ADAPTIVE FEATURE ABSTRACTION FOR TRANSLATING VIDEO TO LANGUAGE

Yunchen Pu *
Department of Electrical and Computer Engineering
Duke University
yunchen.pu@duke.edu

Martin Renqiang Min
Machine Learning Group
NEC Laboratories America
renqiang@nec-labs.com

Zhe Gan
Department of Electrical and Computer Engineering
Duke University
zhe.gan@duke.edu

Lawrence Carin
Department of Electrical and Computer Engineering
Duke University
lcarin@duke.edu

ABSTRACT

Previous models for video captioning often use the output from a specific layer of a Convolutional Neural Network (CNN) as video representations, preventing them from modeling rich, varying context-dependent semantics in video descriptions. In this paper, we propose a new approach to generating adaptive spatiotemporal representations of videos for a captioning task. For this purpose, novel attention mechanisms with spatiotemporal alignment is employed to adaptively and sequentially focus on different layers of CNN features (levels of feature "abstraction"), as well as local spatiotemporal regions of the feature maps at each layer. Our approach is evaluated on three benchmark datasets: YouTube2Text, M-VAD and MSR-VTT. Along with visualizing the results and how the model works, these experiments quantitatively demonstrate the effectiveness of the proposed adaptive spatiotemporal feature abstraction for translating videos to sentences with rich semantics.

1 INTRODUCTION

Videos represent among the most widely used forms of data, and their accurate characterization poses an important challenge for computer vision and machine learning. Generating a natural-language description of a video, termed video captioning, is an important component of video analysis. Inspired by the successful encoder-decoder framework used in machine translation (Cho et al., 2014; Bahdanau et al., 2015; Sutskever et al., 2014) and image caption generation (Kiros et al., 2014; Vinyals et al., 2015; Karpathy & Li, 2015; Mao et al., 2015; Pu et al., 2016; Gan et al., 2016), most recent work on video captioning (Donahue et al., 2015; Venugopalan et al., 2015a;b; Yao et al., 2015; Pan et al., 2016b; Yu et al., 2016) employs a two-dimensional (2D) or three-dimensional (3D) Convolutional Neural Network (CNN) as an encoder, mapping an input video to a compact feature-vector representation. A Recurrent Neural Network (RNN) is typically employed as a decoder, unrolling the feature vector to generate a sequence of words of arbitrary length.

Despite achieving encouraging success in video captioning, previous models suffer important limitations. Often the rich video content is mapped to a single feature vector for caption generation; this approach is prone to miss detailed and localized spatiotemporal information. To mitigate this, one may employ methods to focus attention on local regions of the feature map, but typically this

*Most of this work was done when the author was an intern at NEC Labs America.

is done with features from a selected (usually top) CNN layer. By employing features from a fixed CNN layer, the algorithm is limited in its ability to model rich, context-aware semantics that requires focusing on different feature abstraction levels. As investigated in Mahendran & Vedaldi (2015); Zeiler & Fergus (2014), the feature characteristics/abstraction is correlated with the CNN layer: features from layers at or near the top of a CNN tend to focus on global (extended) visual percepts, while features from lower CNN layers provide more local, fine-grained information. It is desirable to select/weight features from different CNN layers adaptively when decoding a caption, selecting different levels of feature abstraction by sequentially emphasizing features from different CNN layers. In addition to focusing on features from different CNN layers, it is also desirable to emphasize local spatiotemporal regions in feature maps at particular layers.

To realize these desiderata, our proposed decoding process for generating a sequence of words dynamically emphasizes different levels (CNN layers) of 3D convolutional features, to model important coarse or fine-grained spatiotemporal structure. Additionally, the model adaptively attends to different locations within the feature maps at particular layers. While some previous models use 2D CNN features to generate video representations, our model adopts features from a deep 3D convolutional neural network (C3D). Such features have been shown to be effective for video representations, action recognition and scene understanding (Tran et al., 2015), by learning the spatiotemporal features that can provide better appearance and motion information. In addition, the proposed model is inspired by the recent success of attention-based models that mimic human perception (Mnih et al., 2014; Xu et al., 2015).

Our proposed model, adaptive spatiotemporal feature representation with dynamic feature abstraction, involves comparing and evaluating different levels of 3D convolutional feature maps. However, this has three challenges to overcome to directly compare features between layers: (*i*) the features from different C3D levels have distinct dimensions, undermining the use of a multi-layer perceptron (MLP) (Xu et al., 2015; Bahdanau et al., 2015) based attention model; (*ii*) the features represented in each layer are not spatiotemporally aligned, undermining our ability to quantify the value of features at a specific spatiotemporal location, based on information from *all* CNN layers; and (*iii*) the semantic meaning of feature vectors from the convolutional filters of C3D varies across layers, implying that the layer-dependent features are in different semantic spaces and making it difficult to feed the features into the same RNN decoder.

To address these issues, one may use either pooling or MLPs to map different levels of features to the same semantic-space dimension. However, these approaches either lose a lot of feature abstraction information or have too many parameters to generalize well. In our approach, we employ convolution operations to elegantly achieve spatiotemporal alignment among C3D features from different levels and attention mechanisms to dynamically select context-dependent feature abstraction information.

The principal contributions of this paper are as follows: (*i*) A new video-caption-generation model is developed by dynamically modeling context-dependent feature abstractions;(*ii*) New attention mechanisms to adaptively and sequentially emphasize different levels of feature abstraction (CNN layers), while also imposing attention within local spatiotemporal regions of the feature maps at each layer are employed; (*iii*) 3D convolutional transformations are introduced to achieve spatiotemporal and semantic feature consistency across different layers; (*iv*) We demonstrate that the proposed model outperforms other multi-level feature based methods such as hypercolumns (Hariharan et al., 2015) and achieves state-of-the-art performance on several benchmark datasets. We call the proposed algorithm Adaptive SpatioTemporal representation with dynAmic abstRaction (ASTAR).

2 Related Work

Early work on video captioning used a two-step approach, employing role-word detection (*e.g.*, subject, verb and object) and rules for language grammar. In such work, the sentence for video description is first split into several parts, each of which is aligned with visual content. For example, Rohrbach et al. (2013; 2014) learn a Conditional Random Field (CRF) to infer high-level concepts such as object and action; in Guadarrama et al. (2013) semantic hierarchies are used to choose an appropriate level of sentence fragments. Statistical language models, learned from large text corpora, are used to translate the semantic representation to a grammatically correct sentence.

Recent work often develops a probabilistic model of the caption, conditioned on a video. Donahue et al. (2015); Venugopalan et al. (2015a;b); Yu et al. (2016); Pan et al. (2016a) applied a 2D CNN pretrained on ImageNet to video frames, with the top-layer output of the CNN used as features. Given the *sequence* of features extracted from video frames, the video representation is then obtained by a CRF (Donahue et al., 2015), mean pooling (Venugopalan et al., 2015b), weighted mean pooling with attention (Yu et al., 2016), or via the last hidden state of an RNN encoder (Venugopalan et al., 2015a). Yao et al. (2015) replace the 2D CNN features with a 3D CNN to model the short temporal dynamics. These works were followed by Pan et al. (2016b), which jointly embedded the 2D CNN features and spatiotemporal features extracted from a 3D CNN (Tran et al., 2015). However, all of these previous models utilize features extracted from the top layer of the CNN.

There are also some work that combines mult-level features of CNN. Sermanet et al. (2013) employed a combination of intermediate layers with the top layer for pedestrian detection while Hariharan et al. (2015) utilized hypercolumn representation for object segmentation and localization. Our proposed model is mostly related to Ballas et al. (2016), but distinct in important ways. The intermediate convolutional feature maps are leveraged, like Ballas et al. (2016), but an attention model is developed instead of the "stack" RNN in Ballas et al. (2016). In addition, a powerful decoder enhanced with two attention mechanisms is constructed for generating captions, while a simple RNN decoder is employed in Ballas et al. (2016). Finally, we use features extracted from C3D instead of a 2D CNN.

