Straight to the Point: Fast-forwarding Videos via Reinforcement Learning Using Textual Data

Washington Ramos¹  Michel Silva¹  Edson Araujo¹  Leandro Soriano Marcolino²
Erickson Nascimento¹
¹Universidade Federal de Minas Gerais (UFMG), Brazil  ²Lancaster University, UK
¹{washington.ramos, michelms, edsonroteia, erickson}@dcc.ufmg.br, ²l.marcolino@lancaster.ac.uk

Abstract

The rapid increase in the amount of published visual data and the limited time of users bring the demand for processing untrimmed videos to produce shorter versions that convey the same information. Despite the remarkable progress that has been made by summarization methods, most of them can only select a few frames or skims, which creates visual gaps and breaks the video context. In this paper, we present a novel methodology based on a reinforcement learning formulation to accelerate instructional videos. Our approach can adaptively select frames that are not relevant to convey the information without creating gaps in the final video. Our agent is textually and visually oriented to select which frames to remove to shrink the input video. Additionally, we propose a novel network, called Visually-guided Document Attention Network (VDAN), able to generate a highly discriminative embedding space to represent both textual and visual data. Our experiments show that our method achieves the best performance in terms of F1 Score and coverage at the video segment level.

1. Introduction

From the dawn of the digital revolution until this very day, we are witnessing an exponential growth of data, in particular, textual and visual data such as images and videos. New technologies like social media and smartphones massively changed how we exchange and acquire information. For instance, there is a plethora of textual tutorials and instructional videos on the Internet teaching a variety of tasks, from how to cook burritos and tacos, all the way to how to solve partial differential equations (PDEs), and to device operations manuals.

Despite many textual tutorials and instructional videos sharing the increasing growth of available data as well as the content, they differ in a key aspect for users: how long they would take to consume the content. In general, information encoded by producers is more concise in textual data than when they use visual data. For instance, a recipe of tacos or a tutorial explaining how to solve a PDE is described in a few sentences. Instructional videos, for their turn, might have several minutes showing non-relevant information for the task such as a person opening a refrigerator, picking up the pencil, or erasing the blackboard. Such segments could be fast-forwarded without losing the crucial information encoded in the input video to understand the task. Thus, ideally, instructional videos should be concise, similar to a textual description, but still having visually-rich demonstrations of all main steps of the task.

In this paper, we address the problem of accelerating untrimmed videos by using text documents (see Figure 1). For example, a recipe for cooking tacos could be used...
as a guide to select relevant frames from cooking tacos videos. Note that this problem is different from video segmentation [41] or summarization [12, 6, 15, 37] since non-relevant frames are still necessary for a user to understand the flow and temporal coherence of a task, i.e., some segments should be accelerated, but not eliminated.

Our method follows the encoding-decoding framework to create fast-forward videos guided by a set of sentences (i.e., a document). We formulate our fast-forwarding task as a sequential decision-making process, where a reinforcement learning agent observes the encoded text and video frames and decides to increase, decrease, or keep the speed-up rate of the video. The embedding space for user documents and video frames is generated by a novel Visually-guided Document Attention Network (VDAN), which creates representative feature vectors for textual and visual modalities. In this embedding space, the vectors will be close when representing the same semantic concept and distant otherwise. Figure 1 shows a schematic representation of the main steps in our methodology.

Despite remarkable advances in summarization methods [12, 6, 15, 37], most existing approaches do not take into account the temporal continuity, i.e., summarization techniques segment the input videos into several clips, which creates visual gaps between consecutive video segments, and does not preserve the video context. Most recently, the algorithms on fast-forwarding videos have emerged as effective approaches to deal with the tasks of retrieving meaningful segments without losing the temporal continuity [11, 17, 23, 26]. On the flip side, fast forward approaches are limited by the lack of a well-defined semantic definition.

