Video Captioning via Hierarchical Reinforcement Learning

Xin Wang, Wenhu Chen, Jiawei Wu, Yuan-Fang Wang, William Yang Wang

University of California, Santa Barbara
{xwang,wenhuchen,jiawei,wu,yfwang,william}@cs.ucsb.edu

Abstract

Video captioning is the task of automatically generating a textual description of the actions in a video. Although previous work (e.g. sequence-to-sequence model) has shown promising results in abstracting a coarse description of a short video, it is still very challenging to caption a video containing multiple fine-grained actions with a detailed description. This paper aims to address the challenge by proposing a novel hierarchical reinforcement learning framework for video captioning, where a high-level Manager module learns to design sub-goals and a low-level Worker module recognizes the primitive actions to fulfill the sub-goal. With this compositional framework to reinforce video captioning at different levels, our approach significantly outperforms all the baseline methods on a newly introduced large-scale dataset for fine-grained video captioning. Furthermore, our non-ensemble model has already achieved the state-of-the-art results on the widely-used MSR-VTT dataset.

1. Introduction

For most people, watching a brief video and describing what happened (in words) is an easy task. For machines, extracting the meaning from video pixels and generating natural-sounding description is a very challenging problem. However, due to its wide range of applications such as intelligent video surveillance and assistance to visually-impaired people, video captioning has drawn increasing attention from the computer vision community recently. Different from image captioning which aims at describing a static scene, video captioning is more challenging in the sense that a series of coherent scenes need to be understood in order to jointly generate multiple description segments (e.g., see Figure 1).

Current video captioning tasks can mainly be divided into two families, single-sentence generation [38, 16] and paragraph generation [24]. Single-sentence generation tends to abstract a whole video to a simple and high-level descriptive sentence, while paragraph generation tends to grasp more detailed actions, and generates multiple sentences of descriptions. However, even for paragraph generation, the paragraph is often split into multiple, single-sentence generation scenarios associated with ground truth temporal video intervals.

In many practical cases, human activities are too complex to be described with short, simple sentences, and the temporal intervals are hard to be predicted ahead of time without a good understanding of the linguistic context. For instance, in the bottom example of Figure 1, there are five human actions in total: sit on a bed, put a laptop into a bag are happening simultaneously, and then followed by stand up, put the bag on one shoulder and walk out of the room in order. Such fine-grained caption requires a subtle and expressive mechanism to capture the temporal dynamics of the video content and associate that with semantic representations in natural language.

In order to tackle this issue, we propose a “divide and
conquer” solution, which first divides a long caption into many small text segments (e.g. different segments are in different colors as shown in Figure 1), and then employs a sequence model to conquer each segment. Instead of forcing the sequence model to generate the whole sequence in one shot, we propose to guide the model to generate sentences segment by segment. With a higher-level sequence model designing the context of each segment, the low-level sequence model follows the guidance to generate the segment word by word.

In this paper, we propose a novel hierarchical reinforcement learning (HRL) framework to realize this two-level mechanism. The textual and video context can be viewed as the reinforcement learning environment. Our framework is a fully-differentiable deep neural network (see Figure 2) and consists of (1) the higher-level sequence model manager that sets goals at a lower temporal resolution, (2) the lower-level sequence model worker that selects primitive actions at every time step by following the goals from the Manager, and (3) an internal critic that determines whether a goal is accomplished or not. More specifically, by exploiting the context from both the environment and finished goals, the manager emits a new goal for a new segment, and the worker receives the goal as guidance to generate the segment by producing words sequentially. Moreover, the internal critic is employed to evaluate whether the current textual segment is accomplished.