3 METHOD

Consider N training videos, the nth of which is denoted $\mathbf{X}^{(n)}$, with associated caption $\mathbf{Y}^{(n)}$. The length-T_n caption is represented $\mathbf{Y}^{(n)} = (\boldsymbol{y}_1^{(n)}, \ldots, \boldsymbol{y}_{T_n}^{(n)})$, with $\boldsymbol{y}_t^{(n)}$ a 1-of-V ("one hot") encoding vector, with V the size of the vocabulary.

For each video, the C3D feature extractor (Tran et al., 2015) produces a set of features $\mathbf{A}^{(n)} = \{\boldsymbol{a}_1^{(n)}, \ldots, \boldsymbol{a}_L^{(n)}, \boldsymbol{a}_{L+1}^{(n)}\}$, where $\{\boldsymbol{a}_1^{(n)}, \ldots, \boldsymbol{a}_L^{(n)}\}$ are feature maps extracted from L convolutional layers, and $\boldsymbol{a}_{L+1}^{(n)}$ is a vector obtained from the top fully-connected layer.

The convolutional-layer features, $\{\boldsymbol{a}_1^{(n)}, \ldots, \boldsymbol{a}_L^{(n)}\}$, are extracted by feeding the entire video into C3D, and hence the dimensions of $\{\boldsymbol{a}_1^{(n)}, \ldots, \boldsymbol{a}_L^{(n)}\}$ are dependent on the video length (number of frames). As discussed below, we employ spatiotemporal attention at each layer (and between layers), and therefore it is not required that the sizes of $\{\boldsymbol{a}_1^{(n)}, \ldots, \boldsymbol{a}_L^{(n)}\}$ be the same for all videos. However, the fully connected layer at the top, responsible for $\boldsymbol{a}_{L+1}^{(n)}$, assumes that $\boldsymbol{a}_L^{(n)}$ is of the same size for all videos (like the 16-frame-length videos in (Tran et al., 2015)). To account for variable-length videos, and maintain the same fully-connected layer at the top, we employ mean pooling to $\boldsymbol{a}_L^{(n)}$, based on a window of length 16 (as in (Tran et al., 2015)) with an overlap of 8 frames. The particular form of pooling used here (one could also use max pooling) is less important than the need to make the dimension of the top-layer features the same for feeding into the final fully-connected layer.

3.1 CAPTION MODEL

For notational simplicity, henceforth we omit superscript n. The t-th word in a caption, $\boldsymbol{y}_t$, is mapped to an M-dimensional vector $\boldsymbol{w}_t = \mathbf{W}_e \boldsymbol{y}_t$, where $\mathbf{W}_e \in \mathbb{R}^{M \times V}$ is a learned word-embedding matrix, *i.e.*, $\boldsymbol{w}_t$ is a column of $\mathbf{W}_e$ chosen by the one-hot $\boldsymbol{y}_t$. The probability of caption $\mathbf{Y} = \{\boldsymbol{y}_t\}_{t=1,T}$ is defined as

$$p(\mathbf{Y}|\mathbf{A}) = p(\boldsymbol{y}_1|\mathbf{A}) \prod_{t=2}^{T} p(\boldsymbol{y}_t|\boldsymbol{y}_{<t}, \mathbf{A}) \,. \tag{1}$$

Specifically, the first word $\boldsymbol{y}_1$ is drawn from $p(\boldsymbol{y}_1|\mathbf{A}) = \text{softmax}(\mathbf{V}\boldsymbol{h}_1)$, where $\boldsymbol{h}_1 = \tanh(\mathbf{C}\boldsymbol{a}_{L+1})$. Bias terms are omitted for simplicity throughout the paper. All the other words in the caption are then sequentially generated using an RNN, until the end-sentence symbol is generated. Conditional distribution $p(\boldsymbol{y}_t|\boldsymbol{y}_{<t}, \mathbf{A})$ is specified as $\text{softmax}(\mathbf{V}\boldsymbol{h}_t)$, where $\boldsymbol{h}_t$ is recursively updated as $\boldsymbol{h}_t = \mathcal{H}(\boldsymbol{w}_{t-1}, \boldsymbol{h}_{t-1}, \boldsymbol{z}_t)$. $\mathbf{V}$ is a matrix connecting the RNN hidden state to a softmax, for computing a distribution over words. $\boldsymbol{z}_t = \phi(\boldsymbol{h}_{t-1}, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_L)$ is the context vector used in

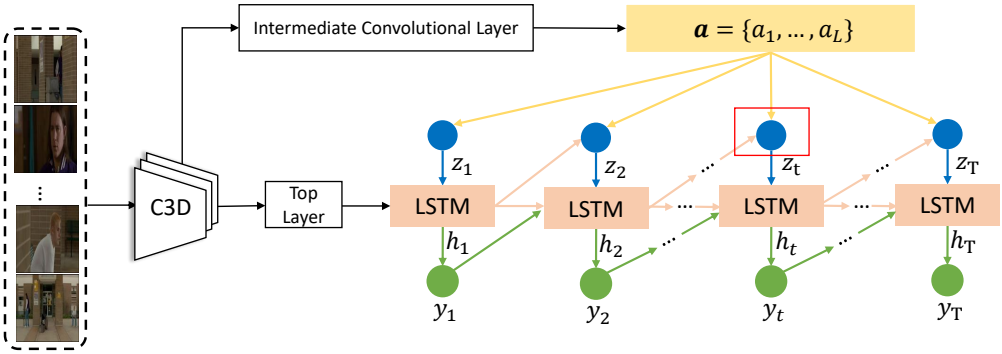


Figure 1: Illustration of our proposed caption-generation model. The model leverages a fully-connected map from the top layer as well as convolutional maps from different mid-level layers of a pretrained 3D convolutional neural network (C3D). The context vector $\boldsymbol{z}_t$ is generated from previous hidden unit $\boldsymbol{h}_{t-1}$ and the convolutional maps $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_L\}$ (the red frame) which is detailed in Fig.2

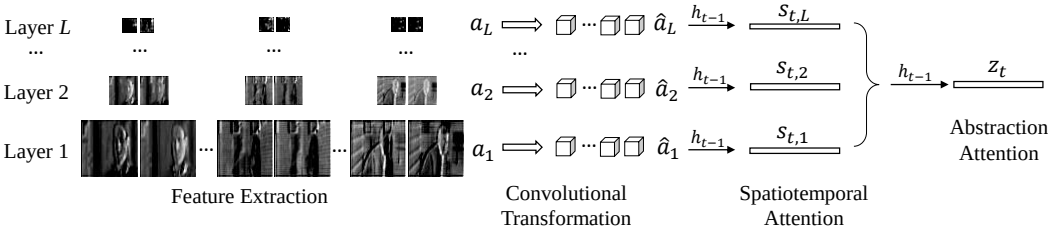


Figure 2: Illustration of our attention mechanism. The video features are extracted by C3D, followed by a 3D convolutional transformation. At each time step, the spatiotemporal attention takes these features and previous hidden units to generate L feature vectors which are fed to the abstraction attention, manifesting the context vector.

the attention mechanism, capturing the relevant visual features associated with the spatiotemporal attention (also weighting level of feature abstraction), as detailed in Sec. 3.2.

Note that the output of the fully-connected-layer, $\boldsymbol{a}_{L+1}$, is only used to generate the first word (encapsulating overall-video features). We found that only using $\boldsymbol{a}_{L+1}$ there works better in practice than using it at each time step of the RNN, as also found in Vinyals et al. (2015); Venugopalan et al. (2015b).

