FFNet: Video Fast-Forwarding via Reinforcement Learning

Shuyue Lan [1] Rameswar Panda [2] Qi Zhu [1] Amit K. Roy-Chowdhury [2]

[1]Northwestern University. {slan@u., qzhu@}northwestern.edu

[2]University of California, Riverside. {rpand002@, amitrc@ece.}ucr.edu

Abstract

For many applications with limited computation, communication, storage and energy resources, there is an imperative need of computer vision methods that could select an informative subset of the input video for efficient processing at or near real time. In the literature, there are two relevant groups of approaches: generating a "trailer" for a video or fast-forwarding while watching/processing the video. The first group is supported by video summarization techniques, which require processing of the entire video to select an important subset for showing to users. In the second group, current fast-forwarding methods depend on either manual control or automatic adaptation of playback speed, which often do not present an accurate representation and may still require processing of every frame. In this paper, we introduce FastForwardNet (FFNet), a reinforcement learning agent that gets inspiration from video summarization and does fast-forwarding differently. It is an online framework that automatically fast-forwards a video and presents a representative subset of frames to users on the fly. It does not require processing the entire video, but just the portion that is selected by the fast-forward agent, which makes the process very computationally efficient. The online nature of our proposed method also enables the users to begin fast-forwarding at any point of the video. Experiments on two real-world datasets demonstrate that our method can provide better representation of the input video (about 6%-20% improvement on coverage of important frames) with much less processing requirement (more than 80% reduction in the number of frames processed).

1. Introduction

Leveraging video input has become increasingly important in many intelligent Internet-of-Things (IoT) applications, such as environment monitoring, search and rescue, smart surveillance, and wearable devices. In these systems, large amount of video needs to be collected and processed by users (human operators or autonomous agents), either locally or remotely through network transmission (or a combination of both). For better system performance, the processing often needs to be done at or near real time. On the other hand, the local nodes/devices typically have limited computation and storage capability and often run on batteries, while the communication network is constrained by bandwidth, speed and reliability [1, 19, 40]. Such discrepancy presents an urgent need for new vision methods that can automatically select an informative subset of the input video for processing, to reduce computation, communication and storage requirements and to conserve energy.

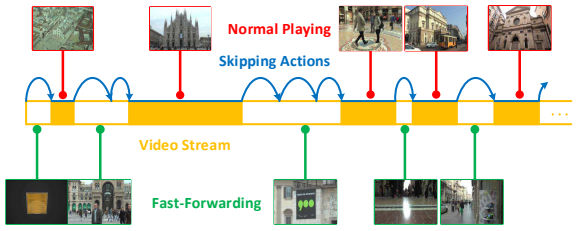

Figure 1. **Overview of Our Proposed Method.** Given a video stream, our FFNet decides which frame to process next and presents it to users while *skipping* the irrelevant frames in an online manner. Top-row shows the representative frames in the normal playing portion and bottom-row shows the irrelevant frames in the fast-forwarding portion. Best viewed in color.

In the relevant literature, with ever expanding volume of video data, there is significant interest in video summarization techniques, which compute a short and informative subset of the original video for human consumption or further processing [4, 9, 26, 49, 50, 51]. However, these techniques require the processing of entire video and often take a long time to generate the subset. There are also video fast-forwarding techniques where the playback speed of the video is adjusted to meet the needs of users [3, 10, 13, 31, 32, 34, 38], but they often do not present an accurate representation and may still require processing of the entire video. Both types of approaches are not suitable for the resource-limited and time-critical systems we discussed above. To address this problem, we started by asking the following question: *Is it possible to develop a*

method for fast-forwarding through a video that is computationally efficient, causal, online and results in informative segments which can be validated through statistical evaluation and user experience?

In this paper, we introduce FastForwardNet (FFNet), a reinforcement learning agent that gets inspiration from video summarization and does fast-forwarding differently from the state-of-the-art methods. It has an online framework that automatically fast-forwards a video and presents a selected subset of frames to users on the fly (see Fig. 1 for an example). The fast-forward agent does not require the processing of entire video. This makes the process very computationally efficient, and communicationally efficient if the video (subset) needs to be transmitted over the network for remote processing. The online nature of our proposed FFNet enables the users to begin fast-forwarding at any point when watching/processing videos. The causal nature of our FFNet ensures that it can work even as the video subset is being generated.

To summarize, the key advantage of our approach is that it automatically selects the most important frames ***without*** processing or even obtaining the entire video. Such capability can significantly reduce resource requirements and lower energy consumption, and is particularly important for resource-constrained and time-critical systems. The main technical contributions of this paper are as follows.

(1) We formulate video fast-forwarding as a Markov decision process (MDP), and propose FFNet for fast-forwarding a potentially very long video while presenting its important and interesting content on the fly.

(2) We propose an online framework to deal with incremental observations without requiring to store and process the entire video. At any point of the video, our approach can jump to potentially important future frames based on analysis of past frames that had been selected.

(3) We demonstrate the effectiveness of our proposed FFNet on two standard challenging video summarization datasets, Tour20 [27] and TVSum [41], achieving real-time speed on all tested videos.

2. Related Work

Our work relates to three major research directions: video fast-forward, video summarization and reinforcement learning. Here, we focus on some representative methods that are closely related to our work.

Video Fast-Forward. Video fast-forward methods are typically used when users are losing patience to watch the entire video. Most commercial video players offer manual control on the playback speed, e.g., Apple QuickTime Player offers 2, 5 and 10 multi-speed fast-forward.