Furthermore, we equip both the manager and worker with an attention module over the video features (Sec 3.2) to introduce hierarchical attention internally so that the manager will focus on a wider range of temporal dynamics while the worker’s attention is narrowed down to local dynamics conditioned on the goals. To the best of our knowledge, this is the first work that strives to develop a hierarchical reinforcement learning approach to reinforce video captioning at different levels. Our main contributions are four-fold:

- We propose a hierarchical deep reinforcement learning framework to efficiently learn the semantic dynamics when captioning a video.
- We formulate an alternative, novel training approach over stochastic and deterministic policy gradient.
- We introduce a new large-scale dataset for fine-grained video captioning, Charades Captions1, and validate the effectiveness of the proposed method in it.
- We further evaluate our approach on MSR-VTT dataset and achieve the state-of-the-art results even when training on a single type of features.

1Charades Captions was obtained by preprocessing the raw Charades dataset [28]. The processed Charades Captions dataset can be downloaded here: http://www.cs.ucsb.edu/~xwang/data/CharadesCaptions.zip

2. Related Work

Video Captioning S2VT [33] first generalized LSTM to video captioning and proposed a sequence-to-sequence model for it. Since then, numerous improvements were introduced, such as attention [39, 41], hierarchical recurrent neural network (RNN) [40, 15, 2, 30, 36], C3D features [26], joint embedding space [20], language fusion [5], multi-task learning [17], etc. But most of them use the maximum-likelihood algorithm, which maximizes the probability of current ground-truth output given previous ground-truth output, while the previous ground-truth is in general unknown during test time. This inconsistency issue known as exposure bias has largely hindered the system performance.

In order to address the inconsistency issue, Ranzato et al. [21] proposed to directly optimize non-differentiable metric scores using the REINFORCE algorithm [37]. But the problem persisted that the expected gradient computed using policy gradient typically exhibited high variance and was often unstable without proper context-dependent normalization. Naturally, the variance could be reduced by adding a baseline [13, 23] or even an actor-critic method that trained an additional critic to estimate the value of each generated word [1, 22, 43]. Pasunuru and Bansal [18] applied policy gradient with baseline on video captioning and presented textual entailment loss to adjust the CIDEr reward. Unfortunately, these previous work for image/video captioning fail to grasp the high-level semantic flow. Our HRL model aims to address this issue with a hierarchical reinforcement learning framework.

Another line of work is dense video captioning [9], which focuses on detecting multiple events that occur in a video and describing each of them. But it does not aim to solve the single-sentence generation scenario. While our method aims to generate one or multiple sentences for a sequence of continuous actions (one or multiple).

Hierarchical Reinforcement Learning Recent work has revealed the effectiveness of hierarchical reinforcement learning frameworks on Atari games [11, 35]. Peng et al. built a composite dialogue policy using hierarchical Q-learning to fulfill complex dialogue tasks like traveling plans [19]. In the typical HRL setting, there was a high-level agent that operated at the lower temporal resolution to set a sub-goal, and a low-level agent that selected primitive actions by following the sub-goal from the high-level agent. Our proposed HRL framework for video captioning is aligned to these studies but has a key difference from the typical HRL setting: instead of having the internal critic to provide an intrinsic reward to encourage the low-level agent to accomplish the sub-goal, we focus on exploiting the extrinsic rewards in different time spans. Besides, we
are the first to consider HRL in the intersection vision and language.

3. Our Approach

3.1. Overview

Our proposed HRL framework follows the general encoder-decoder framework (see Figure 2). In the encoding stage, video frame features \( v = \{ v_i \} \) are first extracted by a pretrained convolutional neural network (CNN) \[ \text{model}, \] where \( i \in \{ 1, ..., n \} \) indexes the frames in the temporal order. Then the frame features are passed through a low-level Bi-LSTM\(^2\) encoder and a high-level LSTM\(^3\) encoder successively to obtain low-level encoder output \( h^E_w = \{ h^E_w_i \} \) (\( E_w \) denotes the encoder associated with the Worker), and high-level encoder output \( h^E_m = \{ h^E_m_i \} \) (\( E_m \) denoting the encoder associated with the Manager), where \( i \in \{ 1, ..., n \} \).

In the decoding stage, our HRL agent plays the role of a decoder, and outputs a language description \( a_1 a_2 ... a_T \in V^T \), where \( T \) is the length of the generated caption and \( V \) is the vocabulary set.