The transition function $\mathcal{H}(\cdot)$ is implemented with Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997). At time t, the LSTM unit consists of a memory cell $\boldsymbol{c}_t$ and three gates: the input $\boldsymbol{i}_t$, forget $\boldsymbol{f}_t$, and output $\boldsymbol{o}_t$ gates. The memory cell transmits the information from the previous step to the current step, while the gates control reading or writing the memory unit through sigmoid functions. Specifically, the hidden units $\boldsymbol{h}_t$ are updated as follows:

$$
\begin{aligned}
\boldsymbol{i}_t &= \sigma(\mathbf{W}_{iw}\boldsymbol{w}_{t-1} + \mathbf{W}_{ih}\boldsymbol{h}_{t-1} + \mathbf{W}_{iz}\boldsymbol{z}_t), & \boldsymbol{f}_t &= \sigma(\mathbf{W}_{fw}\boldsymbol{w}_{t-1} + \mathbf{W}_{fh}\boldsymbol{h}_{t-1} + \mathbf{W}_{fz}\boldsymbol{z}_t), \\
\boldsymbol{o}_t &= \sigma(\mathbf{W}_{ow}\boldsymbol{w}_{t-1} + \mathbf{W}_{oh}\boldsymbol{h}_{t-1} + \mathbf{W}_{oz}\boldsymbol{z}_t), & \tilde{\boldsymbol{c}}_t &= \tanh(\mathbf{W}_{cw}\boldsymbol{w}_{t-1} + \mathbf{W}_{ch}\boldsymbol{h}_{t-1} + \mathbf{W}_{cz}\boldsymbol{z}_t), \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t, & \boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t),
\end{aligned} \tag{2}
$$

where $\sigma(\cdot)$ and $\odot$ denote the sigmoid function and the element-wise multiply operator, respectively. Matrices $\mathbf{W}_{\{i,f,o,c\}}$, $\mathbf{V}$ and $\mathbf{C}$ represent the set of LSTM parameters that will be learned (plus associated biases).

Given the video $\mathbf{X}$ (with features $\mathbf{A}$) and associated caption $\mathbf{Y}$, the objective function is the sum of the log-likelihood of the caption conditioned on the video representation:

$$
\log p(\mathbf{Y}|\mathbf{A}) = \log p(\boldsymbol{y}_1|\mathbf{A}) + \textstyle\sum_{t=2}^{T} \log p(\boldsymbol{y}_t|\boldsymbol{y}_{<t}, \mathbf{A}) \,, \tag{3}
$$

Equation (3) is a function of all model parameters to be learned; they are not explicitly depicted in (3) for notational simplicity. Further, (3) corresponds to a single video-caption pair, and when training we sum over all such training pairs. Our proposed model is illustrated in Figure 1.

3.2 ATTENTION MECHANISM

We first define notation needed to describe attention mechanism $\phi(\boldsymbol{h}_{t-1}, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_L)$. Each feature map, $\boldsymbol{a}_l$, is a 4D tensor, with elements corresponding to two spatial coordinates ($i.e.$, vertical and horizontal dimensions in a given frame), third tensor index in the frame-index dimension, and a fourth dimension associated with the filter index (for the convolutional filters). To be explicit, at CNN layer l, the number of dimensions of this tensor are denoted $n_x^l \times n_y^l \times n_z^l \times n_F^l$, with respective dimensions corresponding to vertical, horizontal, frame, and filter ($e.g.$, n_F^l convolutional filters at layer l). Note that dimensions n_x^l, n_y^l and n_z^l vary with layer level l (getting smaller with increasing l, due to pooling).

We define $\boldsymbol{a}_{i,l}$ as a n_F^l-dimensional *vector*, corresponding to a fixed coordinate in three of the tensor dimensions, $i.e.$, $i \in [1, \ldots, n_x^l] \times [1, \ldots, n_y^l] \times [1, \ldots, n_z^l]$, while sweeping across all n_F^l feature/filter dimensions. Further, define $\boldsymbol{a}_l(k)$ as a 3D tensor, associated with 4D tensor $\boldsymbol{a}_l$. Specifically, $\boldsymbol{a}_l(k)$ corresponds to the 3D tensor manifested from $\boldsymbol{a}_l$, with $k \in \{1, \ldots, n_F^l\}$ a fixed coordinate in the dimension of the filter index. Hence, $\boldsymbol{a}_l(k)$ corresponds to the 3D feature map at layer l, due to the kth filter at that layer.

We introduce two attention mechanisms when predicting word $\boldsymbol{y}_t$: (i) spatiotemporal-localization attention, and (ii) abstraction-level attention; these, respectively, measure the relative importance of a particular spatiotemporal location and a particular CNN layer (feature abstraction) for producing $\boldsymbol{y}_t$, based on the word-history information $\boldsymbol{y}_{<t}$.

To achieve this, we seek to map $\boldsymbol{a}_l \rightarrow \hat{\boldsymbol{a}}_l$, where 4D tensors $\hat{\boldsymbol{a}}_l$ all have the same dimensions, are embedded into same semantic spaces, and are aligned spatialtemporally. Specifically, $\hat{\boldsymbol{a}}_l$, $l = 1, \ldots, L-1$ are aligned in the above ways with $\boldsymbol{a}_L$. To achieve this, we filter each $\boldsymbol{a}_l$, $l = 1, \ldots, L-1$, and then apply max-pooling; the filters seek semantic alignment of the features (including feature dimension), and the pooling is used to spatiotemporally align the features with $\boldsymbol{a}_L$. Specifically, consider

$$\hat{\boldsymbol{a}}_l = f(\textstyle\sum_{k=1}^{n_F^l} \boldsymbol{a}_l(k) * \mathbf{U}_{k,l}), \tag{4}$$

for $l = 1, \ldots, L-1$, and with $\hat{\boldsymbol{a}}_L = \boldsymbol{a}_L$. As discussed above, $\boldsymbol{a}_l(k)$ is the 3D feature map (tensor) for dictionary $k \in \{1, \ldots, n_F^l\}$ at layer l, and $\mathbf{U}_{k,l}$ is a 4D tensor. The convolution $*$ in (4) operates in the three shift dimensions, and $\boldsymbol{a}_l(k) * \mathbf{U}_{k,l}$ manifests a 4D tensor. Specifically, each of the n_F^L 3D "slices" of $\mathbf{U}_{k,l}$ are spatiotemporally convolved (3D convolution) with $\boldsymbol{a}_{k,l}$, and after summing over the n_F^l convolutional filters, followed by $f(\cdot)$, this manifests each of the n_F^L 3D slices of $\hat{\boldsymbol{a}}_l$. Function $f(\cdot)$ is an element-wise nonlinear activation function, followed by max pooling, with the pooling dimensions meant to realize final dimensions consistent with $\boldsymbol{a}_L$, $i.e.$, dimension $n_x^L \times n_y^L \times n_z^L \times n_F^L$. Consequently, $\hat{\boldsymbol{a}}_{i,l} \in \mathbb{R}^{n_F^L}$ is a feature vector with $i \in [1, \ldots, n_x^L] \times [1, \ldots, n_y^L] \times [1, \ldots, n_z^L]$. Note that it may only require one 3D tensor $\mathbf{U}_l$ applied on each 3D slices $\boldsymbol{a}_l(k)$ for each layer to achieve spatiotemporal alignment of the layer-dependent features. However, the features from two distinct layers will not be in the same "semantic" space, making it difficult to assess the value of the layer-dependent features. The multiple tensors in set $\{\mathbf{U}_{k,l}\}$ provide the desired semantic alignment between layers, allowing analysis of the value of features from different layers via a single MLP, in the following (5) and (6).