This paper takes a step forward towards fast-forwarding videos based on the semantics of the content. By using textual data to guide an agent that seeks the best set of frames to be removed, our method emphasizes highly semantic content segments, while preserving the temporal continuity. We evaluate our approach on the challenging YouCook2 dataset [41]. The experiment has shown that our method achieves the best performance in terms of F1 Score and coverage at the video segment level.

Contributions. The contributions of this work can be summarized as follows: i) a new fast-forward method based on a reinforcement learning formulation, which is able to accelerate videos according to frame similarity scores with textual data; ii) a novel Visually-guided Document Attention Network (VDAN) capable of generating a highly discriminative embedding space for textual and visual data.

2. Related Work

Various methods have been proposed in the literature to deal with the task of shortening a video using different approaches such as summarization [12, 6, 15, 37], fast-forward [22, 9, 23, 25, 11], cross-modal techniques [21], and reinforcement learning [11]. In the following, we present the works most related to ours, as well as the most representative techniques of each approach.

Video summarization. Over the past several years, video summarization methods were the big players when dealing with the task of shortening a video [12, 6, 15, 37]. The shorter version is, usually, a summary of the input video composed of a storyboard of keyframes or video skims with the most distinguishable segments [4]. Most of the summarization methods select the frames or skims using a relaxed or non-existent temporal restriction, resulting in visual gaps and breaking the video context.

The strategy adopted by researchers to create the summary varies from clustering visual features of frames [15] and training neural networks that infer the representativeness of a video segment [35, 37], to employing additional information such as user queries, external sensors [12], or textual annotations [21]. Lee et al. [12], in the context of first-person videos, analyzed properties such as social interaction, gaze, and object detection to create a storyboard summary of the input video. Zhang et al. [37] proposed a method to create either a storyboard or skims by modeling long-range dependencies among the video frames using a bi-directional Long Short-Term Memory (LSTM) recurrent network. Yao et al. [35] performed the selection of the relevant segments fusing information from spatial and temporal Deep Convolutional Neural Networks (DCNNs) to identify highlighting moments in sports videos.

Reinforcement learning has also been applied to video summarization [40, 11], motivated by the fact that it had been successfully applied to many challenging tasks, such as mastering complex games like Go [29], Shogi [28], and achieving super-human performance in Atari games [31]. Additionally, it has great application in vision tasks, including visual tracking [36], and active object recognition [18]. Zhou et al. [40] presented an end-to-end unsupervised framework also based on the reinforcement learning paradigm. Their method summarizes videos by applying a diversity-representativeness reward that guides the agent to create more diverse and more representative summaries.

Semantic fast-forward. The lack of context that emerges from the gaps generated by video summarization methods creates a nuisance to the consumers of instructional videos. The existence of a gap also might confuse the user about the whole process. In other words, the user would be unaware if an important step was missed with the gap if the original video is unknown. Fast-forward based methods add time constraint in the frame sampling, which results in a shorter and contiguous version of the input video.
Some approaches also deal with visual stability constraints when sampling the frames, achieving smooth final videos by modeling the sampling step as an optimization problem [22, 9, 10, 23]. A drawback of applying previous fast-forward methods in instructional videos is that the whole video would be sped-up. A recipe video, for example, might have a long and straightforward step, like boiling vegetables. This step would be sped-up with the same rate as a shorter and complicated task like filleting a fish.

Semantic fast-forward methods, on the other hand, emphasize segments with a high semantic load. The emphasis effect is achieved by accelerating the relevant segments with a lower speed-up rate when compared with the rate applied to the rest of the video. Okamoto and Yanai [17] proposed a method to fast-forward guidance videos emphasizing video segments containing pedestrian crosswalks or turning movements in street corners. Ramos et al. [23] presented a semantic fast-forward method for first-person videos dealing with visual stability constraints with emphasis on faces/pedestrians. Silva et al. [26] extended the work of Ramos et al., including an automatic parameter setting based on Particle Swarm Optimization (PSO) algorithm, a Convolutional Neural Network to assign frame scores based on Internet users’ preferences, and a video stabilizer proper to fast-forwarded videos [27]. Silva et al. [25] proposed a semantic fast-forward to first-person videos by modeling the frame sampling as a Minimum Sparse Reconstruction problem. A drawback of this work is that it requires a preprocessing step that is time-consuming and relies on the accuracy of other methods.