Current automatic fast-forward approaches mostly focus on adapting the playback speed based on either the similarity of each candidate clip to the query clip [31] or the motion activity patterns present in a video [3, 29, 30]. Some recent works use mutual information between frames to describe the fast-forward policy [11, 12], or use shortest path distance over the graph that is constructed with semantic information extracted from frames [34, 38]. This family of methods is most relevant to our goal. A similar family of work (hyperlapse) [32, 10, 13] aiming at speed-up and smoothing has also been developed for creating fast-forwarded videos. In contrast to these prior works, we develop a deep reinforcement learning strategy for the fast-forward policy. Our proposed framework (FFNet) is an online and causal system that does not need the entire video to get the fast-forward policy, making it very efficient in terms of computation, communication and storage needs.

Video Summarization. The goal of video summarization is to produce a compact summary that contains the most important parts of a video. Much progress has been made to summarize a video using either supervised learning based on video-summary pairs [6, 9, 49, 50, 33, 26] or unsupervised approaches based on low-level visual indices [5, 8, 20, 7] (see reviews [24, 43]). Leveraging crawled web images or videos is another recent trend for video summarization [14, 15, 41, 28]. Closely related to video summarization, the authors in [37] develop a framework for creating storylines from photo albums.

Most relevant to our approach is the work of online video summarization, which compiles the most salient and informative portion of a video by automatically scanning through the video stream, in an online fashion, to remove repetitive and uninteresting content. Various strategies have been studied, including Gaussian mixture model [25], online dictionary learning [51], and submodular optimization [4]. Our approach significantly differs from these methods in that it only processes a subset of frames instead of the entire video. To the best of our knowledge, this is the first work to address video fast-forwarding in generating an informative summary from a video.

Reinforcement Learning. Apart from the recent success in playing Go games and Atari [23, 39], deep reinforcement learning(DRL) has also achieved promising performance in several vision tasks, such as object detection [21], visual tracking [48], pose estimation [17] and image captioning [35]. [47] employs a computationally intensive reinforcement learning strategy for action detection in short video clips. In contrast to [47], our framework is an online and causal system that enables users to begin fast-forwarding at any point while watching videos (online) and can work even as summary is being generated (causal). Markov Decision Process (MDP) has been widely used for several vision tasks. For example, in [42], the authors formulate a policy learning as MDP for activity recognition. Q-learning (a reinforcement learning method) is one way to

solve MDP problems [45]. In the proposed FFNet, we use a multi-layer neural network to represent the Q-value function, similar to [36, 46]. We are not aware of any prior work in reinforcement learning that deals with fast-forwarding while summarizing long duration videos.

3. Methodology

In this section, we provide the details of FFNet. We start with an overview of our approach in Sec. 3.1, present detailed formulations in Sec. 3.2 and Sec. 3.3, and then explain the training algorithm in 3.4.

3.1. Solution Overview

Our goal is to fast-forward a long video sequence by skipping unimportant frames in an online and real-time fashion (see Fig. 1). Given the current frame being processed, the goal of FFNet is to decide the number of frames to skip next. Those frames within the skipping interval will not be processed. Then, the video frames we present to users include the frames processed by FFNet and their neighboring windows (which are not processed).

We formulate the above fast-forwarding problem using a Markov decision process (MDP) and develop our FFNet as a reinforcement learning agent, i.e., a Q-learning agent that learns a policy to skip unimportant frames and present the important ones to users. During test time, given a raw video, fast-forwarding is a sequential process. At each step $k = 1, ..., K$ of an episode, we process the current frame, decide how many future frames to skip, and jump to the frame after the skipped ones for next processing. We present the processed frames and their neighboring ones (with processed frames as window centroids) to users as important subsets of the video.

3.2. MDP Formulation for FFNet

We consider fast-forwarding as a control problem that can be formulated as an MDP with the following elements.
State: A state s_k describes the current environment at the k-th step of the episode. Given a video sequence, we consider a single frame as a state, defined in terms of the extracted feature vector of the current frame.
Action: An action a_k is performed at step k by the system and leads to an update of the state. We define a discrete set of possible actions $A = \{a^1, a^2, ..., a^M\}$, which represents the possible numbers of frames to skip.
Reward: An immediate reward $r_k = r(s_k, a_k, s_{k+1})$ is received by the system when it transits from one state s_k to another state s_{k+1} after taking action a_k (Sec. 3.3).

The accumulated reward is then defined as

$$R = \sum_k \gamma^{k-1} r_k = \sum_k \gamma^{k-1} r(s_k, a_k, s_{k+1}) \quad (1)$$

where $\gamma \in [0, 1]$ denotes the discount factor for the rewards in the future.
Policy: The policy π determines the action to be chosen in every state visited by the system, i.e., it selects the action that maximizes the expected accumulated reward for current and future actions as

$$\pi(s_k) = arg \max_a E[R|s_k, a, \pi] \quad (2)$$

In this case, the policy in FFNet decides how many frames to skip when the system is at certain frame (state).

3.3. Design of the Immediate Reward

In this part, we introduce the definition of the immediate reward r_k for a_k in state s_k. For a raw video available in the training set, we assume each frame i has a binary label $l(i)$. $l(i) = 1$ indicates that frame i is an important frame, and $l(i) = 0$ means it is an unimportant one.

Given a video and its labels, we define the immediate reward as follows:

$$r_k = -SP_k + HR_k \quad (3)$$

The immediate reward consists of two parts that model the "skip" penalty (SP) and the "hit" reward (HR), as explained below.