The HRL agent is composed of three components: a low-level worker, a high-level manager, and an internal critic. The manager operates at a lower temporal resolution and emits a goal when needed for the worker to accomplish, and the worker generates a word for each time step by following the goal proposed by the manager. In other words, the manager asks the worker to generate a semantic segment, and the worker generates the corresponding words in the next few time steps in order to fulfill the job. The internal critic determines if the worker has accomplished the goal and sends a binary segment signal to the manager to help it update goals. The whole pipeline terminates once an end of sentence token is reached.

3.2. Policy Network

Attention Module As mentioned above, the CNN-RNN encoder receives the video inputs to generate a sequence of vectors \( h^E_w = \{ h^E_w_i \} \) and \( h^E_m = \{ h^E_m_i \} \). One may directly take them as the inputs to the worker and the manager. We instead adopt an attention mechanism to better capture the temporal dynamics, and form the context vector for their use. In our model, both the manager and the worker are equipped with an attention module.

The left-hand side of Figure 3 is a demo attention module for the worker, at each time step \( t \), the context vector \( c^W_t \) is computed as a weighted sum over the encoder’s all hidden states \( \{ h^E_w_i \} \)

\[
c^W_t = \sum \alpha^W_{t,i} h^E_w_i \tag{1}
\]

These attention weights \( \{ \alpha^W_{t,i} \} \) act as an alignment mechanism by giving higher weights to certain encoder hidden states which match the worker’s current status, and are defined as

\[
\alpha^W_{t,i} = \frac{\exp(e_{t,i})}{\sum_{k=1}^{n} \exp(e_{t,k})} \tag{2}
\]

where

\[
e_{t,i} = w^T \tanh(W_a h^E_w_i + U_a h^W_{t-1} + b_a) \tag{3}
\]

where \( w, W_a, U_a \) and \( b_a \) are learned parameters; \( h^W_{t-1} \) is the worker LSTM’s hidden state at previous step.

The manager’s attention module follows the same paradigm as the worker’s, which can be described by replacing the corresponding terms in Equation 1, 2, and 3.
Manager and Worker As is shown in Figure 3, the concatenation of $[c^M_t, h^W_{t-1}]$ is fed as the input to the manager LSTM to produce the semantically meaningful goal. With the help of the context and the sentence state at previous time steps, the manager can obtain the knowledge of the environment status. The output of the manager LSTM $h^M_t$ is then projected as a latent continuous goal vector $g_t$. Formally,

$$h^M_t = S^M(h^M_{t-1}, [c^M_t, h^W_{t-1}])$$

$$g_t = u_M(h^M_t)$$

where $S^M$ denotes the non-linear function of the manager LSTM and $u_M$ is a function to project hidden states into goal space.

The worker receives the goal $g_t$, takes the concatenation of $[c^W_t, g_t, a_{t-1}]$ as the input, and outputs the probabilities $\pi_t$ over all actions $a_t \in V$ after a series of computations:

$$h^W_t = S^W(h^W_{t-1}, [c^W_t, g_t, a_{t-1}])$$

$$x_t = u_W(h^W_t)$$

$$\pi_t = \text{SoftMax}(x_t)$$

where $S^W$ is the non-linear function of the worker LSTM and $u_W$ is also a function to project hidden states into the input to softmax layer.

Internal Critic In order to determine whether the worker has accomplished a goal $g_t$, we employ an internal critic to evaluate worker’s progress. The internal critic uses an RNN structure, which takes a word sequence as the input to discriminate whether an end has been reached. Let $z_t$ denote the signal of internal critic and $h^I_t$ denote the hidden state of the RNN at time step $t$, formally we describe the probability $p(z_t)$ as follows:

$$h^I_t = \text{RNN}(h^I_{t-1}, a_t)$$

$$p(z_t) = \text{sigmoid}(W_z h^I_t + b_z)$$

where $a_t$ is the action taken by the worker and $W_z, b_z$ denotes the parameters of the feed-forward neural network. In order to train the parameters of the linear layer and recurrent network, we propose to maximize the likelihood of given ground truth signal $\{z^*_t\}$:

$$\arg \max_t \sum \log p(z^*_t | a_1, \ldots, a_{t-1})$$

Once the critic model is optimized, we will fix it to service the usage of the manager.