In a recent work, Lan et al. [11] introduced the Fast Forward Net (FFNet). Their methodology summarizes videos on the fly using a reinforcement learning based method to select frames with the most memorable views according to human-labeled data. Similar to FFNet and Zhou et al., we also apply an agent that is trained by the reinforcement learning paradigm; however, our approach is a step towards training agents to work in a cross-modal embedding space.

**Cross-modal embedding.** Recently, the algorithms on cross-modal embedding have emerged as promising and effective approaches to deal with a variety of tasks such as video description [19] and text-based image or video retrieval [16, 5, 1, 19], to name a few. Virtually all these methods rely on creating a shared embedding space, where features from multiple modalities can be compared.

A successful application of a cross-modal approach has been presented by Plummer et al. [21]. The authors created a video summary approach that selects the best subset of video segments by analyzing their visual features (e.g., representativeness, uniformity, and interestingness) along with vision-language modeling. Salvador et al. [24] applied a multi-modal neural model to learn a common embedding space for images and recipes and tackled the task of retrieving recipes from image queries. Carvalho et al. [2] extended the method of Salvador et al. to use a different loss function. Wang et al. [32] proposed an adversarial learning strategy to align both modalities.

Most of these works perform document retrieval using as query an image representing the final result of a recipe. Our proposed cross-modal embedding (VDAN), on the other hand, provides the semantic distance between each frame in the instructional video and the textual steps described in the document, i.e., the recipe.

**3. Methodology**

Our method is based on an encoding-decoding framework to create fast-forwarded videos. The first stage of our method consists of the novel Visually-guided Document Attention Network (VDAN). The VDN creates an embedding space for encoding documents and images. In the second stage, we formulate the fast-forwarding task as a sequential decision-making process. We train a reinforcement learning agent that observes the encoded text and video frame and decodes them into a distribution over the actions for increasing, decreasing, or maintaining the current speed-up rate of the output video. Figure 2 illustrates the main steps of our approach.

**3.1. Visually-guided Document Attention Network**

Since our ultimate goal is to create a fast-forward video by dropping non-relevant frames given an input document, we propose the Visually-guided Document Attention Network (VDAN). Our network takes a document and an image as input and, guided by the visual features, creates representative feature vectors for both modalities. By training VDN, we aim at creating an embedding space in which textual and visual features are aligned. We argue the aligned embedding vectors help our agent make sense of the semantic proximity between a frame and the document, and then learn the best policy to discard non-relevant frames as far as the document is concerned (e.g., a recipe).

Formally, let \( D = \{p_1, p_2, \ldots, p_N\} \) be the document composed of \( N \) sentences, and \( I \) be the image that feeds the network. In our task, \( D \) is represented by a document composed of a set of textual instructions, and \( I \) is a video frame. VDN produces \( d \)-dimensional embeddings \( e_D \in \mathbb{R}^d \) and \( e_I \in \mathbb{R}^d \) for textual and visual data, respectively, given the parameters \( \theta_E = \{\theta_D, \theta_I\} \).

**Document Encoder.** To encode \( D \), we employ a Hierarchical Recurrent Neural Network (H-RNN) coupled with a soft attention mechanism in each level [34, 39], since an H-RNN can capture long-range temporal dependencies [38].

