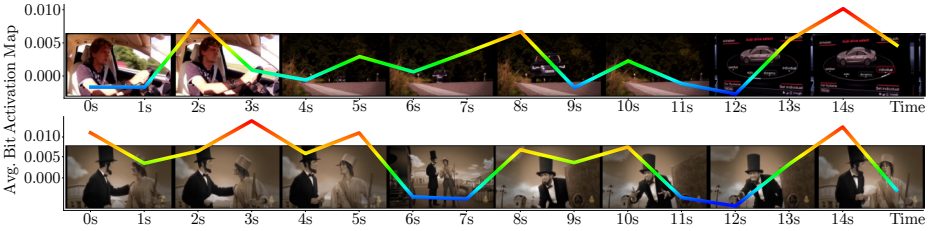
How Task-Specific are the Features? Here we evaluate the task-specificity by analyzing how much information is removed by the compression. We use FeatComp-10 learned on a source TGIF-QA task to extract the compressed features x_{bin}; then train a new VideoQA model for different target tasks on top of x_{bin}. For fair comparison, we use the same $\mathcal{M}$ and decoding layers and follow the previous practice to initialize $\mathcal{M}$ from the model pre-trained on COCO Captions and Visual Genome, and randomly initialize the decoding layers. Then we train the network with learning rate 5×10^{-5} for 20 epochs. Table 2 shows the results, where we also cite *Q-only* from Figure 3. Compression from the same task gives the best result as expected. Note that *Action* and *Transition* are similar tasks in that they query for similar actions but *Transition* asks change of actions over time. *FrameQA* instead queries for objects, which has minimal similarity with *Action*/*Transition*. Compression from a highly relevant source task (*Action* vs. *Transition*) gives pretty high performance. However, compression from an irrelevant source task (*Action*/*Transition* vs. *FrameQA*) yields similar performance as *Q-only* (0-bits), which again implies that the learned compression discards any information unnecessary for its source task.

4.2 Qualitative Analysis

Here we study qualitatively what the learned bits are capturing. We apply Grad-CAM [34] directly on the compressed features x_{bin} to find the salient regions over frames. Grad-CAM is originally a tool for localizing the regions sensitive to class prediction. It computes the weighted average of feature maps based on its gradients from the target class, which localizes the regions that contribute most to that class prediction. Here we instead construct *Bit Activation Map* (BAM) by treating each binary bit x^i_{bin} as a "class": 1 is a positive class while 0 is negative. We calculate BAM at the last convolutional layer of ResNet. We average BAM for all bits where for x^i_{bin}=0 we multiply them with -1. Note that no class annotations, labels or predictions are used for BAM. Figure 4a shows



(a) Bit Activation Map w.r.t. FeatComp-10 compressed features x_{bin} on TGIF-Action. The learned tiny feature is able to capture salient regions useful for the source task of FeatComp-10.



(b) Averaged Bit Activation Map over video frames on MSRVTT-QA. We can see that learned tiny feature can capture scene changes — the bit activation peaks appear when there is a different scene useful for the source task of FeatComp-10.

Fig. 4: Qualitative visualization examples of bit activation map w.r.t FeatComp-10 compressed features x_{bin}. Note that no question, answer, prediction, or class label was used for these visualizations—This is visualizing what the network used to generically compress the video.

heat map visualization results on TGIF-Action for FeatComp-10. The learned compression is capturing salient regions (e.g., eyes, lips, legs) which align with human perception. We also study in Figure 4b how important temporal signals are captured, where we average the BAM over frames. The frames with high BAM scores tend to capture more important semantic information across time that is relevant to the task.

Additionally, we investigated whether the bits capture task-specific information by analyzing the correspondences between VideoQA vocabularies and compressed features. We calculate a word feature as the averaged FeatComp-10 bit features over the videos whose associated QAs contain this word, and find its top-7 closest words based on Euclidean distance. Table 3 shows some sample results on TGIF-Action. Neighbor words tend to demonstrate semantic associations. E.g., 'eyes' is closely related to 'blink', 'smile' and 'laugh'; 'step' is associated with its similar actions like 'walk' and 'jump', implying that the compression captures task-relevant semantic information.

