A Convolutional Temporal Encoder for Video Caption Generation

Qingle Huang and Zicheng Liao

Institute of Artificial Intelligence
Computer Science, Zhejiang University

Abstract

We propose a convolutional temporal encoding network for video sequence embedding and caption generation. The mainstream video captioning work is based on recurrent encoder of various forms (e.g. LSTMs and hierarchical encoders). In this work, a multi-layer convolutional neural network encoder is proposed. At the core of this encoder is a gated linear unit (GLU) that performs a linear convolutional transformation of input with a nonlinear gating, which has demonstrated superior performance in natural language modeling. Our model is built on top of this unit for video encoding and integrates several up-to-date tricks including batch normalization, skip connection and soft attention. Experiment on two large-scale benchmark datasets (MSAD and M-VAD) generates strong results and demonstrates the effectiveness of our model.

1 Introduction

The problem of video captioning has been drawing increasing attention, not only for the lack of text labels for the vast amount of video data known as the "dark matter" of the Internet, but also for its intersection with two significant domains and the easiness to be modeled as a sequence-to-sequence translation problem. Most existing work takes the approach of a sequence encoder followed by a decoder. The decoder is unanimously some form of recurrent network that predicts one word at a time conditioned on previous state, and the encoder differs in the form of pooling over frame features (e.g. [27]), or standard recurrent neural network transformation (e.g. [18, 28, 29]), or deep recurrent neural nets that exploit hierarchical structure of a temporal sequence (e.g. [1, 17]). In this work we propose a new video sequence encoding scheme for the task for caption generation.

Specifically, our approach sets apart from existing work in that we use a feed-forward *convolutional* neural network for the sequence encoding. The input video clip is first represented as a sequence of static CNN features, a 10-layer gated convolutional neural network is then applied over the temporal domain of the sequence, transforming it into a hidden state for a recurrent network (the decoder) to generate the target sentence. Figure 1 (bottom path) shows our video captioning architecture and comparison to previous approaches.

There is good reason for the use of a recurrent neural network decoder for sequence generation: a recurrent network models the conditional distribution $P(\mathbf{v}|\mathbf{z}) = P(v_1,...,v_n|\mathbf{z})$ without having to make independence assumptions $P(v_1,...,v_n|\mathbf{z}) = \Pi_i P(v_i|v_{i-1},...,v_1,\mathbf{z})$, which cannot be expressively represented with a feed forward network. However, the same recurrent network might have been overly used for the problem of sequence *encoding*. The

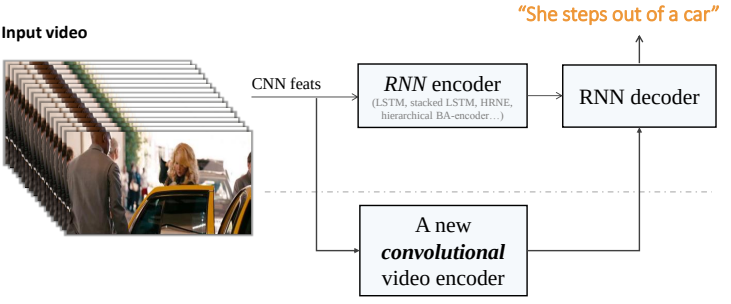


Figure 1: Existing video-to-text translation models unanimously take a recurrent neural encoder-decoder framework. In this work we propose a convolution video encoder that performs as a promising alternative to counterpart recurrent video encoders.

role of the encoder is to embed the input into a continuous space that captures large context or long-range dependencies, for which a feed forward convolutional network has been very successful in visual encoding for image classification and other tasks. In NLP community, there is also recent work on convolutional encoding for language modeling that achieved strong performance over alternative recurrent encoders (e.g. [2, 6]). Our work takes inspiration from these work and extends it to video sequence encoding for caption generation.

The approach of a convolutional neural network encoding offers a number of advantages. First, RNNs are a powerful tool for modeling sequential data, but the dependence of each timestep on the previous timestep's output limits the space for parallelization. Second, modeling long sequences with recurrent network is intrinsically difficult – gradient backprop is numerically instable with large timesteps, and information from various timesteps are inseparably aggregated to affect current state. In comparison, CNNs enjoy both parallelism and scalability to the input size. From the other view, despite the differences, CNNs and RNNs bear the same spirit of parameter sharing scheme: Unfolded RNNs are multi-layer neural network with shared parameters across timesteps. And CNNs are kernel convolution of fixed parameters over spatial domain. In this work, we apply such convolutional operator to the temporal domain of video clips with a 10-layer network and achieve favorable results against strong recurrent counterparts.