Our H-RNN is composed of two levels of encoding: i) the sentence-level and ii) the document-level, as illustrated
in Figure 2. Each level contains bi-directional GRU [3] units that produce hidden state vectors. These vectors feed the attention layer. Let \( w_{i1}, w_{i2}, \ldots, w_{iM_i} \) denote the distributional word representation [20] of each word in sentence \( p_i \). The sentence-level encoder produces a hidden state vector \( h_{ij} = f_p(w_{ij}; h_{i(j-1)}, \theta_{R_p}) \) at each timestep \( j \) given the word embedding \( w_{ij} \), the previous hidden state \( h_{i(j-1)} \), and the parameters \( \theta_{R_p} \). As stated by Yang [34], words present different contributions to the meaning of a sentence. Therefore, we feed \( h_{ij} \) to the attention module defined as:

\[
\begin{align*}
    u_{ij} &= \tanh(W_p h_{ij}), \\
    \alpha_{ij} &= \frac{\exp(u_{ij}^T c_p)}{\sum_j \exp(u_{ij}^T c_p)}, \\
    p_i &= \sum_j \alpha_{ij} h_{ij},
\end{align*}
\]

where \( u_{ij} \) is a hidden representation of \( h_{ij} \), \( \alpha_{ij} \) gives the importance weights for each \( h_{ij} \), \( p_i \) is the sentence-level embedding for the sentence \( p_i \), \( c_p \) is a word-level context vector which acts as a fixed query to find the informative word, and \( W_p \) is a projection matrix. The alignment between \( c_p \) and \( u_{ij} \) defines the score used to compute \( \alpha_{ij} \).

In the document-level encoding, each \( p_i \) is used to produce a hidden state vector \( h_i = f_d(p_i; h_{i-1}, \theta_{R_d}) \). Different sentences may also present different contributions to the document. In our approach, the instructional characteristic of the document increases the probability of a given video frame being similar to only one instruction. Thus, similar to the sentence-level counterpart, we also employ an attention module, which is parameterized by \( W_d \) and \( \theta_d \). As a result, after feeding the document-level encoder with all vectors \( p_i \), it yields the document-level encoding \( d \). Finally, we project \( d \) into the embedding space using a fully connected network \( f_D \) parameterized by \( \theta_D \). Thus, \( e_d = f_D(d; \theta_D) \).

**Image Encoder.** To produce the image embedding \( e_I \), we first extract the image features with a ResNet-50 [8] encoder, producing an intermediate vector \( \phi(I) \in \mathbb{R}^5 \). Then, we project \( \phi(I) \) into the embedding space using a fully connected network \( f_I \) parameterized by \( \theta_I \) as follows \( e_I = f_I(\phi(I); \theta_I) \). To guide the document-level attention module to attend the correct sentence, we set the first hidden state vector of the document-level encoder as \( h_0 = \phi(I) \). For the sentence-level, however, we set \( h_0 = \mathbf{0} \).

Both document and image encoders also include an \( \ell_2 \) normalization layer to make \( e_D \) and \( e_I \) unit norm vectors. The attention module learns to attend the correct words and sentences to produce an embedding \( e_D \) more aligned to \( e_I \).

**Training.** For each image \( I \) in the training set, we create a positive and a negative document, \( D^+ \) and \( D^- \), to compose the training pairs \( <D^+, I> \) and \( <D^-, I> \). A positive document \( D^+ \) consists of the sentences that describe the image \( I \) and, additionally, sentences that describe a randomly selected image \( I' \). The strategy of adding sentences that do not describe the image helps the document-level attention module to attend the proper sentences at training time. To create the negative document, \( D^- \), we randomly select two other images \( I' \) and \( I'' \), and collect their respective sen-
tences. At each training step, we shuffle all the sentences in the document for generalization purposes.

In order to create more aligned embeddings, we optimize \( \theta_{\text{enc}} = \{ \theta_W, \theta_p, \theta_{\text{enc}} \} \) by minimizing the cosine embedding loss as follows:

\[
L_{\text{enc}}(\hat{D}, \hat{I}; \theta_{\text{enc}}) = \begin{cases} 
1 - \cos(e_D, e_I), & \text{if } y = 1 \\
\max(0, \cos(e_D, e_I) - \eta), & \text{otherwise,}
\end{cases}
\]  

where \( \hat{D} \) and \( \hat{I} \) are the training document and image, respectively, \( y \) is equal to 1 if \( \hat{D} \) and \( \hat{I} \) correspond, and \( \eta \) is a margin parameter, which is set to 0 in our problem.