With $\{\hat{\boldsymbol{a}}_l\}_{l=1,L}$ semantically and spatiotemporally aligned, we now seek to jointly quantify the value of a particular spatiotemporal region and a particular feature layer ("abstraction") for prediction of the next word. For each $\hat{\boldsymbol{a}}_{i,l}$, the attention mechanism generates two positive weights, α_{ti} and β_{tl}, which measure the relative importance of location i and layer l for producing $\boldsymbol{y}_t$ based $\boldsymbol{y}_{<t}$. Attention weights α_{ti} and β_{tl} and context vector $\boldsymbol{z}_t$ are computed as

$$e_{ti} = \boldsymbol{w}_\alpha^T \tanh(\mathbf{W}_{a\alpha}\hat{\boldsymbol{a}}_i + \mathbf{W}_{h\alpha}\boldsymbol{h}_{t-1}), \quad \alpha_{ti} = \text{softmax}(\{e_{ti}\}), \quad \boldsymbol{s}_t = \psi(\{\hat{\boldsymbol{a}}_i\}, \{\alpha_{ti}\}), \tag{5}$$

$$b_{tl} = \boldsymbol{w}_\beta^T \tanh(\mathbf{W}_{s\beta}\boldsymbol{s}_{tl} + \mathbf{W}_{h\beta}\boldsymbol{h}_{t-1}), \quad \beta_{tl} = \text{softmax}(\{b_{tl}\}), \quad \boldsymbol{z}_t = \psi(\{\boldsymbol{s}_{tl}\}, \{\beta_{tl}\}), \tag{6}$$

where $\hat{\boldsymbol{a}}_i$ is a vector composed by stacking $\{\hat{\boldsymbol{a}}_{i,l}\}_{l=1,L}$ (all features at position i). e_{ti} and b_{tl} are scalars reflecting the importance of spatiotemporal region i and layer t to predicting $\boldsymbol{y}_t$, while α_{ti} and β_{tl} are *relative* weights of this importance, reflected by the softmax output. $\psi(\cdot)$ is a function that returns a single feature vector when given a set of feature vectors, and their corresponding weights across all i or l. Vector $\boldsymbol{s}_{tl}$ reflects the sub-portion of $\boldsymbol{s}_t$ associated with layer l.

In (5) we provide attention in the spatiotemporal dimensions, with that spatiotemporal attention shared across all L (now aligned) CNN layers. In (6) the attention is further refined, focusing attention in the layer dimension.

To make the following discussion concrete, we describe the attention function within the context of $\boldsymbol{z}_t = \psi(\{\boldsymbol{s}_{tl}\}, \{\beta_{tl}\})$. This function setup is applied in the same way to $\boldsymbol{s}_t = \psi(\{\hat{\boldsymbol{a}}_i\}, \{\alpha_{ti}\})$.

Soft attention Following Bahdanau et al. (2015), we formulate the soft attention model by computing a weighted sum of the input features

$$\boldsymbol{z}_t = \psi(\{\boldsymbol{s}_{tl}\}, \{\beta_{tl}\}) = \textstyle\sum_{l=1}^{L} \beta_{tl} \boldsymbol{s}_{tl}. \tag{7}$$

The model is differentiable for all parameters and can be learned end-to-end using standard back-propagation.

Hard attention Let $\boldsymbol{m}_t \in \{0, 1\}^L$ be a vector of all zeros, and a single one, and the location of the non-zero element of $\boldsymbol{m}_t$ identifies the location to extract features for generating the next word. We impose

$$\boldsymbol{m}_t \sim \mathsf{Mult}(1, \{\beta_{tl}\}), \qquad \boldsymbol{z}_t = \textstyle\sum_{l=1}^{L} m_{tl} \boldsymbol{s}_{tl}. \tag{8}$$

In this case, optimizing the objective function in (3) is intractable. However, the marginal log-likelihood can be lower-bounded as

$$\log p(\mathbf{Y}|\mathbf{A}) = \log \textstyle\sum_{\boldsymbol{m}} p(\boldsymbol{m}|\mathbf{A}) p(\mathbf{Y}|\boldsymbol{m}, \mathbf{A}) \geq \sum_{\boldsymbol{m}} p(\boldsymbol{m}|\mathbf{A}) \log p(\mathbf{Y}|\boldsymbol{m}, \mathbf{A}), \tag{9}$$

where $\boldsymbol{m} = \{\boldsymbol{m}_t\}_{t=1,\dots,T}$. Inspired by importance sampling, the multi-sample stochastic lower bound has been recently used for latent variable models (Ba et al., 2015; Burda et al., 2016), defined as

$$\mathcal{L}^K(\mathbf{Y}) = \textstyle\sum_{\boldsymbol{m}^{1:K}} p(\boldsymbol{m}^{1:K}|\mathbf{A}) \Big[\log \frac{1}{K} \sum_{k=1}^{K} p(\mathbf{Y}|\boldsymbol{m}^k, \mathbf{A}) \Big], \tag{10}$$

where $\boldsymbol{m}^1, \dots, \boldsymbol{m}^K$ are independent samples. This lower bound is guaranteed to be tighter with the increase of the number of samples K (Burda et al., 2016), thus providing a better approximation of the objective function than (9). As shown in Mnih & Rezende (2016), the gradient of $\mathcal{L}^K(\mathbf{Y})$ with respect to the model parameters is

$$\nabla \mathcal{L}^K(\mathbf{Y}) = \textstyle\sum_{\boldsymbol{m}^{1:K}} p(\boldsymbol{m}^{1:K}|\mathbf{A}) \sum_{k=1}^{K} \Big[L(\boldsymbol{m}^{1:K}) \nabla \log p(\boldsymbol{m}^k|\mathbf{A}) + \omega_k \nabla p(\mathbf{Y}|\boldsymbol{m}^k, \mathbf{A}) \Big], \tag{11}$$

where $L(\boldsymbol{m}^{1:K}) = \log \frac{1}{K} \sum_{k=1}^{K} p(\mathbf{Y}|\boldsymbol{m}^k, \mathbf{A})$ and $\omega_k = \frac{p(\mathbf{Y}|\boldsymbol{m}^k, \mathbf{A})}{\sum_j p(\mathbf{Y}|\boldsymbol{m}^j, \mathbf{A})}$. A variance reduction technique is introduced in Mnih & Rezende (2016) by replacing the above gradient with an unbiased estimator

$$\nabla \mathcal{L}^K(\mathbf{Y}) \approx \textstyle\sum_{k=1}^{K} \Big[\hat{L}(\boldsymbol{m}^k|\boldsymbol{m}^{-k}) \nabla \log p(\boldsymbol{m}^k|\mathbf{A}) + \omega_k \nabla p(\mathbf{Y}|\boldsymbol{m}^k, \mathbf{A}) \Big], \tag{12}$$

where

$$\hat{L}(\boldsymbol{m}^k|\boldsymbol{m}^{-k}) = L(\boldsymbol{m}^{1:K}) - \log \tfrac{1}{K} \big(\textstyle\sum_{j \neq k} p(\mathbf{Y}|\boldsymbol{m}^j, \mathbf{A}) + f(\mathbf{Y}, \boldsymbol{m}^{-k}, \mathbf{A}) \big) \big), \tag{13}$$

$$f(\mathbf{Y}, \boldsymbol{m}^{-k}, \mathbf{A}) = \exp(\tfrac{1}{K-1} \textstyle\sum_{j \neq k} \log p(\mathbf{Y}|\boldsymbol{m}^j, \mathbf{A}). \tag{14}$$

When learning the model parameters, the lower bound (10) is optimized via the gradient approximation in (12).

As an alternative method, one may first produce abstraction-level attention weights β_l, followed by generating spatiotemporal attention weights α_i, *i.e.*, switching the order of (5) and (6). However, we empirically found that this method provides slightly worse performance (we implemented and tested both) in our experiments, partially due to the increase of the number of parameters.

4 Experiments

4.1 Datasets

We present results on three benchmark datasets: Microsoft Research Video Description Corpus (YouTube2Text) (Chen & Dolan, 2011), Montreal Video Annotation Dataset (M-VAD) (Torabi et al., 2015), and Microsoft Research Video to Text (MSR-VTT) (Xu et al., 2016).

The Youtube2Text contains 1970 Youtube clips, and each video is annotated with around 40 sentences. For fair comparison, we used the same splits as provided in Venugopalan et al. (2015b), with 1200 videos for training, 100 videos for validation, and 670 videos for testing.

The M-VAD is a large-scale movie description dataset, which is composed of 46587 movie snippets annotated with 56752 sentences. We follow the setting in Torabi et al. (2015), taking 36920 videos for training, 4950 videos for validation, and 4717 videos for testing.