First, SP_k in Eqn.(3) defines the penalty for skipping the interval t_k in step k:

$$SP_k = \frac{\sum_{i \in t_k} \mathbf{1}(l(i) = 1)}{T} - \beta \frac{\sum_{i \in t_k} \mathbf{1}(l(i) = 0)}{T} \quad (4)$$

where $\mathbf{1}(\cdot)$ is an indicator function that equals to 1 if the condition holds. T is the largest number of frames we may skip, taken as a normalized term. $\beta \in [0, 1]$ is a trade-off factor between the penalty for skipping important frames and the reward for skipping unimportant frames.

Then, the second term HR_k in Eqn.(3) defines the reward for jumping to an important frame or a position near an important frame. To model this reward, we first transfer the one-frame label to a Gaussian distribution in a time window. More specifically, a frame i will have a reward effect on the positions in its nearby window that is defined as

$$f_i(t) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(t-i)^2}{2\sigma^2}), t \in [i - w, i + w] \quad (5)$$

where w controls the window size of the Gaussian distribution. In the experiment section, we set $\sigma = 1, w = 4$. The reason for this transfer is that the reward should be given if the agent jumps to a position that is close to the important frame. To some extent, it jumps to a potentially important area. Assume in time step k, the agent jumps to the z_{th} frame in the original video. Based on the above definition, the HR_k is computed as

$$HR_k = \sum_{i=z-w}^{z+w} \mathbf{1}(l(i) = 1) \cdot f_i(z) \quad (6)$$

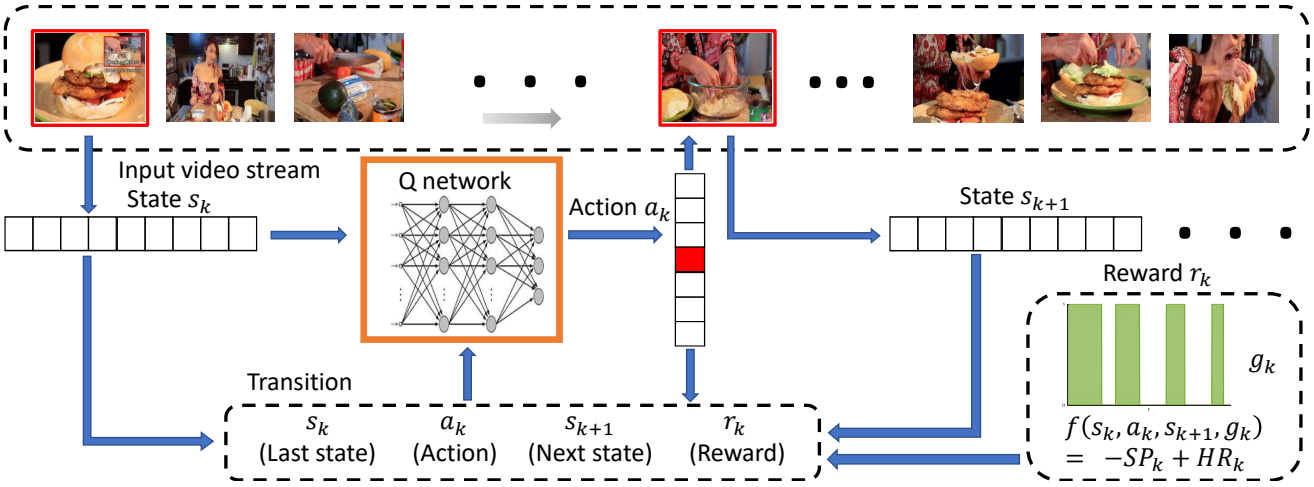

Figure 2. **Model of our FFNet.** We learn a strategy for fast-forwarding videos. At each time step k we use the Q network to select the action a_k, i.e., the number of frames to skip next. The state s_{k+1} is updated with the frame it jumps to. Then, the reward r_k for action a_k in state s_k is computed with the interval annotation g_k. A transition in a quadruple form (s_k a_k s_{k+1} r_k) is used to update the Q network.

3.4. Learning the Fast-Forwarding Policy

During the operation of FFNet, we want to maximize the accumulated reward R in Eqn.(1). Our goal is to find an optimal policy π^* that maps the state to the corresponding action to fulfill the requirement. With Q-learning, we evaluate the value of action $E[R|s, a, \pi]$ as $Q(s, a)$. In classical Q-learning method, the Q-value is updated by

$$\begin{aligned} Q_{k+1}(s_k, a_k) =& (1-\alpha)Q_k(s_k, a_k) \\ &+ \alpha(r_k + \gamma \max_{a_{k+1}} Q_k(s_{k+1}, a_{k+1})) \end{aligned} \tag{7}$$

where $\alpha \in (0, 1]$ represents the learning rate during the training process.

In this problem, we have finite actions but infinite states. No direct assignment of Q-values can be made, thus we use the neural network to approximate the Q-value. The Q-function in this work is modeled by a similar multilayer perception (MLP) structure as in [46]. The input is the current state vector, and the output is a vector of the estimated Q-value for each action given the state. The optimal value of the accumulated reward in time step k is achieved by taking action a_k and represented as $Q^*(s_k, a_k)$, which can be calculated by Bellman equation in a recursive fashion:

$$Q^*(s_k, a_k) = r_k + \gamma \max_{a_{k+1}} Q^*(s_{k+1}, a_{k+1}) \tag{8}$$

where γ is the same discount factor in the definition of the accumulated reward in Eqn. (1). Note that when using gradient descent, Eqn.(8) is consistent with the Q-learning update equation Eqn. (7).

With the above update equation, we use the mean squared error between the target Q-value and the output of MLP as the loss function. During the training process, we apply ϵ-greedy strategy to better explore the state space, which picks a random action with probability ϵ and the action that has $Q^*(s, a)$ with probability 1-ϵ.