3.2. Semantic Fast-Forward Network (SFF-RL)

In the second stage of our methodology, we define our decoding phase, in which the agent observes the encoded vectors \( e_D \) and \( e_I \) and sample an action over the action space to adjust the speed-up rate accordingly.

We formulate the problem of selecting frames as a Markov Decision Process (MDP). In our formulation, we train an agent to maximize the expected sum of discounted rewards:

\[
R_t = \mathbb{E} \left[ \sum_{n=0}^{T-t-1} \gamma^n r_{t+n} \right],
\]  

where \( t \) is the current timestep, \( r_{t+n} \) is the reward \( n \) timesteps into the future, and \( T \) is the total number of timesteps. At each timestep, one frame is selected; therefore, \( t \) also indicates the current number of the selected frames. \( \gamma \in (0, 1] \) is a discount factor. In our case, however, future rewards are equally important, and hence we use \( \gamma = 1 \).

In this problem, the agent observes a video and a text, and must take actions to create an optimal accelerated version of the input video. Since we want to keep the overall coherence of the video instead of trimming it, we model an agent that navigates in the video space. I.e., the agent has velocity \( v \) and acceleration \( \omega \) and based on the current velocity, the next frame is selected. Therefore, the agent goes through the whole video, but skips frames according to a dynamically changing velocity. At each timestep, the agent can increase, decrease, or keep its current acceleration, which will, in turn, affect the velocity. Since we apply Model-free Reinforcement Learning, the transition function does not need to be pre-defined, nor learned; as the agent focuses directly on learning the best policy. In the following, we define all elements used in our MDP formulation.

State and actions. In order to allow an agent to effectively navigate through the video space, we define the state vector as the concatenation of the document and frame embeddings, i.e., \( s_t = [e_D; e_I] \in S \). To create a fast-forwarded video using a textual input, our agent adaptively adjusts the speed-up rate such that the video segments which semantically match the input text are exhibited in a lower speed and the others in a higher speed. Thus, the agent’s action space \( A \) has three actions: i) decelerate; ii) do nothing; and iii) accelerate. As mentioned, the agent has a current velocity \( v \), and hence would skip the next \( v \) frames for whichever action it takes. Decelerate and accelerate update the velocity and acceleration states of the agent as \( v = v - \omega \) and \( \omega = \omega - 1 \) for decelerate, and \( v = v + \omega \) and \( \omega = \omega + 1 \) for accelerate, while do nothing keeps the current \( v \) and \( \omega \). Additionally, acceleration and velocity saturate at certain values \( \omega_{\text{max}} \) and \( v_{\text{max}} \), respectively, and they are always greater than or equal to 1. Note that \( \omega \) does not correspond to a physical acceleration, allowing the agent to quickly adjust the velocity to collect more rewards when the semantic level changes.

Reward function. The goal of the agent is to learn a policy \( \pi(a|s_t, \theta_{\pi}) \), which represents the probability of the agent taking a certain action \( a \in A \) given the state \( s_t \) and the parameters \( \theta_{\pi} \). The reward should encourage the agent to increase or decrease the speed-up of the video, given the semantic similarity between the visual and textual data in the upcoming frame. Therefore, we design an immediate reward proportional to the text and frame features alignment. Thus, at training time, after taking the action \( a_t \sim \pi(a|s_t, \theta_{\pi}) \) in the \( t \)th step, the agent receives the following reward signal:

\[
r_t = e_D \cdot e_I.
\]  

Note that the agent receives higher rewards if \( e_D \) and \( e_I \) point to the same direction in the embedding space, which encourages it to reduce the speed and accumulate more rewards since the temporal neighboring frames are more likely to yield higher reward values.