3.3. Learning

As described in Sec. 3.2, the manager policy is actually deterministic, which can be further denoted as $g_t = \mu_{\theta_m}(s_t)$ with $\theta_m$ representing the parameters of the manager, while the worker policy is a stochastic policy denoted by $a_t \sim \pi_{\theta_w}(a_t | s_t, g_t)$, where $\theta_w$ represents the parameters of the worker. The reason why the worker policy is stochastic is that its action $a_t$ is selecting a word from the vocabulary $V$. But for the manager, the generated goal is latent, which cannot be directly supervised. Thus with a deterministic manager policy, we can warm start both the manager and worker simultaneously by viewing them as a composite agent.

In this section, we first derive the mathematical reinforcement learning methods for the policies separately (Sec. 3.3.1 and 3.3.2), and then introduce the training algorithm of the proposed HRL method (Sec. 3.3.4). We also discuss the reward definitions (Sec. 3.3.3) and imitation learning of our HRL policy (Sec. 3.3.5).

3.3.1 Stochastic Worker Policy Learning

We consider a standard reinforcement learning setup. At each step $t$, the worker select an action $a_t (a_t \in V)$ conditioned on $g_t$ from the manager. The environment responds with a new state $s_{t+1}$ and a scalar reward $r_t$. The process continues until a $<$EOS$>$ token is generated. The objective of the worker is to maximize the discounted return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$. Thus its loss function can be written as

$$L(\theta_w) = -\mathbb{E}_{a_t \sim \pi_{\theta_w}} [R(a_t)]$$

(11)

to minimize the negative expected reward function. Based on REINFORCE algorithm [37], the gradient of non-differentiable, reward-based loss function can be derived as

$$\nabla_{\theta_w} L(\theta_w) = -\mathbb{E}_{a_t \sim \pi_{\theta_w}} [R(a_t) \nabla_{\theta_w} \log \pi_{\theta_w}(a_t)]$$

(12)
In practice \( L(\theta_w) \) is typically estimated with a single sample from \( \pi_{\theta_w} \):

\[
\nabla_{\theta_w} L(\theta_w) \approx -R(a_t)\nabla_{\theta_w} \log \pi_{\theta_w}(a_t)
\]

(13)

The policy gradient given by REINFORCE can be further generalized to reduce the variance without changing the expected gradient, by subtracting the reward with a baseline [31]:

\[
\nabla_{\theta_w} L(\theta_w) \approx - (R(a_t) - b_t^w) \nabla_{\theta_w} \log \pi_{\theta_w}(a_t)
\]

(14)

where \( b_t^w \) is the estimated baseline, which can be a function of \( \theta_w \) or \( t \) [21]. In our case, the baseline is estimated by a linear regressor with the worker’s hidden state \( h_t^W \) as the input. During back propagation, the gradient passing is cut off between the worker LSTM and the baseline estimator.

For a better understanding of the policy gradient, we can further derive the loss function using the chain rule

\[
\nabla_{\theta_w} L(\theta_w) = \sum_{t=1}^{T} \frac{\partial L}{\partial x_t} \frac{\partial x_t}{\partial \theta_w}
\]

(15)

where \( x_t \) is the input to the SoftMax layer (see Equation 7). Using REINFORCE with baseline the estimation of \( \frac{\partial L}{\partial x_t} \) is given by [42]:

\[
\frac{\partial L}{\partial x_t} = (R(a_t) - b_t^w)(\pi_{\theta_w}(a_t) - 1_{a_t})
\]

(16)

which means if the reward \( R(a_t) \) of the sample word \( a_t \) is greater than the baseline \( b_t \), the gradient is negative and thus the model encourages the distribution by increasing the probability of the word, otherwise, it discourages the distribution accordingly.