The model of our FFNet is shown in Fig. 2. Given a video, the fast-forward agent starts from the first frame. The FFNet Q is initialized with random parameters. For the current frame in time step k, we first extract the feature vector to get the state s_k. Based on the current Q network and the state s_k, one action a_k is chosen using the ϵ-greedy strategy and the agent jumps to a new frame based on the action. Then the current state transits to s_{k+1}, i.e., the feature extracted from the new current frame. With the interval labels g_k of the video, we compute the immediate reward r_k for performing this action. The transition (s_k, a_k, s_{k+1}, r_k) then is sent to update the Q network. More details about our training algorithm are presented in Algorithm 1.

4. Experiments

In this section, we present extensive experiments and comparisons to demonstrate the effectiveness and efficiency of our proposed framework for fast-forwarding videos.

4.1. Experimental Setup

Datasets. We conduct experiments on two publicly available summarization datasets, namely Tour20 [27] and TVSum [41]. Both datasets are very diverse. Tour20 consists of 140 videos of about 20 tourist attractions selected from the Tripadvisor travelers choice landmarks 2015 list. TVSUM contains 50 videos downloaded from YouTube in 10 categories, as defined in the TRECVid Multimedia Event Detection task. To the best of our knowledge, Tour20 is the largest publicly available summarization dataset with 140

Algorithm 1 Training Algorithm for FFNet

Input: a set of videos $\{V\}$ and annotations $\{G\}$
Output: Q-value neural network Q
Init_MLP( ) $\rightarrow Q$
Initialize: memory M = [empty], $explore\_rate$ ϵ = 1
for $i = 1$ to N **do**
  Training_Video_Selection (V, G) $\rightarrow v_i, g_i$
  $frame_{curr} = 0$
  Process($frame_{curr}$) $\rightarrow s_{curr}$
  while $frame_{curr}$ <Size(v_i) **do**
    $$a_{curr} = \begin{cases} a^k \in A, k = random(n), & prob. = \epsilon \\ arg\max Q(s_{curr}, a'), & o.w. \end{cases}$$
    $frame_{next}$ = Action(a_{curr}, $frame_{curr}$)
    Process($frame_{next}$) $\rightarrow s_{next}$
    r = Reward(s_{curr}, a_{curr}, s_{next}, g_i)
    $input = s_{curr}$
    $$target = \begin{cases} r + \gamma \max_{a'} Q(s_{next}, a'), & a = a_{curr} \\ Q(s_{curr}, a), & o.w. \end{cases}$$
    $(input, target) \rightarrow M$
    $s_{next} \rightarrow s_{curr}$
    if $M > batchsize$ **then**
      Training(M,Q) $\rightarrow Q$
      $\epsilon = \max(\epsilon - \triangle \epsilon, \epsilon_{min})$
      Empty(M)
    end if
  end while
end for

videos totaling about 7 hours. Both datasets provide multiple user-annotated summaries for each video. For Tour20 dataset, we combine all three user summaries as human-created summary (labels for training). For TVSum dataset, we first average the frame-level importance scores to compute shot-level scores, and then select top 20% shots for each video as human-created summary.

Implementation Details. Our FFNet is implemented using TensorFlow library on a Tesla K80 GPU. We use a 4-layer neural network to approximate the Q function. The discount factor γ for the rewards in the future is set as 0.8. The exploration rate for Q-learning decays from 1 and stops at 0.1, with a rate of 0.00001. The memory size is set as 128 transitions. We train the Q network up to 1000 epochs for Tour20 and 800 epochs for TVSum.

Performance Measures. Similar to [9], we report a coverage metric at video segment level, which measures how well the results of fast-forward methods cover the important frames in the ground truth obtained from human labeling. More specifically, a segment selected by a method is considered as true positive if the number of important frames it covers (based on the ground truth labels) exceeds a certain threshold called the *hit number*. We evaluate on different hit numbers ranging from 1 to 20 throughout our experiments.

It is important to note that for the intelligent applications we target (e.g., smart surveillance, search and rescue), when measuring system performance, covering the important frames is more critical than skipping the unimportant ones, since such coverage determines the system's capability to identify important events and possibly react to them.

Compared Methods. We compare our approach with several methods that fall into two categories: (1) offline processing methods including Microsoft Hyperlapse (MH) [13], Spectrual Clustering (SC) [44] and Sparse Modeling Representative Selection (SMRS) [5]; and (2) online methods including LiveLight (LL) [51] and Online Kmeans (OK) [2]. Please see supplementary for more details.

Experimental Settings. We use Alexnet [16] fc7 features (4096-dimensional) to represent each video frame and tune the parameters in each method to have the best performance. For each method (including ours), we generate a subset of video frames that has the same length as in ground truth to make a fair comparison. We use the desktop version of Microsoft Hyperlapse (MH) to generate the subset videos in a 4x speed-up rate. For online k-means (OK) and spectral clustering (SC), we set the number of clusters to 20, as in [28]. In LiveLight (LL), the dictionary is initialized as the first 10% of segments in a video. For FFNet, we use an action space of 25, i.e., skipping from 1 to 25 frames and the trade-off factor β is set to 0.8 throughout the experiments. For each dataset, we randomly select 80% of videos for training and use the remaining 20% for testing. We run 5 rounds of experiments and report the average performance.

4.2. Coverage Evaluation

Fig. 3 and Fig. 4 show the mean segment-level coverage achieved by different methods on Tour20 and TVSum dataset, respectively. Each point in these figures represents the segment-level coverage achieved by an algorithm given certain hit number. For example, in Fig. 3, our proposed FFNet achieves a segment-level coverage of about 90% for a hit number of 10. This means that for about 90% of the segments selected in ground truth (i.e., the important segments), at least 10 frames in each of them are selected/covered by FFNet. When the hit number is smaller or equal to 7, the coverage of FFNet is 100%, i.e., every important segment has at least 7 frames selected by FFNet.