Objective. Apart from aligning the textual and visual features produced by VDAN, the overall objective of our framework also tries to maximize the expected cumulative reward \( R_t \) at each timestep \( t \). We follow the REINFORCE algorithm [33] to learn the parameters \( \theta_{\pi} \) in order to maximize the expected utility

\[
J(\theta_{\pi}) = \sum_{a \in A} \pi(a|s_t, \theta_{\pi}) R_t.
\]  

In order to improve the learning performance, we employ the advantage function approach [30], and maximize the expected advantage:

\[
J'(\theta_{\pi}) = \sum_{a \in A} \pi(a|s_t, \theta_{\pi})(R_t - v(s_t|\theta_v)),
\]  

where \( v(s_t|\theta_v) \) is a function parameterized by \( \theta_v \), which predicts our expected cumulative reward at state \( s_t \). The gradient of \( J' \), \( \nabla_{\theta_{\pi}} J'(\theta_{\pi}) \), is given by

\[
\sum_{a \in A} \pi(a|s_t, \theta_{\pi}) \nabla_{\theta_{\pi}} \log \pi(a|s_t, \theta_{\pi})(R_t - v(s_t|\theta_v)).
\]
Usually, Monte Carlo sampling is applied, due to the high dimension of the action sequence space, leading to the following approximation for the gradient:

$$\nabla J'(\theta_\pi) \approx \sum_t \nabla_{\theta_\pi} \log \pi(a_t|s_t, \theta_\pi)(R_t - v(s_t|\theta_v)),$$

(9)

where \(a_t\) is the action taken at time \(t\). Hence, we minimize the following loss function:

$$L'(\theta_\pi) = -\sum_t (\log \pi(a_t|s_t, \theta_\pi))(R_t - v(s_t|\theta_v)).$$

(10)

Also, it is usually recommended to add the entropy of the policy output into the loss, in order to have a greater action diversity [13]. Therefore, our final policy loss is given by

$$L_{dec}(\theta_\pi) = L'(\theta_\pi) - \sum_t \beta \cdot H(\pi(a_t|s_t, \theta_\pi)),$$

(11)

where \(\beta\) is a constant to balance the entropy importance. In our experiments, we set \(\beta = 0.01\).

Additionally, we also need to learn the state-value function \(v(s_t|\theta_v)\). We do that by minimizing the mean squared error:

$$L_v(\theta_v) = \sum_t (v(s_t|\theta_v) - R_t)^2 .$$

(12)

Both losses \(L_v\) and \(L_{dec}\) can now be minimized using stochastic gradient descent.

At test time, we use \(\arg \max_a \pi(a|s_t, \theta_\pi)\) as the chosen action for the agent in a given timestep \(t\).

4. Experiments

In this section, we investigate the performance of our method evaluating it both qualitatively and quantitatively on different recipe videos.

4.1. Experimental Setup

Dataset and Evaluation Metric. We extract a subset of videos from the YouCook2 dataset [41] to compose the sets for training and testing our approach. The videos in the dataset were collected from YouTube and are distributed over 89 recipes such as grilled cheese, hummus, etc. Each video has up to 16 English sentences localized by timestamps, and each of these sentences corresponds to a video segment in which the instruction is being executed. Since we aim at creating shorter videos that convey the same information of the original video, we only evaluated our method in videos whose instruction segments correspond to at most 25% of its length, resulting in a total of 121 videos.

To evaluate the performance of each method, we computed the F1 Score and, following Gygli et al. [7] and Lan et al. [11], we also evaluate our method in terms of coverage at video segment level. While the F1 Score consists of a weighted average of Precision and Recall, the coverage at video segment level gives the quality of the coverage of frames manually annotated as relevant. We consider that an important segment has been taken if the number of relevant frames (according to the ground truth) selected by an approach is higher than a threshold, the hit number.