5 Applications of Few-Bit VideoQA

The problem of Few-Bit VideoQA has practical applications to data storage efficiency and privacy. Below we discuss how our approach can compress a video

Table 3: Top-7 closest words from FeatComp-10 on TGIF-Action.

Word	Most Similar Words
head	bob neck stroke fingers raise cigarette open
wave	touch man bob hands hand stroke raise
spin	around ice pants jump foot kick knee
step	walk spin around ice pants jump kick
guitar	object who keys black a strum bang
eyes	blink smile laugh cigarette lip tilt sleeve

dataset into a tiny dataset and use that to achieve the same task (Section 5.1); then we demonstrate the privacy advantages of few-bit features (Section 5.2).

5.1 Tiny Datasets

With our task-specific compression approach, we can represent a video using only a few bits, which allows extreme compression of gigabytes of video datasets into almost nothing. For example, TGIF-QA contains 72K videos whose MPEG4-encoded format takes up about 125GB of storage; with FeatComp-10, we end up with a 90KB dataset. We follow Section 4.1 to train a model using stored compressed features x_{bin}, with source and target tasks being the same, to evaluate the feature quality. Table 4 compares the compression size and testing accuracy on TGIF-QA and MSRVTT-QA tasks. Our tiny datasets achieve similar performance with Figure 3 at all bit levels. In addition to MPEG4 video and uncompressed Floats, we also report the size of Floats compressed with the off-the-shelf lossless compression standard ZIP.[6] The result implies that the original features indeed contain a lot of information not easily compressible, and we are learning meaningful compression.

5.2 Privacy Advantages from Tiny Features

Here we demonstrate how our tiny features offer privacy advantages.

Advantages of Data Minimization. Intuitively, we expect 10 bits of information to not contain very much sensitive information. The principle of Shannon Information[7], or the pigeonhole principle, tells us that 10 bits of information cannot contain a full image (approximately 10KB). Using 10 bits as an example, we can divide sensitive information into two groups based on if it can be captured by 10 bits or not in Table 5.

Note we assume the data was not in the training set as otherwise the model could be used to extract that potentially sensitive information (training networks

[6] `numpy.savez_compressed`
[7] en.wikipedia.org/wiki/Information_content

Table 4: VideoQA accuracies with our compressed datasets at different bit levels. Looking at 10-bit, we can get 100,000-fold storage efficiency, while maintaining good performance.

Datasets	TGIF-QA				MSRVTT-QA			
	Size	Task Accs.			Size	Task Accs.		
		Action	FrameQA	Transition		What	Who	All
1-bit set	9KB	68.3	47.3	82.4	1.3KB	24.8	37.7	31.8
10-bit set	90KB	76.9	51.7	85.0	13KB	27.4	44.1	33.7
1000-bit set	9MB	80.8	54.4	87.2	1.3MB	28.3	46.0	34.9
Floats set	16.2GB	82.4	57.5	87.5	12.3GB	31.7	45.6	37.0
Comp. Floats set	14.0GB	82.4	57.5	87.5	9.5GB	31.7	45.6	37.0
MPEG4 set	125GB	82.4	57.5	87.5	6.3GB	31.7	45.6	37.0

Impossible (bits)	Possible (bits)
Credit Card Num. (≈53)	Gender (≈2)
Social Security Num. (≈30)	Skin Color (≈3)
Street Address (≈28)	Social Class (≈3)
License Plate (≈36)	...
Personal Image (≈100,000)	
Phone Number (≈33)	
...	

Table 5: What sensitive information can be stored in 10 bits?

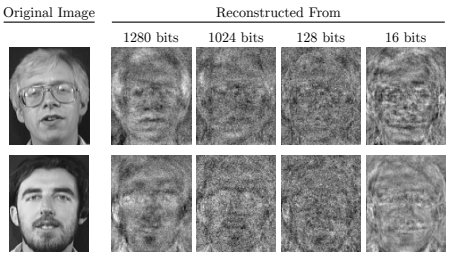


Fig. 5: Feature inversion with increasingly compressed features.

with differential privacy can relax this assumption [1]). By capturing only 10 bits, we can assure a user that the stored data does not contain any classes of sensitive information that require more bits to be stored.