When comparing FFNet with other methods in Fig. 3, we have the following observations:

- For smaller hit numbers (say, under 10), our approach achieves excellent coverage (90% or above) and significantly outperforms all other methods (about 10%-20% better). This shows that **the subset selected by FFNet is able to provide more complete coverage of the important information throughout the video stream**.

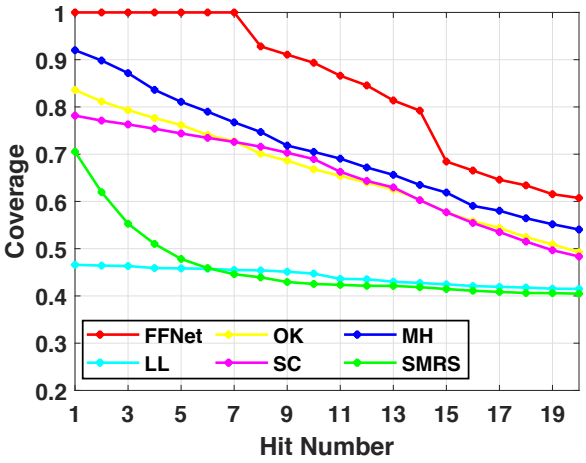

Figure 3. **Segment-level coverage on Tour20 dataset with different hit number thresholds.** Our FFNet (red line on top) outperforms all other methods by a significant margin.

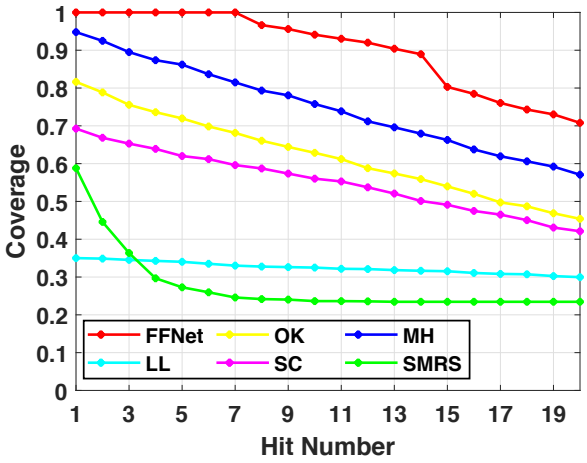

Figure 4. **Segment-level coverage on TVSum dataset under different hit number thresholds.** Our FFNet (red line on top) outperforms all other methods by a significant margin.

- As expected, the coverage of any method goes down with the increase of hit number requirement. Nevertheless, for larger hit numbers (say, 10-20), our approach FFNet still outperforms all other methods. This shows **its consistency in providing better coverage performance**.

Similar result can be seen in Fig. 4 for the TVSum dataset. Notice that for all methods (including ours), performance on Tour20 is not as good as on TVSum. We believe the difference is due to the fact that Tour20 dataset contains some videos capturing static objects and taken from a fixed camera. In this case, the state at each time step in our MDP is the same, which may confuse the Q-learning agent.

Comparison with State-of-the-Art Summarization Methods: We additionally compare our FFNet with the state-of-the-art video summarization methods[9, 28, 50] and one supervised learning baseline (Sup) (implemented as regression) without reinforcement learning. Limited to space, we only present coverage at hit number of 10 in Table 1. Note that we are only able to compare with [50] on the TVSum dataset as the pre-trained model is publicly available by the authors. Our approach outperforms all the baselines by a significant margin, showing that the summary selected by FFNet is able to provide more complete coverage of the important information throughout the video stream. Performance improvement over the Sup baseline shows the advantage of longer time horizon of the reinforcement learning policy in fast-forwarding videos.

Qualitaive Results. Fig. 5 demonstrates a qualitative example from Tour20 dataset (see supplementary file for more of such examples). It clearly demonstrates that our FFNet is able to fast-forward through the unimportant parts and find the most important/relevant parts from a video, and is close to the ground truth (human-created summaries). At the top of Fig. 5 are the representative video segments selected by

Dataset	[9]	[28]	[50]	Sup	FFNet
Tour20	0.754	0.826	-	0.685	0.893
TVSum	0.738	0.877	0.553	0.526	0.941

Table 1. **Coverage achieved by different methods at hit number 10.** "Sup" represents the supervised learning baseline with supervision being the # of frames to jump to the next informative frame as per groundtruth. Our approach performs the best.

our approach. The second row is the ground truth (GT). The remaining rows represent the segments selected by the other methods for the same video. At the beginning, our policy takes larger steps to skip frames that show only clouds without any interesting events. Once the roadside scenes (e.g., shopping area, walking tourists) start, the model begins to take small steps and presents most of the original segments. To summarize, we observe the following.

- For most of the important parts, our FFNet chooses not to skip and presents most of the original segments.
- For unimportant parts, FFNet takes larger jumping steps and smoothly skips frames.