Baselines. We pit our method against the FFNet [11] and the work of Silva et al. [25] on sparse adaptive sampling (SSFF). While SSFF holds state of the art in semantic fast-forward, FFNet, similar to our approach, is a fast-forward method based on the reinforcement learning paradigm.

Implementation Details. In our experiments, we use the MSCOCO dataset [14] to compose positive and negative pairs to train the VDAN. MSCOCO contains 113,287 training images with 5 captions each, and 5,000 images respectively for validation and testing. For the VDAN, we use the glove embeddings set pre-trained in the Wikipedia 2014 and Gigaword 5 sets provided by Pennington et al. [20]. We set \(d = 128\) as the dimension of the embedding space and the size of the hidden state vectors \(h_i\) and \(h_t\) to 1024 and 2048, respectively. \(f_I\) and \(f_D\) are implemented as two independent fully connected neural networks composed of a single hidden layer with 512 neurons. We train VDAN for 30 epochs with a batch size of 64 and obtain the model that had the best performance on validation. The policy network \(\pi(a_t|s_t, \theta_\pi)\) and the value-state function \(v(s_t|\theta_v)\) are implemented as two independent neural networks with two hidden layers composed of 256 and 128 neurons, respectively. We trained our agent during 100 epochs. Both the VDAN network and the SFF-RL are trained using Adam with a learning rate of 0.00001 for optimization. The value-state approximator, however, is trained with a learning rate of 0.001 for faster convergence. We set \(\omega_{max}\) and \(v_{max}\) as 5 and 20, respectively. We trained the Lan et al.’s agent with the same number of epochs as ours, with exploration decay \(\epsilon = 0.0001\). The other parameters are set, as suggested by the authors. Our approach is fully implemented in the PyTorch library, and the experiments were conducted in a single NVIDIA GeForce GTX 1080Ti GPU.

4.2. Results

Quantitative results. Table 1 shows the results in terms of precision, recall, and F1 Score in our test set. The recipes marked with a * symbol present two videos; therefore, we computed the mean value of such videos to present in the table. Results for Precision, Recall, and F1 Score show that our method achieves a better performance in comparison to state-of-the-art techniques. We merit the generalization capacity of our agent for the higher recall values presented. At test time, our agent successfully considered the frame
Table 1. Precision, Recall and F1 Score results for our test set. The * symbol indicates recipes for which we have collected two recipes and reported their average values. Our method outperforms the baseline competitors in most cases. The best results are in bold.