Furthermore, considering that there are 8B unique people in the world, the identity of a person in the world can be captured in 33 bits, so we cannot reconstruct the identity of a person from 10 bits. The identity can be identifying information, photographic identity, biometrics, etc. This is consistent with the following experiment where feature-inversion techniques do not seem to work, whereas they work on regular compressed 16,384 bit features (as in Table 1). This effectively de-identifies the data.

Robustness to Feature Inversion Storing only a few bits also has applications to preventing reconstruction of input from stored features (*i.e.* Feature Inversion). Various methods exist for inverting features [28, 10] typically by optimizing an input to match the feature output, with an optional regularization. To evaluate this, we started with a re-implementation[8] of the Frederikson *et al.*

[8] github.com/Koukyosyumei/secure_ml

attack [10] for model inversion, but instead of class probability, we used a MSE loss between the target feature and the model feature output before the last linear layer, or after the binarizer. We used a two layer neural network trained on the AT&T Faces Dataset [2]. In Figure 5 we demonstrate how the inversion deteriorates as the model uses fewer bits. The 1,280 bits result has no compression (40 32-bit numbers) whereas the rest of the results have increasing compression. All models were trained until at least 97.5% accuracy on the training set.

Feature Quantization In Frederikson *et al*. [10] the authors note that rounding the floating point numbers at the 0.01 level seems to make reconstruction difficult and offer that as mitigation strategy. This suggests that compression of features will likely help defend against model inversion and we verify this here. For comparison, rounding a 32-bit float (less than 1) at the 0.01 level corresponds to 200 different numbers which can be encoded in 8 bits, a 4-fold compression. Our method can compress 512 32-bit floats (16,384 bits) into 10 bits, a 1,638-fold compression, while having a defined performance on the original task.

k-Anonymity of Tiny Features For a system that has N users and stores 10 bits (1,024 unique values), we can assure the user that:

> Any data that is stored is indistinguishable from the data from approximately $N/1{,}024$ other users.

This ensures privacy by implementing k-anonymity [33] assuming the 10 bits are uniformly distributed, for example if $N{=}10^7$, $k{\approx}10{,}000$. Finally, we can use a variable number of stored bits to ensure any stored bits are sufficiently non-unique, similar to hash-based k-anonymity for password checking [23].

6 Conclusion

In this work, we introduce a novel Few-Bit VideoQA problem, which aims to do VideoQA with only few bits of video information used; we propose a simple and generic FeatComp approach that can be used to learn task-specific tiny features. We experiment over VideoQA benchmarks and demonstrate a surprising finding that a video can be effectively compressed using as little as 10 bits towards accomplishing a task, providing a new perspective of understanding how much visual information helps in VideoQA. Furthermore, we demonstrate the storage, efficiency, and privacy advantages of task-specific tiny features — tiny features ensure no sensitive information is contained while still offering good task performance. We hope these results will influence the community by offering insight into how much visual information is used by question answering systems, and opening up new applications for storing large amounts of features on-device or in the cloud, limiting privacy issues for stored features, or transmitting only privacy-robust features from a device.

7 Acknowledgement

We would like to thank Vicente Ordonez for his valuable feedback on the manuscript.