The MSR-VTT is a newly collected large-scale video dataset, consisting of 20 video categories. The dataset was split into 6513, 2990 and 497 clips in the training, testing and validation sets. Each video has about 20-sentence descriptions. The ground-truth captions in the testing set are not available now. Thus, we split the original training dataset into a training set of 5513 clips and a testing set of 1000 clips.

We convert all captions to lower case and remove the punctuation, yielding vocabulary sizes $V = 12594$ for Youtube2Text, $V = 13276$ for M-VAD, and $V = 8071$ for MSR-VTT.

4.2 TRAINING PROCEDURE

We consider the RGB frames of videos as input, and all videos are resized to 112×112 spatially, with 2 frames per second; note that the temporal sample rate of the videos are consequently consistent, but not necessarily the total number of frames. The C3D (Tran et al., 2015) is pretrained on Sports-1M dataset Karpathy et al. (2014), consisting of 1.1 million sports videos belonging to 487 categories. We extract the features from four convolutional layers and one fully connected layer, named as *pool2, pool3, pool4, pool5* and *fc-7* in the C3D (Tran et al., 2015), respectively. The model architecture of C3D is provided in Appendix B. The kernel sizes of the convolutional transformation in (4) are $7 \times 7 \times 7$, $5 \times 5 \times 5$ and $3 \times 3 \times 3$ for layer *pool2, pool3* and *pool4* with $3 \times 3 \times 3$, $2 \times 2 \times 2$ and $1 \times 1 \times 1$ zero padding, respectively. $f(\cdot)$ is implemented by ReLU (Nair & Hinton, 2010), followed by 3D max-pooling with $8 \times 8 \times 8$, $4 \times 4 \times 4$ and $2 \times 2 \times 2$ ratios. More details for achieving spatiotemporal alignment are provided in Appendix C .

All recurrent matrices in the LSTM are initialized with orthogonal initialization (Saxe et al., 2014). We initialize non-recurrent weights from a uniform distribution in $[-0.01, 0.01]$ and all the bias terms are initialized to zero. Word embedding vectors are initialized with the publicly available *word2vec* vectors that were trained on 100 billion words from Google News, which have dimensionality 300, and were trained using a continuous bag-of-words architecture (Mikolov et al., 2013). The embedding vectors of words not present in the pre-trained set are initialized randomly. The number of hidden units in the LSTM is set as 512 and we use mini-batches of size 32. Gradients are clipped if the norm of the parameter vector exceeds 5 (Sutskever et al., 2014). The number of samples for multi-sample stochastic lower bound is set to 10. We do not perform any dataset-specific tuning and regularization other than dropout (Srivastava et al., 2014) and early stopping on validation sets. The Adam algorithm Kingma & Ba (2014) with learning rate 0.0002 is utilized for optimization. All experiments are implemented in Torch (Collobert et al., 2011).

4.3 EVALUATION

The widely used BLEU (Papineni et al., 2002), METEOR (Banerjee & Lavie, 2005) and CIDEr (Vedantam et al., 2015) metrics are employed to quantitatively evaluate the performance of our video caption generation model, and other models in the literature. For each dataset, we show three types of results, using part of or all of our model to illustrate the role of each component:

1. *C3D fc7 + LSTM* : The LSTM caption decoder is employed, only using features extracted from the top fully-connected layer. No context vector $\boldsymbol{z}_t$ is generated from intermediate convolutional layer features.
2. *Spatiotemporal attention + LSTM*: The context vector $\boldsymbol{z}_t$ is included, but only features extracted from a certain convolutional layer are employed, *i.e.*, $\boldsymbol{z}_t$ is equal to $\boldsymbol{s}_t$ in (5). The spatiotemporal attention is implemented with the soft attention in (7). For example, "C3D fc7 + pool2" means we leverage the features from layer *pool2* to generate attention weights and context vector.

Table 1: Results on BLEU-4, METEOR and CIDEr metrics compared to other state-of-the-art results and baselines on Youtube2Text and M-VAD datasets. [1,2,3,4] represent Venugopalan et al. (2015a); Pan et al. (2016b); Ballas et al. (2016); Yu et al. (2016), respectively.

Methods	Youtube2Text			M-VAD		
	BLEU-4	METEOR	CIDEr	BLEU-4	METEOR	CIDEr
S2VT (VGG) [1]	-	29.2	-	-	5.6	-
LSTM-E (VGG + C3D) [2]	45.3	31.0	-	-	6.7	-
GRU-RCN [3]	47.90	31.14	67.82	-	-	-
h-RNN (C3D+VGG) [4]	49.9	32.6	65.8	-	-	-
Baselines						
ResNet + LSTM	44.08	30.99	66.88	0.81	6.22	5.54
C3D fc7 + LSTM	45.34	31.21	66.12	0.83	6.31	5.96
Spatiotemporal attention + LSTM						
C3D fc7 + pool2	45.46	31.53	67.38	0.98	6.42	6.01
C3D fc7 + pool3	48.07	33.52	69.97	1.12	6.71	6.97
C3D fc7 + pool4	48.18	34.47	70.97	1.24	6.89	7.12
C3D fc7 + pool5	47.75	33.35	69.71	1.02	6.49	6.48
Baselines of multi-level features with attention						
MLP + soft attention	35.99	23.56	44.21			
Max-pooling + soft attention	45.35	31.50	62.99			
Average-pooling + soft attention	48.53	33.59	65.07			
Hypercolumn + soft attention	44.92	30.18	63.18			
ASTAR						
Soft Attention	**51.74**	**36.39**	**72.18**	**1.94**	**7.72**	7.98
Hard Attention	51.64	34.99	72.08	1.82	7.12	**8.12**

3. *ASTAR*: This is our proposed model developed in Sec. 3.2. We present results based on both soft attention and hard attention.

To compare our method with other multi-level feature methods, we show several baseline results on Youtube2Text dataset:

1. *MLP* : Each $\boldsymbol{a}_l$ is feed to a different MLP to achieve the same dimension of $\boldsymbol{a}_L$, *i.e.* $\hat{\boldsymbol{a}}_l = \mathsf{MLP}(\mathsf{vec}(\boldsymbol{a}_l))$ for $l = 1, \ldots, L-1$, where $\mathsf{vec}(\cdot)$ denotes vectorization. The context vector $\boldsymbol{z}_t$ is obtained by abstraction-level and spatiotemporal attention.
2. *Max/Average-pooling*: A max or average pooling operation is utilized to achieve saptiotemporal alignment, and an MLP is then employed to embed the feature vectors into the same semantic space, *i.e.* for each $\hat{\boldsymbol{a}}_l$ with $l = 1, \ldots, L$:

$$\tilde{\boldsymbol{a}}_{i,l}(k) = \max_{j \in \mathcal{N}_{i,l}} \boldsymbol{a}_{j,l}(k) \text{ or } \frac{1}{|\mathcal{N}_{i,l}|} \sum_{j \in \mathcal{N}_{i,l}} \boldsymbol{a}_{i,l}(k) \qquad \text{for } k = 1, \ldots, n_F^l \tag{15}$$

$$\hat{\boldsymbol{a}}_{i,l} = \mathsf{MLP}(\tilde{\boldsymbol{a}}_{i,l}) \tag{16}$$

   where $\mathcal{N}_{i,l}$ is the receptive field (see Appendix C for the definition) of $\boldsymbol{a}_{i,L}$ in the $l-$th layer. Similarly, the context vector $\boldsymbol{z}_t$ is computed by abstraction-level and spatiotemporal attention.
3. *Hypercolumn*: This method is similar with max/average-pooling but replace the pooling operation with hypercolumn representation. All the features are concatenated into one long vector for every location and only a saptiotemporal attention is employed to generate the context vector.

In these methods, the attention weights are produced by the soft attention in (7).

To verify the effectiveness of our video caption generation model and C3D features, we also implement a strong baseline method based on the LSTM encoder-decoder network (Cho et al., 2014), where ResNet He et al. (2016) is employed as the feature extractor on each frame. We denote results using this method as *ResNet + LSTM* .