### 3.3.2 Deterministic Manager Policy Learning

The key to our HRL framework is to effectively learn the goal \( g_t \) generated by the manager and then guides the worker to achieve the latent objective. But the difficulty of training the manager is that it does not directly interact with the environment because the action it takes is to produce a latent vector \( g_t \) in a continuous high-dimensional space, which indirectly influences the environment by directing the Worker’s behavior. Therefore, we are especially interested in coming up solutions to encourage the manager towards more effective caption generation.

Inspired by the deterministic policy gradient algorithms [29, 12], we propose to learn the deterministic policy \( \mu_{\theta_m}(s_t) \) from trajectories generated by the stochastic worker policy \( \pi_{\theta_w}(a_t; s_t, g_t) \). When training the target manager policy, we fix the worker policy as an Oracle behavior policy. More specifically, the manager outputs a goal \( g_t \) at step \( t \) and the worker then runs \( c \) steps to generate the expected segment \( e_{t,c} = a_t, a_{t+1}, \ldots, a_{t+c-1} \) by following the goal \( \epsilon \) is length of the generated segment. Since the worker is fixed as an Oracle behavior policy, we only need to consider the training of the manager. Then the environment responds with an new state \( s_{t+c} \) and a scalar reward \( r(e_{t,c}) \). Thus the objective becomes minimizing the negative discounted return \( R_t(e_{t,c}) \), in formula

\[
L(\theta_m) = -\mathbb{E}_{g_t}[R(e_t)\pi(e_t; s_t, g_t) = \mu_{\theta_m}(s_t)]
\]

(17)

After applying the chain rule to the loss function with respect to the manager’s parameters \( \theta_m \), the manager is updated with

\[
\nabla_{\theta_m} L(\theta_m) = -\mathbb{E}_{g_t}[R(e_t)\nabla_{\theta_m} \pi(e_t; s_t, g_t) \nabla_{\theta_m} \mu_{\theta_m}(s_t)]
\]

(18)

The above gradients can be approximated from a single sampled segment \( e_{t,c} \) and after adopting policy gradient on the worker policy,

\[
\nabla_{\theta_m} L(\theta_m) = -R(e_t)\nabla_{\theta_m} \log \pi(e_t; s_t) \nabla_{\theta_m} \mu_{\theta_m}(s_t)
\]

(19)

Since the worker LSTM is indeed a Markov decision process and the probability of the current action \( a_t \) is conditioned on the action \( a_{t-1} \) at previous step (see Equation 6,7,8), we have

\[
\log \pi(e_{t,c}) = \log \pi(a_{t..a_{t+c-1}}) = \sum_{i=t}^{t+c-1} \log \pi(a_i)
\]

(20)

Combining Equation 19 and 20, then the gradients become

\[
\nabla_{\theta_m} L(\theta_m) = -R(e_t) \sum_{i=t}^{t+c-1} \nabla_{\theta_m} \log \pi(a_i) \nabla_{\theta_m} \mu_{\theta_m}(s_t)
\]

(21)

The final gradients for the manager training is obtained by adding the baseline estimator to reduce the variance as follows:

\[
\nabla_{\theta_m} L(\theta_m) = \\
-R(e_t) \sum_{i=t}^{t+c-1} \nabla_{\theta_m} \log \pi(a_i) \nabla_{\theta_m} \mu_{\theta_m}(s_t)
\]

(22)

where \( b_t^m \) is the baseline estimator, which is a linear regressor with the manager’s hidden state \( h_t^M \) as the input.