There are also some limitations of our model. Fig. 6 shows a failure case of FFNet. This video records a water fountain scene near Burj Khalifa, captured by a nearly static camera. From beginning to end, the frames are almost the same, except for the change of the water fountain shape and some moving pedestrians. Our FFNet is able to stress on several segments, but they do not match well with the ground truth. We believe this is due to the fact that the Q agent gets similar state after each transition, which make it confused about the pattern of fast-forwarding policy for this particular video. We expect our approach could be made more robust to handle such videos by explicitly using semantic analysis [22] and could also benefit from domain

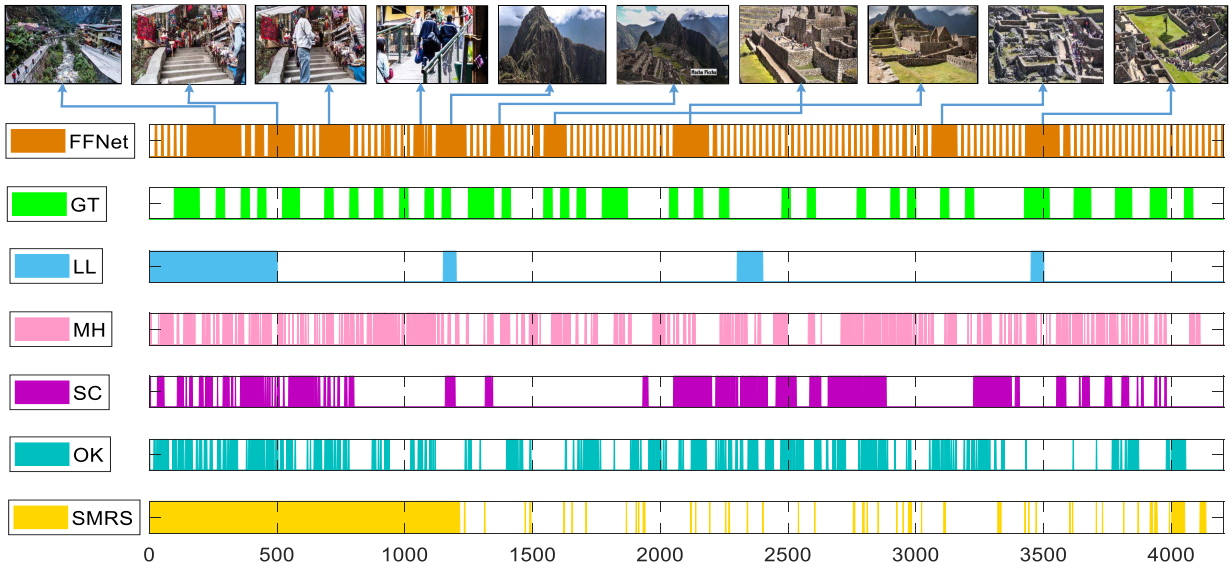

Figure 5. **Exemplar summaries generated while fastforwarding a video of Machu Picchu from the Tour20 dataset.** The frames on top represent segments in our FFNet fast-forwarding result. The rows below illustrate the selected portions using different methods. The X-axis is the frame index over time. Notice that the segments selected by FFNet contains most of the important content labeled in the ground truth, including the roadside scene at the starting point, shopping area, walking tourists, different angles of the natural environment near the attraction, and the main citadel with zoom-in and zoom-out views. Figure is best viewed in color.

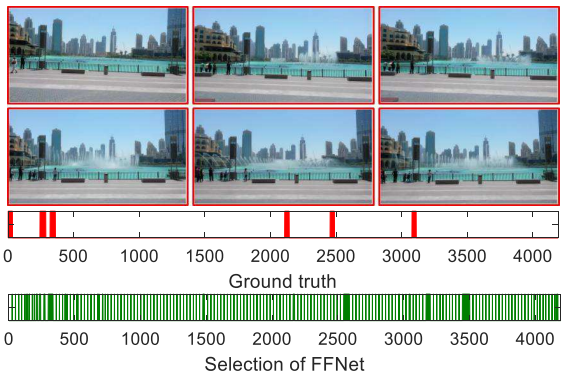

Figure 6. **A failure case of FFNet.** Six frames represent the six important segments in ground truth. The ground truth selection is illustrated in the top row in red, and the selection from FFNet is illustrated below in green (figure is best viewed in color).

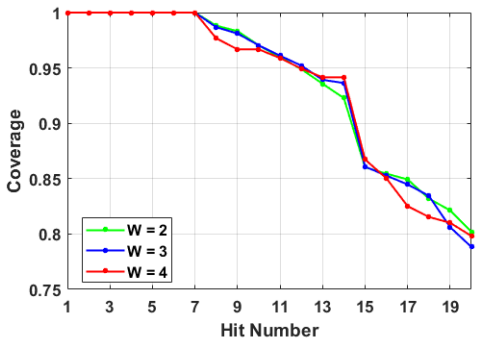

Figure 7. **Effect of window size in reward.** As can be seen, it has little effect on the performance. Best viewed in color.

adaptation techniques [18] for more challenging datasets.

Effect of Window Size: We test our approach on TVSum dataset with 3 cases of window size w in HR_k, set to 2, 3, and 4. Fig. 7 shows that window size has little effect on the performance, indicating that our method is robust to the change in window size.

4.3. User Study

In addition to the above quantitative analysis, we performed a subjective evaluation study involving four human subjects to assess the quality of the selected video frames from different methods.

We choose a random subset of videos from each dataset, and run every method on them. All participants are asked to rate the overall quality of each selected subset of video frames by assigning a rating from 1 to 10, where 1 corresponding to "The selected frames are not at all informative in covering the important content from the original video" and 10 corresponding to "The selected frames are extremely informative in covering the important content from the original video". For each video, the human rating is computed as the averaged rating from all participants (see supplemen-

Dataset	OK	SC	MH	LL	SMRS	FFNet
Tour20	7.96	8.18	8.49	5.28	4.18	**8.70**
TVSum	7.30	7.01	8.10	4.56	3.10	**8.95**

Table 2. **Human ratings for selected video frames from different methods.** The rating for each method is generated by averaging the ratings from all participants. Higher scores indicate better coverage of the important content. Our FFNet achieves the highest rating on both Tour20 and TVSum datasets.

tary for more details). Table 2 shows the average ratings for both Tour20 and TVSum datasets. For both datasets, consistent with the quantitative analysis results, our FFNet outperforms all other methods in covering the important content.