4.4 QUANTITATIVE RESULTS

Results are summarized in Tables 1 and 2, and we obtain state-of-the-art results on both Youtube2Text and M-VAD datasets. On the MSR-VTT dataset, our results also significantly outperform the strong baselines, demonstrating the effectiveness of the proposed model. Note that no comparative results exist in the literature for MSR-VTT. As a reference, with similar settings, Xu et al. (2016) using *C3D fc7 + LSTM* achieved 39.9 on BLEU-4 and 29.3 on METEOR using the standard testing set, while state-of-the-art results are 40.8 on BLEU-4, 28.2 on METEOR, and 44.8 on CIDEr, achieved by team "v2t_navigator" from RUC and CMU [1].

Table 2: Results on BLEU-4, METEOR and CIDEr metrics compared to baselines on MSR-VTT.

Method	BLEU-4	METEOR	CIDEr
Baselines			
ResNet + LSTM	39.54	26.59	45.22
C3D fc7 + LSTM	40.17	26.86	45.95
Spatiotemporal attention + LSTM			
C3D fc7 + pool2	40.43	26.93	47.15
C3D fc7 + pool3	42.04	27.18	48.93
C3D fc7 + pool4	41.98	27.42	48.21
C3D fc7 + pool5	40.83	27.01	47.86
ASTAR			
soft attention	43.72	**29.67**	50.21
hard attention	**43.89**	28.71	**50.29**

It is worth noting that our model consistently yields significant improvements over models with only spatiotemporal attention, which further achieve better performance than only taking the C3D top fully-connected layer features; this demonstrates the importance of leveraging intermediate convolutional layer features. In addition, our model outperforms all the baseline results of multi-level feature based method, demonstrating the effectiveness of our 3D convolutional transformation operation. It is partly because the sparse connectivity of convolutional operation, which indicates fewer parameters are required. Furthermore, We achieve these results using a single model, without averaging over an ensemble of such models.

4.5 QUALITATIVE RESULTS

Following Xu et al. (2015), we visualize the attention components learned by our model on Youtube2Text in Figure 3. As can be seen from Figure 3, the spatiotemporal attention aligns the objects in the video well with the corresponding words. In addition, the abstraction-level attention tends to focus on low level features when the model will generate a noun and high level features when an article or a preposition is to be generated. More results are provided in Appendix A.1.

Examples of generated captions from unseen videos on Youtube2Text are shown in Figure 4. We find the results with abstraction-layer attention (indicated as "soft attention" or " hard attention") is generally equal to or better than the best results, compared to those only taking a certain convolutional-layer feature (indicated as "Pool2" *etc.*) . It demonstrates the effectiveness of our abstraction layer attention. More results are provided in Appendix A.2 and http://www.cs.toronto.edu/pub/cuty/videocaption/.

5 CONCLUSION AND FUTURE WORK

We have proposed a novel video captioning model, that adaptively selects/weights the feature abstraction (CNN layer), as well as the location within a layer-dependent feature map. We have implemented the attention using both "hard" and "soft" mechanisms, with the latter typically delivering better performance. Our model achieves excellent video caption generation performance, and has the capacity to provide interpretable alignments seemingly analogous to human perception.

We have focused on analysis of videos and associated captions. Similar ideas may be applied in the future to image captioning. Additionally, the CNN parameters were learned separately as a first step, prior to analysis of the captions. It is also of interest to consider CNN-parameter refinement conditioned on observed training captions.

[1] http://ms-multimedia-challenge.com/leaderboard

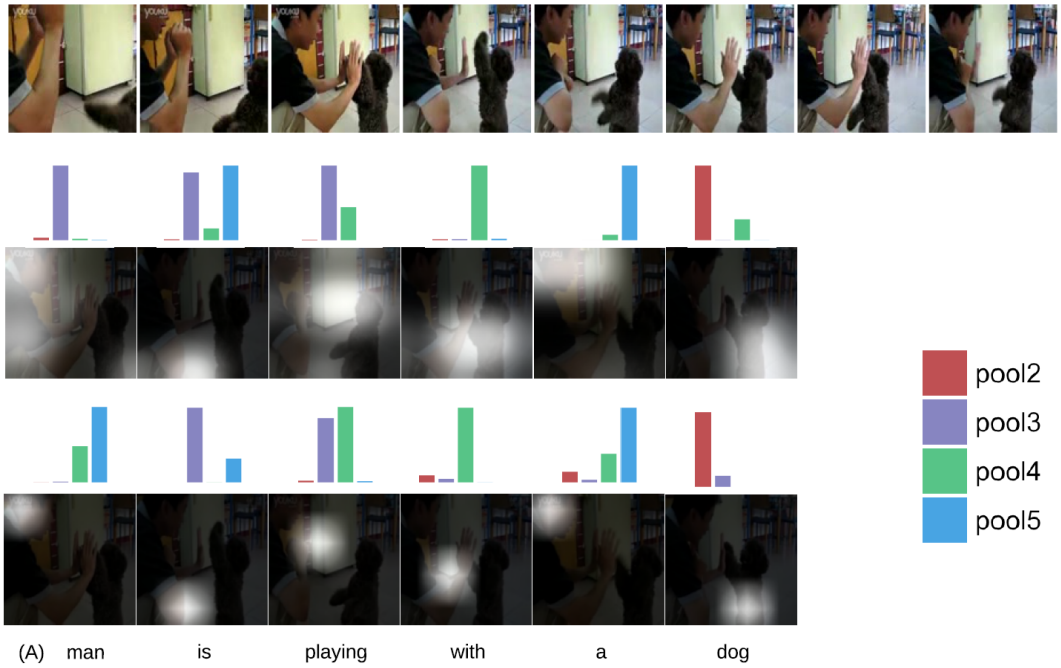
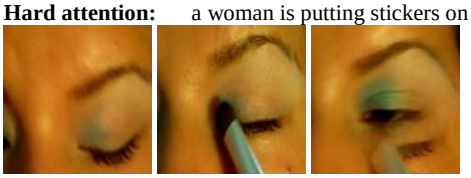


Figure 3: Visualization of attention weights aligned with input video and generated caption on Youtube2Text: (top) Sampled frames of input video, (middle) soft attention, (bottom) hard attention. We show the frame with the strongest attention weights. The bar plot above each frame corresponds layer attention weights β_{tl} when the corresponding word (under the frame) is generated.

Pool2: a baby is walking
Pool3: a bear is walking
Pool4: a bear is walking
Pool5: a badger is walking
Soft attention: a turtle is walking
Hard attention: a turtle is walking

Pool2: a girl is smiling
Pool3: a girl is talking
Pool4: a woman is licking a face
Pool5: a girl is putting stickers on her face
Soft attention: a woman is putting stickers on her face
Hard attention: a woman is putting stickers on her face

Pool2: a group of men are playing
Pool3: a group of men are playing
Pool4: a girl is doing gymnastics
Pool5: a gymnast falls down
Soft attention: a woman is doing gymnastics
Hard attention: a girl is doing gymnastics



Pool2: a woman is peeling a toad
Pool3: a woman is applying eye makeup
Pool4: a woman is licking a brush
Pool5: a woman is plucking her face
Soft attention: a woman is applying eye shadow
Hard attention: a woman is applying eye makeup

Figure 4: Examples of generated captions with sampled frames of input video on YouTube2Text.

REFERENCES

J. Ba, R. Grosse, R. Salakhutdinov, and B. Frey. Learning wake-sleep recurrent attention models. In *NIPS*, 2015.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. In *ICLR*, 2016.

S. Banerjee and A. Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL workshop*, 2005.

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. In *ICLR*, 2016.

D. Chen and W. B. Dolan. Collecting highly parallel data for paraphrase evaluation. In *ACL*, 2011.

K. Cho, B. V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

Z. Gan, C. Gan, X. He, Y. Pu, K. Tran, J. Gao, L. Carin, and L. Deng. Semantic compositional networks for visual captioning. In *ArXiv*, 2016.