A major challenge of learning in continuous action spaces is exploration. We follow the known DDPG [12] to construct an exploration policy \( \mu' \) by adding perturbation \( \epsilon \) sampled from a Gaussian distribution \( \mathcal{N} \) to our manager policy

\[
\mu'(s_t) = \mu_{\theta_m}(s_t) + \epsilon
\]

(23)

and the variance of Gaussian noise can be chosen to suit the environment.
3.3.3 Reward Definition

Recent work on image captioning [23] has shown that CIDEr as a reward performs the best among the traditional evaluation metrics (e.g., CIDEr, BLEU or METEOR) for image/video captioning and can gain improvement on all other metrics. In our model, we also use CIDEr score to compute the reward. But instead of directly using the final CIDEr score of the whole generated caption as the reward for each word \(a_t\), we adopt delta CIDEr score as the immediate reward. Let \(f(x) = \text{CIDEr}(\text{sent} + x) - \text{CIDEr}(\text{sent})\), where \text{sent} is the previous generated caption. Then the discounted return for the worker is

\[
R(a_t) = \sum_{k=0}^{\infty} \gamma^k f(a_{t+k})
\]

where \(k\) denotes the time step of the worker’s temporal resolution, and the discounted return for the manager is

\[
R(e_t) = \sum_{n=0}^{\infty} \gamma^n f(e_{t+n})
\]

where \(n\) is the time step of the manager’s lower temporal resolution. Note that our approach is not limited to CIDEr score, other reasonable rewards (e.g. deltaBLEU [4]) can also be applied to the HRL framework.

3.3.4 Training Algorithm

Above we illustrate the learning methods to train the manager and the worker. In Algorithm 1 we present the pseudocode of our HRL training algorithm for video captioning. The manager policy and the worker policy are trained alternately. Basically, when training the worker, we assume the manager is well-posed, so we disable the goal exploration and only update the worker policy according to Equation 14; when training the manager, we treat the worker as the Oracle behavior policy, so we generate the caption by greedy decoding and only update the manager policy following Equation 22.

During testing, goal exploration is disabled, and beam search is employed to generate the results. Only one forward pass is needed at test time.

Algorithm 1 HRL training algorithm

Require: Training pairs <video, GT caption>
1: Randomly initialize the model parameters \(\theta\)
2: Load the pretrained CNN model and internal critic
3: for iteration=1,M do
4:    Randomly sample a minibatch
5:    if Train-Worker then
6:        Disable the goal exploration
7:        Run a forward pass to get the sampled caption \(a_1a_2...a_T\)
8:        Calculate \(R(a_t)\) for each \(a_t\)
9:        Freeze the manager
10:       Update the worker policy using Equation 14
11:   else if Train-Manager then
12:       Initialize a random process \(N\) for goal exploration
13:       Run a forward pass to get the greedily decoded caption \(c_1c_2...c_n\)
14:       Calculate \(R(e_t)\) for each \(e_t\)
15:       Freeze the worker
16:       Update the manager policy using Equation 22
17:  end if
18: end for

3.3.5 Imitation Learning

A major challenge for a reinforcement learning agent to have good convergence property is that the agent must start with a good policy at the beginning stage. For our model, we apply the cross-entropy loss optimization to warm start both the worker and the manager simultaneously, where the manager is completely treated as the latent parameters. \(\theta\) be the parameters of the whole model and \(a_1^*, a_2^*, ..., a_T^*\) be the ground-truth word sequence, then the cross-entropy loss is defined as

\[
L(\theta) = -\sum_{t=1}^{T} \log(\pi_{\theta}(a_t^*; a_1^*, ..., a_{t-1}^*))
\]

4. Experimental Results

4.1. Datasets

MSR-VTT MSR-VTT [38] is a dataset for general video captioning, which is derived from a wide variety of video categories (7,180 videos from 20 general categories), and contains 10,000 video clips (6,513 for training, 497 for validation, and the remaining 2,990 for testing). Each video contains 20 human annotated reference captions collected by Amazon Mechanical Turk (AMT).