4.4. Processing Efficiency

All prior methods (OK, MH, LL, SC, SMRS) require processing the entire video (100%). In contrast, our FFNet does not process the frames it skips over. In average, it only processes 18.67% of the video frames, which could greatly improve computation efficiency, reduce resource requirement, and lower energy consumption. Note that the requirements on storage and communication are also reduced, but not as much. This is because the neighboring windows of the processed frames are also considered as important for users, and should be stored and transmitted (if needed).

In Fig. 8, we take the video MC10 in Tour20 dataset as an example to illustrate the processing percentage over time for different methods. Microsoft Hyperlapse (MH), Spectral clustering (SC) and SMRS are offline methods that process the entire video. Online Kmeans (OK) takes frames up to the current time, and its processing percentage is linear with respect to the frame number. LiveLight (LL) updates every 50 frames, therefore the processing percentage has a stepwise shape over time. Our FFNet processes the frames at a dynamic speed based on the video content, and eventually only needs to process about 26% of the total video frames.

On a Tesla K80 GPU, the average processing time per frame of FFNet is $8.9357 * 10^{-3}s$, which indicates an average frame rate of 112 fps. Most of the processing time devotes to feature extraction. The fast-forward process only takes 11.28% of the time. On less-capable embedded processors, we may not achieve such high frame rate, but the low processing percentage should still help improve computation efficiency and achieving near real-time speed.

We also analyze the average running time of different methods and observe that our approach is significantly faster than the compared baselines. For a example, on a random subset videos from TVSum dataset, our proposed FFNet takes only 0.71s on to achieve 97% coverage (at hit number 10) while the second fastest baseline SC takes 3.99s to achieve 58% coverage and the third fastest baseline OK takes 11.58s with 71.27% coverage.

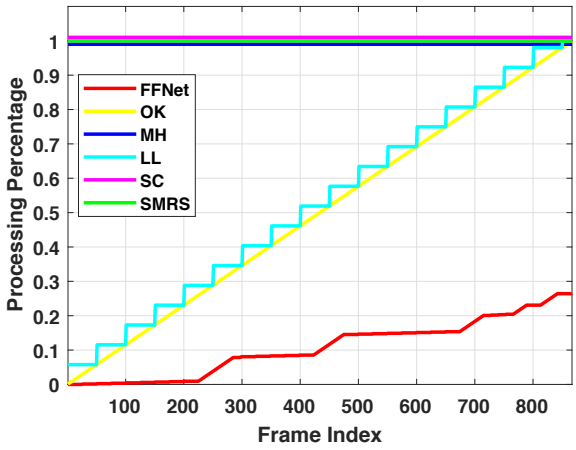

Figure 8. **Example of frame processing percentage over time for different methods.** We take the video MC10 (867 frames) in Tour20 dataset as an example. Offline methods MH, SC and SMRS need to process the entire video before generating the subset. For online methods OK and LL, the processing percentage increases with time and reaches 100% in the end. Our FFNet processes the frames based on the video content and eventually only need to process about 26% of the total frames.

Supplementary Material. Additional results and discussions along with qualitative summaries are included in the supplementary material. We also provide details on the datasets and user study in the supplementary material.

5. Conclusion

In this paper, we present a supervised framework (FFNet) for fast-forwarding videos in an online fashion, by modeling the fast-forwarding operation as an Markov decision process and solving it with a Q-learning method. Quantitative and qualitative results demonstrate that FFNet outperforms multiple baseline methods in both performance and efficiency. It provides an informative subset of video frames that have better coverage of the important content in original video. At the same time, it only processes a small percentage of video frames, which improves computation efficiency and reduces requirements on various resources. In the future, we plan to work on integrating this method with practical system constraints like energy and available bandwidth. It would also be interesting to extend our approach by introducing memory in the form of LSTMs–we leave this as part of the future work.

Acknowledgements: This work was partially supported by NSF grants CNS-1544969, IIS-1724341 and CCF-1553757. The work is primarily carried out while authors Lan and Zhu were at UC Riverside.

References

[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury. A survey on wireless multimedia sensor networks. *Computer networks*, 51(4):921–960, 2007. 1

[2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007. 5

[3] K.-Y. Cheng, S.-J. Luo, B.-Y. Chen, and H.-H. Chu. Smartplayer: user-centric video fast-forwarding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 789–798, 2009. 1, 2

[4] E. Elhamifar and M. C. D. P. Kaluza. Online summarization via submodular and convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[5] E. Elhamifar, G. Sapiro, and R. Vidal. See all by looking at a few: Sparse modeling for finding representative objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2, 5

[6] B. Gong, W. Chao, K. Grauman, and F. Sha. Diverse sequential subset selection for supervised video summarization. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 2

[7] G. Guan, Z. Wang, S. Mei, M. Ott, M. He, and D. D. Feng. A Top-Down Approach for Video Summarization. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 11(1):4, 2014. 2

[8] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool. Creating summaries from user videos. In *European Conference on Computer Vision (ECCV)*, 2014. 2

[9] M. Gygli, H. Grabner, and L. Van Gool. Video summarization by learning submodular mixtures of objectives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2, 5, 6