S. Guadarrama, N. Krishnamoorthy, G. Malkarnenkar, S. Venugopalan, R. Mooney, T. Darrell, and K. Saenko. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *ICCV*, 2013.

B. Hariharan, . Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

K. He, X. Zhang, S. Ren, and Sun J. Deep residual learning for image recognition. In *CVPR*, 2016.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

A. Karpathy and F. Li. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

R. Kiros, R. Salakhutdinov, and R. Zemel. Multimodal neural language models. In *ICML*, 2014.

A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.

J. Mao, W. Xu, Y. Yang, J. Wang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-RNN). In *ICLR*, 2015.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

A. Mnih and D. J. Rezende. Variational inference for monte carlo objectives. In *ICML*, 2016.

V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. In *NIPS*, 2014.

V. Nair and G. E. Hinton. Rectied linear units improve restricted boltzmann machines. In *ICML*, 2010.

P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. In *CVPR*, 2016a.

Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui. Jointly modeling embedding and translation to bridge video and language. In *CVPR*, 2016b.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. Bleu: a method for automatic evaluation of machine translation. *Transactions of the Association for Computational Linguistics*, 2002.

Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational autoencoder for deep learning of images, labels and captions. In *NIPS*, 2016.

A. Rohrbach, M. Rohrbach, W. Qiu, A. Friedrich, M. Pinkal, and B. Schiele. Coherent multi-sentence video description with variable level of detail. In *GCPR*. 2014.

M. Rohrbach, W. Qiu, I. Titov, S. Thater, M. Pinkal, and B. Schiele. Translating video content to natural language descriptions. In *ICCV*, 2013.

A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*, 2014.

P. Sermanet, K. Kavukcuoglu, and Y. Chintala, S.and LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *CVOR*, 2013.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

A Torabi, H Larochelle C Pal, and A Courville. Using descriptive video services to create a large data source for video annotation research. In *arXiv preprint arXiv:1503.01070*, 2015.

D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

R. Vedantam, Z. C. Lawrence, and D. Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, 2015.

S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence-video to text. In *ICCV*, 2015a.

S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. In *NAACL*, 2015b.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

J. Xu, T. Mei, T. Yao, and Y. Rui. Msr-vtt: A large video description dataset for bridging video and language. In *CVPR*, 2016.

K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

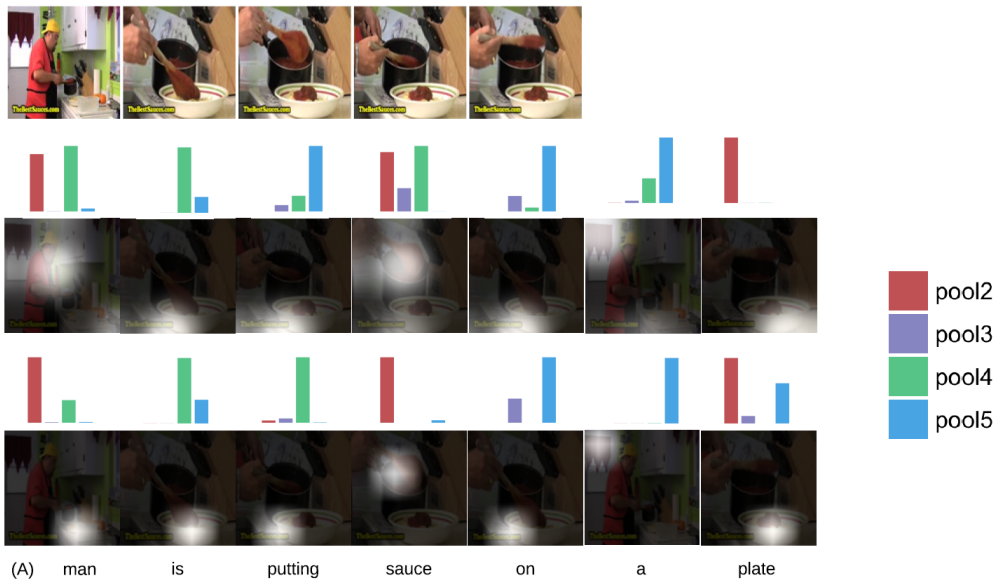
L Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In *ICCV*, 2015.

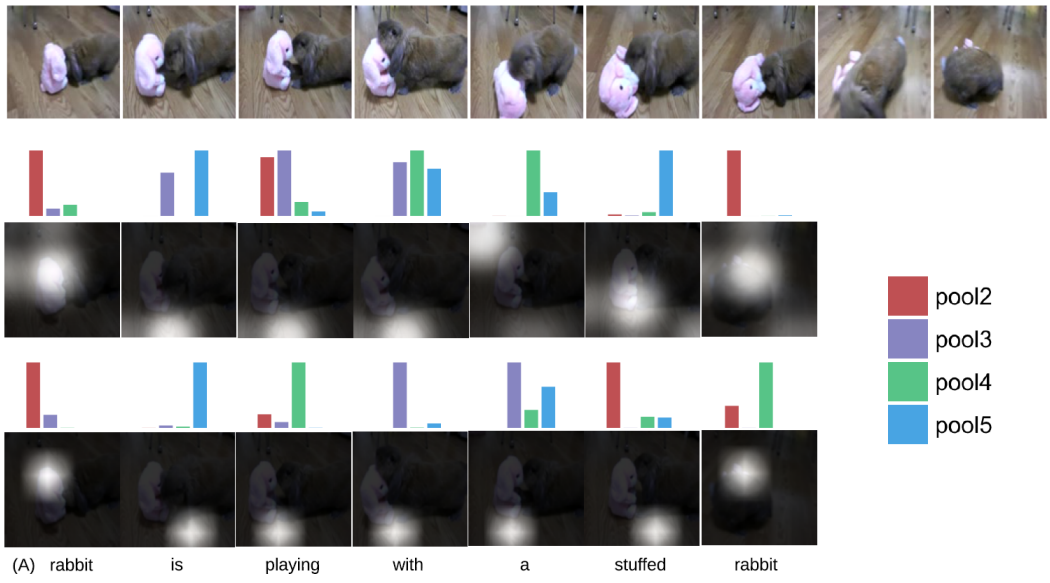
H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *CVPR*, 2016.

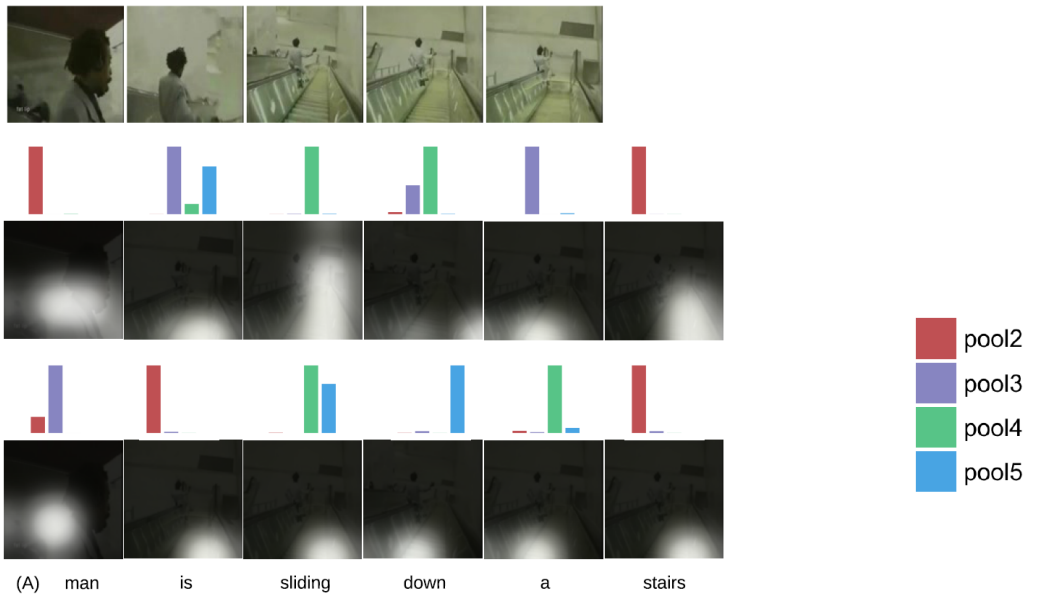
M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