Charades Captions Charades [28] is a large-scale dataset composed of 9,848 videos of daily indoors activities collected through AMT. 267 different users were presented with a sentence script (e.g. a person fixes the bed then throws pillow on it) that included objects and actions from a fixed vocabulary, and the users recorded a video following the script using provided objects and actions. The original dataset contains 66,500 temporal annotations for 157 action classes, 41,104 labels for 46 object classes, and 27,847 textual descriptions of the videos.
Table 1: Comparison with state of the arts on MSR-VTT dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU@4</th>
<th>METEOR</th>
<th>ROUGE-L</th>
<th>CIDEr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean-Pooling</td>
<td>30.4</td>
<td>23.7</td>
<td>52.0</td>
<td>35.0</td>
</tr>
<tr>
<td>Soft-Attention</td>
<td>28.5</td>
<td>25.0</td>
<td>53.3</td>
<td>37.1</td>
</tr>
<tr>
<td>S2VT</td>
<td>31.4</td>
<td>25.7</td>
<td>55.9</td>
<td>35.2</td>
</tr>
<tr>
<td>v2t_navigator</td>
<td>40.8</td>
<td>28.2</td>
<td>60.9</td>
<td>44.8</td>
</tr>
<tr>
<td>Aalto</td>
<td>39.8</td>
<td>26.9</td>
<td>59.8</td>
<td>45.7</td>
</tr>
<tr>
<td>VideoLAB</td>
<td>39.1</td>
<td>27.7</td>
<td>60.6</td>
<td>44.1</td>
</tr>
<tr>
<td>XE-baseline</td>
<td>41.3</td>
<td>27.6</td>
<td>59.9</td>
<td>44.7</td>
</tr>
<tr>
<td>RL-baseline</td>
<td>40.6</td>
<td>28.5</td>
<td>60.7</td>
<td>46.3</td>
</tr>
<tr>
<td>HRL (Ours)</td>
<td>41.3</td>
<td>28.7</td>
<td>61.7</td>
<td>48.0</td>
</tr>
</tbody>
</table>

Table 2: Results on Charades Captions dataset. We reported BLEU (B), METEOR (M), ROUGH-L (R) and CIDEr (C) scores of our HRL method and two baselines for comparison.

<table>
<thead>
<tr>
<th>Method</th>
<th>B@1</th>
<th>B@2</th>
<th>B@3</th>
<th>B@4</th>
<th>M</th>
<th>R</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>XE-baseline</td>
<td>55.0</td>
<td>36.4</td>
<td>23.6</td>
<td>15.0</td>
<td>18.7</td>
<td>39.0</td>
<td>16.7</td>
</tr>
<tr>
<td>RL-baseline</td>
<td>57.6</td>
<td>41.4</td>
<td>28.0</td>
<td>18.8</td>
<td>17.7</td>
<td>39.8</td>
<td>21.6</td>
</tr>
<tr>
<td>HRL-16</td>
<td>64.4</td>
<td>44.3</td>
<td>29.4</td>
<td>18.8</td>
<td>19.5</td>
<td>41.4</td>
<td>23.2</td>
</tr>
<tr>
<td>HRL-32</td>
<td>64.0</td>
<td>43.4</td>
<td>28.4</td>
<td>17.9</td>
<td>19.2</td>
<td>41.0</td>
<td>21.3</td>
</tr>
<tr>
<td>HRL-64</td>
<td>61.7</td>
<td>43.0</td>
<td>28.8</td>
<td>18.8</td>
<td>18.7</td>
<td>31.2</td>
<td>23.6</td>
</tr>
</tbody>
</table>

While the Charades dataset is mainly used for action recognition and segmentation, one should note that the collected textual descriptions are very detailed and depict the fine-grained human activities happening in long videos. Thus, we preprocessed the raw Charades dataset by combining the textual descriptions and sentence scripts verified through AMT\(^4\), and built a new large-scale dataset for detailed video captioning – Charades Captions, which consists of 6,963 videos for training, 500 for validation and 1,760 for testing. Each video clip is annotated by multiple (typically 2-5) captions. The captions are more detailed and longer than those of MSR-VTT (average caption length: 24.13 vs 9.28 words), which is more suitable for fine-grained video captioning.