[10] T. Halperin, Y. Poleg, C. Arora, and S. Peleg. Egosampling: Wide view hyperlapse from egocentric videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017. 1, 2

[11] J. Jiang and X.-P. Zhang. A new player-enabled rapid video navigation method using temporal quantization and repeated weighted boosting search. In *Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Computer Society Conference on*, 2010. 2

[12] J. Jiang and X.-P. Zhang. A smart video player with content-based fast-forward playback. In *Proceedings of the 19th ACM international conference on Multimedia*, 2011. 2

[13] N. Joshi, W. Kienzle, M. Toelle, M. Uyttendaele, and M. F. Cohen. Real-time hyperlapse creation via optimal frame selection. *ACM Transactions on Graphics*, 34(4):63, 2015. 1, 2, 5

[14] A. Khosla, R. Hamid, C.-J. Lin, and N. Sundaresan. Large-scale video summarization using web-image priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[15] G. Kim, L. Sigal, and E. P. Xing. Joint summarization of large-scale collections of web images and videos for storyline reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012. 5

[17] A. Krull, E. Brachmann, S. Nowozin, F. Michel, J. Shotton, and C. Rother. Poseagent: Budget-constrained 6d object pose estimation via reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[18] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 7

[19] T. Ma, M. Hempel, D. Peng, and H. Sharif. A survey of energy-efficient compression and communication techniques for multimedia in resource constrained systems. *IEEE Communications Surveys & Tutorials*, 15(3):963–972, 2013. 1

[20] Y. F. Ma, X. S. Hua, and H. J. Zhang. A generic framework of user attention model and its application in video summarization. *IEEE Transactions on Multimedia*, 2005. 2

[21] S. Mathe, A. Pirinen, and C. Sminchisescu. Reinforcement learning for visual object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[22] T. Mei, L.-X. Tang, J. Tang, and X.-S. Hua. Near-lossless semantic video summarization and its applications to video analysis. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 9(3):16, 2013. 6

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 2

[24] A. G. Money and H. Agius. Video summarisation: A conceptual framework and survey of the state of the art. *Journal of Visual Communication and Image Representation*, 19(2):121–143, 2008. 2

[25] S.-H. Ou, C.-H. Lee, V. S. Somayazulu, Y.-K. Chen, and S.-Y. Chien. Low complexity on-line video summarization with gaussian mixture model based clustering. In *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*, 2014. 2

[26] R. Panda, A. Das, Z. Wu, J. Ernst, and A. K. Roy-Chowdhury. Weakly supervised summarization of web videos. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2

[27] R. Panda, N. C. Mithun, and A. Roy-Chowdhury. Diversity-aware multi-video summarization. *IEEE Transactions on Image Processing*, 26(10):4712–4724, 2017. 2, 4

[28] R. Panda and A. K. Roy-Chowdhury. Collaborative summarization of topic-related videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 5, 6

[29] K. A. Peker, A. Divakaran, et al. An extended framework for adaptive playback-based video summarization. In *Internet Multimedia Management Systems IV*, 2003. 2

[30] K. A. Peker, A. Divakaran, and H. Sun. Constant pace skimming and temporal sub-sampling of video using motion activity. In *IEEE International Conference on Image Processing (ICIP)*, 2001. 2

[31] N. Petrovic, N. Jojic, and T. S. Huang. Adaptive video fast forward. *Multimedia Tools and Applications*, 26(3):327–344, 2005. 1, 2

[32] Y. Poleg, T. Halperin, C. Arora, and S. Peleg. Egosampling: Fast-forward and stereo for egocentric videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2

[33] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid. Category-specific video summarization. In *European Conference on Computer Vision (ECCV)*, 2014. 2

[34] W. L. Ramos, M. M. Silva, M. F. Campos, and E. R. Nascimento. Fast-forward video based on semantic extraction. In *IEEE International Conference on Image Processing (ICIP)*, 2016. 1, 2

[35] Z. Ren, X. Wang, N. Zhang, X. Lv, and L.-J. Li. Deep reinforcement learning-based image captioning with embedding reward. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[36] M. Riedmiller. Neural fitted q iteration-first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning (ECML)*, 2005. 3

[37] G. A. Sigurdsson, X. Chen, and A. Gupta. Learning visual storylines with skipping recurrent neural networks. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[38] M. M. Silva, W. L. S. Ramos, J. P. K. Ferreira, M. F. M. Campos, and E. R. Nascimento. Towards semantic fast-forward and stabilized egocentric videos. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 2

[39] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 2

[40] D. Singh, G. Tripathi, and A. J. Jara. A survey of internet-of-things: Future vision, architecture, challenges and services. In *Internet of things (WF-IoT), 2014 IEEE world forum on*, 2014. 1

[41] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes. Tvsum: Summarizing web videos using titles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 4

[42] Y.-C. Su and K. Grauman. Leaving some stones unturned: dynamic feature prioritization for activity detection in streaming video. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[43] B. T. Truong and S. Venkatesh. Video abstraction: A systematic review and classification. *ACM transactions on multimedia computing, communications, and applications*, 3(1):3, 2007. 2

[44] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007. 5

[45] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992. 3

[46] T. Wei, Y. Wang, and Q. Zhu. Deep reinforcement learning for building hvac control. In *Proceedings of the 54th Annual Design Automation Conference*, 2017. 3, 4

[47] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[48] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Young Choi. Action-decision networks for visual tracking with deep reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[49] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. Summary transfer: Exemplar-based subset selection for video summarizatio. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2

[50] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. Video summarization with long short-term memory. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 2, 6

[51] B. Zhao and E. P. Xing. Quasi real-time summarization for consumer videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 1, 2, 5