A MORE RESULTS

A.1 VISUALIZATION OF ATTENTION WEIGHTS






pool2
pool3
pool4
pool5
(A) man is sliding down a stairs
(Two) men are loading luggage into a car

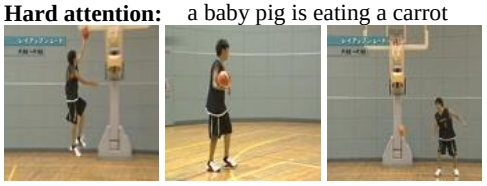
A.2 GENERATED CAPTIONS

Pool2: a man is smiling
Pool3: a man is eating a banana
Pool4: a man is talking
Pool5: a man is talking
Soft attention: a man is eating a banana
Hard attention: a man is eating a banana

Pool2: a monkey is running
Pool3: a monkey is doing martial arts
Pool4: a monkey is fighting
Pool5: a monkey is fighting
Soft attention: a monkey is doing martial arts
Hard attention: a monkey is doing martial arts

Pool2: a puppy is playing
Pool3: a puppy is playing with a toy
Pool4: a puppy is playing
Pool5: a baby panda is playing
Soft attention: a pig is eating a carrot
Hard attention: a baby pig is eating a carrot

Pool2: a man is lifting weights
Pool3: a man is sharpening a knife
Pool4: a man is doing something
Pool5: a man is lifting weights
Soft attention: a man is putting a knife in a vice
Hard attention: a man is sharpening a knife



Pool2: a man is kicking a basketball
Pool3: a boy is hitting a basketball
Pool4: a man is hitting a basketball
Pool5: a man is dribbling a basketball
Soft attention: a man is dribbling a basketball
Hard attention: a man is dribbling a basketball

Pool2: a man is dancing
Pool3: a man is performing on stage
Pool4: a man is playing a guitar
Pool5: a man is playing a guitar
Soft attention: a band is performing on a stage
Hard attention: a band is playing a guitar

Pool2: a person is cutting a vegetable
Pool3: a person is cutting a vegetable
Pool4: a person is cutting a vegetable
Pool5: a man is cutting a vegetable
Soft attention: a woman is slicing a vegetable
Hard attention: a woman is slicing a vegetable

Pool2: a man is talking
Pool3: a man is talking
Pool4: a man is singing
Pool5: a man is singing
Soft attention: a man and woman are talking
Hard attention: a man and woman are sitting on a motorcycle

B MODEL ARCHITECTURE OF C3D

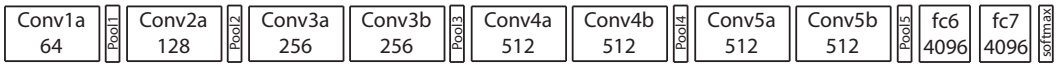


Figure 5: C3D net is composed of 8 3D convolution layers, 5 3D max-pooling layers, 2 fully connected layers, and a softmax output layer. All 3D convolution kernels are $3 \times 3 \times 3$ with $1 \times 1 \times 1$ padding and stride 1 in both spatial and temporal dimensions. Number of filters are denoted in each box. The 3D max-pooling layers are named from *pool1* to *pool5*. All pooling ratios are $2 \times 2 \times 2$, except for *pool1* is $1 \times 2 \times 2$ (1 is in the temporal dimension). Each fully connected layer has 4096 output units. (Tran et al., 2015)

C DIMENSIONS AND RECEPTIVE FIELD OF C3D FEATURES

The dimensions for features extracted from *pool2, pool3, pool4* and *pool5* are $28 \times 28 \times N/2 \times 128$, $14 \times 14 \times N/4 \times 256$, $7 \times 7 \times N/8 \times 512$ and $4 \times 4 \times N/16 \times 512$, respectively. N is the number of frames of input video. After the convolutional transformation, the dimensions will be all $4 \times 4 \times N/16 \times 512$.

To prove these features are spatiotemporal aligned, we first provide the receptive field for 3D convolutional layer and 3D pooling layer. Let $\mathbf{Y} = \text{3D-Conv}(\mathbf{X})$, where 3D-Conv is the 3D convolutional layer with kernel size $3 \times 3 \times 3$. The features indexed by $i = [i_x, i_y, i_z]$ in $\mathbf{Y}$ is obtained by convolving a subset of $\mathbf{X}$ indexed by $j = [j_x, j_y, j_z]$ with convolutional kernel, where $j_x \in [i_x - 1, i_x, i_x + 1]$, $j_y \in [i_y - 1, i_y, i_y + 1]$ and $j_z \in [i_z - 1, i_z, i_z + 1]$. Then, we call that the receptive field of $i = [i_x, i_y, i_z]$ in $\mathbf{Y}$ is $[i_x - 1, \ldots, i_x + 1] \times [i_y - 1, \ldots, i_y + 1] \times [i_z - 1, \ldots, i_z + 1]$ in $\mathbf{X}$. Similarly, if $\mathbf{Y} = \text{3D-pooling}(\mathbf{X})$ with pooling ratio $2 \times 2 \times 2$, the receptive field of $i = [i_x, i_y, i_z]$ in $\mathbf{Y}$ is $[2i_x - 1, 2i_x] \times [2i_y - 1, 2i_y] \times [2i_z - 1, 2i_z]$ in $\mathbf{X}$.

We then provide the receptive field of features $\boldsymbol{a}_l$ from each layer in the input video in Table 3 and receptive field of features after convolutional transformation, $\hat{\boldsymbol{a}}_l$, in the original feature $\boldsymbol{a}_l$ in Tabel 4. The features are all indexed by $i = [i_x, i_y, i_z]$. Combining Table 3 and Tabel 4, we can find the receptive field of $\hat{\boldsymbol{a}}_l$ indexed by $i = [i_x, i_y, i_z]$ for all l in the input video are all $[32i_x - 63, \ldots, 32i_x + 30] \times [32i_y - 63, \ldots, 32i_y + 30] \times [16i_z - 32, \ldots, 16i_z + 15]$.

We index the top-left element in the first frame as $[1, 1, 1]$. Note that the index of receptive field could be negative due to padding.

Table 3: Receptive field of $\boldsymbol{a}_l$ in input video

Layer name	Receptive field
Pool2	$[4i_x - 7, \ldots, 4i_x + 2] \times [4i_y - 7, \ldots, 4i_y + 2] \times [2i_z - 4, \ldots, 2i_z + 1]$
Pool3	$[8i_x - 15, \ldots, 8i_x + 6] \times [8i_y - 15, \ldots, 8i_y + 6] \times [4i_z - 8, \ldots, 4i_z + 3]$
Pool4	$[16i_x - 31, \ldots, 16i_x + 14] \times [16i_y - 31, \ldots, 16i_y + 14] \times [8i_z - 16, \ldots, 8i_z + 7]$
Pool5	$[32i_x - 63, \ldots, 32i_x + 30] \times [32i_y - 63, \ldots, 32i_y + 30] \times [16i_z - 32, \ldots, 16i_z + 15]$

Table 4: Receptive field of $\hat{\boldsymbol{a}}_l$ in the corresponding $\boldsymbol{a}_l$

Layer name	Receptive field
Pool2	$[8i_x - 14, \ldots, 8i_x + 7] \times [8i_y - 14, \ldots, 8i_y + 7] \times [8i_z - 14, \ldots, 8i_z + 7]$
Pool3	$[4i_x - 6, \ldots, 4i_x + 4] \times [4i_y - 6, \ldots, 4i_y + 4] \times [4i_z - 6, \ldots, 4i_z + 4]$
Pool4	$[2i_x - 2, \ldots, 2i_x + 1] \times [2i_y - 2, \ldots, 2i_y + 1] \times [2i_z - 2, \ldots, 2i_z + 1]$