<table>
<thead>
<tr>
<th>Test Set</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SSFF</td>
<td>FFNet</td>
<td>Ours</td>
</tr>
<tr>
<td>Waldorf salad</td>
<td>0.20</td>
<td>0.19</td>
<td>0.34</td>
</tr>
<tr>
<td>Grilled cheese</td>
<td>0.22</td>
<td>0.23</td>
<td>0.27</td>
</tr>
<tr>
<td>Corn dogs</td>
<td>0.14</td>
<td>0.17</td>
<td>0.18</td>
</tr>
<tr>
<td>Hash browns</td>
<td>0.27</td>
<td>0.22</td>
<td>0.32</td>
</tr>
<tr>
<td>Bangers and mash</td>
<td>0.21</td>
<td>0.29</td>
<td>0.32</td>
</tr>
<tr>
<td>Foie gras*</td>
<td>0.18</td>
<td>0.21</td>
<td>0.17</td>
</tr>
<tr>
<td>Escargot</td>
<td>0.33</td>
<td>0.23</td>
<td>0.30</td>
</tr>
<tr>
<td>Sauerkrut*</td>
<td>0.28</td>
<td>0.22</td>
<td>0.25</td>
</tr>
<tr>
<td>Goulash</td>
<td>0.20</td>
<td>0.30</td>
<td>0.27</td>
</tr>
<tr>
<td>Beef bourguignon</td>
<td>0.25</td>
<td>0.25</td>
<td>0.26</td>
</tr>
<tr>
<td>Wiener schnitzel</td>
<td>0.28</td>
<td>0.23</td>
<td>0.29</td>
</tr>
<tr>
<td>Pasta e fagioli*</td>
<td>0.24</td>
<td>0.43</td>
<td>0.33</td>
</tr>
<tr>
<td>Hummus</td>
<td>0.30</td>
<td>0.25</td>
<td>0.52</td>
</tr>
<tr>
<td>Udon noodle soup</td>
<td>0.22</td>
<td>0.18</td>
<td>0.11</td>
</tr>
<tr>
<td>Indian lamb curry*</td>
<td>0.17</td>
<td>0.23</td>
<td>0.16</td>
</tr>
<tr>
<td>Dal makhani</td>
<td>0.22</td>
<td>0.38</td>
<td>0.20</td>
</tr>
<tr>
<td>Wanton noodle</td>
<td>0.21</td>
<td>0.19</td>
<td>0.20</td>
</tr>
<tr>
<td>Masala dosa</td>
<td>0.11</td>
<td>0.14</td>
<td>0.16</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td><strong>0.22</strong></td>
<td><strong>0.24</strong></td>
<td><strong>0.26</strong></td>
</tr>
<tr>
<td><strong>Std</strong></td>
<td><strong>0.06</strong></td>
<td><strong>0.07</strong></td>
<td><strong>0.09</strong></td>
</tr>
</tbody>
</table>

Figure 3. Comparison of the segment-level coverage on our test set. Our method outperforms the competitors in all hit numbers.

and the document to be related. Then, it reduced the speed and acceleration to capture as most relevant frames as it can while maintaining a high precision. Note that our method also outperforms the competitors in terms of precision in several cases. A notable exception is the video of the “Udon noodle soup” recipe. In this case, our method achieved an F1 Score of 0.07. The reason for such a score is that the vectors produced by VDAN along the segments with the instructions were not well aligned in the embedding space, making the agent to accelerate or even decelerate for a short period of time. This case is illustrated in the Figure 5.

Figure 3 depicts the coverage results at segment level. Each point in the plot represents the coverage at a certain hit number. Our method achieves the best performance considering all hit numbers, covering approximately 20% of the important segments when using the higher threshold.

Qualitative results. We present our qualitative results in Figure 4. The colored bars represent the frames selected by each method, while the contiguous black blocks represent the ground-truth segments. The frames and their associated instructions are depicted above the bars. Note that our method presents a denser frame selection when the recipe instructions are shown in the video, in both cases, which indicates that the agent learned a reasonable policy. I.e., by observing the recipe and the video frame, the agent acts correctly when navigating through the video.

Ablation Study. To verify that the word-level attention contributes to the success of the embedding space created by VDAN, we computed the average of the distribution of cosine distances between corresponding and non-corresponding pairs of images and documents. We observed that when adding the word-level attention, the val-
Due to the slight improvement, we used both the word and sentence-level attention in all experiments.

5. Conclusions

In this paper, we proposed a novel methodology based on a reinforcement learning formulation to accelerate instructional videos, where the agent is trained to decide which frames to remove based on textual data. We also presented a novel network called Visually-guided Document Attention Network (VDAN) that creates a highly discriminative embedding space to represent both textual and visual data. Our approach outperforms both FFNet and SSFF methods in terms of F1 Score and coverage at video segment level.

Acknowledgments. We thank the agencies CAPES, CNPq, FAPEMIG, and Petrobras for funding different parts of this work. We also thank NVIDIA Corporation for the donation of a TITAN Xp GPU.
References


[19] Yingwei Pan, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui. Jointly modeling embedding and translation to bridge video and language. In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pages 4594–4602, 2016. 3


Ting Yao, Tao Mei, and Yong Rui. Highlight detection with pairwise deep ranking for first-person video summarization. In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pages 982–990, June 2016.