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (2016)
2. Cambridge, A.L.: The orl database. http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
3. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: CVPR (2017)
4. Chadha, A., Andreopoulos, Y.: Deep perceptual preprocessing for video coding. In: CVPR (2021)
5. Chen, X., Fang, H., Lin, T.Y., Vedantam, R., Gupta, S., Dollár, P., Zitnick, C.L.: Microsoft coco captions: Data collection and evaluation server. arXiv (2015)
6. Choi, J., Han, B.: Task-aware quantization network for jpeg image compression. In: ECCV (2020)
7. Djelouah, A., Campos, J., Schaub-Meyer, S., Schroers, C.: Neural inter-frame compression for video coding. In: ICCV (2019)
8. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: ICCV (2019)
9. Feng, R., Wu, Y., Guo, Z., Zhang, Z., Chen, Z.: Learned video compression with feature-level residuals. In: CVPR Workshops (2020)
10. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security (2015)
11. Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., Parikh, D.: Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In: CVPR (July 2017)
12. He, T., Sun, S., Guo, Z., Chen, Z.: Beyond coding: Detection-driven image compression with semantically structured bit-stream. In: PCS (2019)
13. Hu, Z., Chen, Z., Xu, D., Lu, G., Ouyang, W., Gu, S.: Improving deep video compression by resolution-adaptive flow coding. ECCV (2020)
14. Hu, Z., Lu, G., Xu, D.: Fvc: A new framework towards deep video compression in feature space. In: CVPR (2021)
15. Huang, D., Chen, P., Zeng, R., Du, Q., Tan, M., Gan, C.: Location-aware graph convolutional networks for video question answering. In: AAAI (2020)
16. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML (2015)
17. Jang, Y., Song, Y., Yu, Y., Kim, Y., Kim, G.: Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In: CVPR (2017)
18. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., et al.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. IJCV (2017)
19. Le, T.M., Le, V., Venkatesh, S., Tran, T.: Hierarchical conditional relation networks for video question answering. In: CVPR (2020)
20. Lei, J., Li, L., Zhou, L., Gan, Z., Berg, T.L., Bansal, M., Liu, J.: Less is more: Clipbert for video-and-language learning via sparse sampling. In: CVPR (2021)
21. Lei, J., Yu, L., Bansal, M., Berg, T.L.: Tvqa: Localized, compositional video question answering. arXiv preprint arXiv:1809.01696 (2018)
22. Li, L., Chen, Y.C., Cheng, Y., Gan, Z., Yu, L., Liu, J.: Hero: Hierarchical encoder for video+ language omni-representation pre-training. In: EMNLP (2020)

23. Li, L., Pal, B., Ali, J., Sullivan, N., Chatterjee, R., Ristenpart, T.: Protocols for checking compromised credentials. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (2019)
24. Lin, J., Liu, D., Li, H., Wu, F.: M-LVC: Multiple frames prediction for learned video compression. In: CVPR (2020)
25. Lin, T., Liu, X., Li, X., Ding, E., Wen, S.: Bmn: Boundary-matching network for temporal action proposal generation. In: ECCV (2019)
26. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
27. Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., Gao, Z.: DVC: An end-to-end deep video compression framework. In: CVPR (2019)
28. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: CVPR (2015)
29. Miech, A., Alayrac, J.B., Smaira, L., Laptev, I., Sivic, J., Zisserman, A.: End-to-end learning of visual representations from uncurated instructional videos. In: CVPR (2020)
30. Miech, A., Zhukov, D., Alayrac, J.B., Tapaswi, M., Laptev, I., Sivic, J.: HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In: ICCV (2019)
31. Minderer, M., Sun, C., Villegas, R., Cole, F., Murphy, K., Lee, H.: Unsupervised learning of object structure and dynamics from videos. arXiv preprint arXiv:1906.07889 (2019)
32. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV (2015)
33. Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression (1998)
34. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: ICCV (2017)
35. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. IEEE Transactions on circuits and systems for video technology (2012)
36. Sun, C., Myers, A., Vondrick, C., Murphy, K., Schmid, C.: Videobert: A joint model for video and language representation learning. In: ICCV (2019)
37. Tapaswi, M., Zhu, Y., Stiefelhagen, R., Torralba, A., Urtasun, R., Fidler, S.: MovieQA: Understanding Stories in Movies through Question-Answering. In: CVPR (2016)
38. Toderici, G., O'Malley, S.M., Hwang, S.J., Vincent, D., Minnen, D., Baluja, S., Covell, M., Sukthankar, R.: Variable rate image compression with recurrent neural networks. arXiv preprint arXiv:1511.06085 (2015)
39. Toderici, G., Vincent, D., Johnston, N., Jin Hwang, S., Minnen, D., Shor, J., Covell, M.: Full resolution image compression with recurrent neural networks. In: CVPR (2017)
40. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: CVPR (2018)
41. Wallace, G.K.: The JPEG still picture compression standard. IEEE transactions on consumer electronics (1992)
42. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Transactions on circuits and systems for video technology (2003)

43. Wu, C.Y., Singhal, N., Krahenbuhl, P.: Video compression through image interpolation. In: ECCV (2018)
44. Xu, D., Zhao, Z., Xiao, J., Wu, F., Zhang, H., He, X., Zhuang, Y.: Video question answering via gradually refined attention over appearance and motion. In: ACM MM (2017)
45. Zhu, L., Xu, Z., Yang, Y., Hauptmann, A.: Uncovering the temporal context for video question answering. IJCV (2017)