Caption Segmentation In order to train the internal critic that determines if a goal is accomplished, we preprocessed the ground truth captions of the training sets of both datasets by breaking each caption into multiple semantic chunks. We segmented the captions mainly based on the Noun Phrase (NP) and Verb Phrase (VP) labels provided by the constituency parsing results (We utilized the open source toolkits Stanford CoreNLP\(^3\) [14] and NLTK\(^6\) for constituency parsing). For instance, the caption The person then tidies his area after he is done eating was segmented into three sub-phrases, The person, then tidies his area and after he is done eating with labels NP, VP and VP respectively. However, all we need to train the internal critic were the chunks, and labels were not used.

4.2. Experimental Setup

Evaluation Metrics We adopted four diverse automatic evaluation metrics: BLEU, METEOR, ROUGE-L, and CIDEr-D. We used the standard evaluation code from MSCOCO server [3] to obtain the results.

Training Details All the hyperparameters were tuned on the validation set. For both datasets, we sampled each video at 3fps and extracted ResNet-152 features [6] from these sampled frames without fine-tuning. More training details can be found in the supplementary material.

4.3. Results and Analysis

Comparison with state of the arts on MSR-VTT In Table 1, we compared our single-sentence captioning results with the-state-of-the-art methods on MSR-VTT dataset. We listed the results of Mean-Pooling [34], Soft-Attention [39] and S2VT [33] as reported in previous work [26]. We also compared with the top-3 results from MSR-VTT challenge, including v2t navigator [8], Aalto [27], VideoLAB [20]. We implemented two baseline methods: an attention-based sequence-to-sequence model trained with cross-entropy loss (XE-baseline), and the same model trained with policy gradient and CIDEr score as the RL reward (RL-baseline). As shown in Table 1, our XE-baseline achieved comparable results with the state-of-the-art results, and our RL-baseline further improved on all metrics. Moreover, our novel HRL method outperformed all the other algorithms listed in the table, which proved the effectiveness of our proposed method.

Result Analysis on Charades Captions Since there were no other papers reporting results on Charades Captions, we mainly compared our HRL model with our implementation of XE-baseline and RL-baseline. Meanwhile, we explored the dimension of the latent goal vector (We used HRL-X to denote the HRL model with a goal dimension of X). As can be observed from Table 2, all our HRL models outperformed the baseline methods and brought significant improvements in different evaluation metrics. Note that our HRL model achieved bigger improvement over the baseline methods on Charades Captions dataset than on MSR-VTT. Given that fact that the average caption length of Charades Captions was much longer than that of MST-VTT (24.13 vs 9.28 words), the difference of the improvement gaps demonstrated that our HRL model can gain better improvement on detailed descriptions of longer videos.

Among the HRL models, HRL-16 achieved the best on
Learning Curve
For a more intuitive view of the models, we drew the learning curves of the CIDEr scores on validation set (see Figure 5). Note that the RL-baseline model was first warmed up with cross-entropy loss, and then improved using the REINFORCE algorithm. Particularly, after we trained the XE-baseline model, we switched to policy gradient and continued training the RL-baseline model on it. HRL models were resumed training on a shorter warm-start period. As is shown in Figure 5, the HRL models converged faster and achieved better peak points than the baseline methods. HRL-16 reached the highest point.

5. Conclusion
In this paper, we propose a hierarchical reinforcement learning framework for video captioning, which aims at improving the fine-grained generation of video descriptions with rich activities. Our HRL model obtains the state-of-the-art performance on both the widely used MSR-VTT dataset and the newly introduced Charades Captions dataset for fine-grained video captioning.

In the future, we plan to explore the attention space and utilize features from multiple modalities to boost our HRL agent. We believe that the results of our method can be further improved by employing different types of features, i.e. C3D features [32], optical flows, etc. Meanwhile, we will investigate the HRL framework in other similar sequence generation tasks like video/document summarization.

Acknowledgement
We would thank Ramakanth Pasunuru and Ruotian Luo for clarifying the technical details of their paper/code, and Wenhan Xiong for his help on debugging the model. Personally, Xin would appreciate the care from his girlfriend (now his wife) when he was busy working on the paper.
References