The main contribution of this work is the exploration of the convolutional GLU module in the domain of video captioning, where existing work unanimously used recurrent encoders. The use of GLU allows the temporal input to be encoded in a whole different way, and it runs about 5× faster. We tested the model on two standard video captioning benchmark datasets, the MSVD [3] and M-VAD [24] dataset. Our model achieves METEOR score of 33.1 and 7.11 on the two datasets respectively, comparing to 33.1 and 6.80 by the HRNE model [17] and 32.4 and 7.3 by a hierarchical boundary-aware encoder [1] in controlled configurations, suggesting it as a promising alternative to the widely used recurrent encoders.

2 Related Work

Recurrent temporal encoding for video captioning: The task of video captioning is a sequence to sequence translation problem, which is typically formulated as a feature encoding module followed by a recurrent decoding network. Most work differs from the way the

video sequence is encoded. The very early work of Venugopalan et al. [27] uses a simple average pooling of all frames CNN features, and inputs it to a recurrent neural network to predict words. A later work by Venugopalan et al. [28] encodes the input video frames with a same recurrent neural network (e.g. LSTM) as the decoder network. During encoding, per-frame CNN features (VGG, GoogLeNet, etc.) are supplied as input to an RNN in the manner of machine translation. Later on, Yao et al. [29] use 3x3x3 spatiotemporal convolutional features and Pan et al. [18] use more sophisticated C3D features [25], both as input to a a recurrent network for sequence encoding.

Stacked recurrent nets and Hierarchical encoding: A natural extension for sequential encoding is to add hierarchical abstraction over the temporal domain. Graves et al. [10] introduce a deep, or stacked, recurrent neural network to learn long range contexts for speech recognition. Ng et al. [16] employ a five-layer stacked LSTM encoder for video classification. Hierarchical recurrent network structure has also been a popular choice for video captionings. Rather than the simple stacked RNN structure, the HRNE model proposed by Pan et al. [17] forms a hierarchical RNN structure that uncovers transitions with different granularity and reduces the length of information flow in the network. The recent boundary-aware video encoder [1] propose an alternative LSTM cell that can identify discontinuities between video frames or segments to form a hierarchical recurrent network. The work of Yu et al. [30] also generates texts from video clips. It builds a hierarchical recurrent network in the decoder network, which consists of a sentence generator and a paragraph generator, the latter is built on top of the output of the former. Our work focuses on video encoding and uses a standard LSTM decoder as most relevant work does. Our multi-layer encoding network shares similar spirit with that of the HRNE model [17] and the boundary aware encoder [1], but it is a stack of parallelable convolutional layers over the input sequence.

Temporal encoding with Convolution: There is a few early work implementing the idea of temporal encoding with convolutional networks in the natural language processing community. Dauphin et al. [6] are the first to introduce a feed-forward convolutional network to estimate word probability distribution, which outperforms traditional recurrent networks (i.e. LSTM) in terms of performance and speed, and demonstrates the effectiveness of a non-recurrent approach in sequence modeling. Similarly, Bradbury et al. [2] introduce a quasi-RNN model, which eliminates major sequential dependencies with convolution operator, and produces favorable results in language modeling tasks over stacked LSTMs. Semeniuta et al. [21] introduce a hybrid convolutional-recurrent VAE model for text generation. Our model resembles this model in that we also hybrids a convolutional encoder and a recurrent decoder. In computer vision, the C3D network [25] employs *spatiotemporal* convolutional encoding for videos. However, the temporal range of its input is an extension of static frames to local 16-frame chunks. Our convolutional encoder is designed to be applied to the entire temporal domain. Besides, for video sequences, our temporal encoding is part of a two-step decomposition with preceding spatial encoding by existing models such as CNNs.

3 Method

This section describes our video encoder as a multi-layer temporal convolutional neural network, and a variant single-layer LSTM decoder with soft attention. Although the encoder is designed and tested only for video captioning, it is in general applicable to any sequence-to-sequence modeling problem.

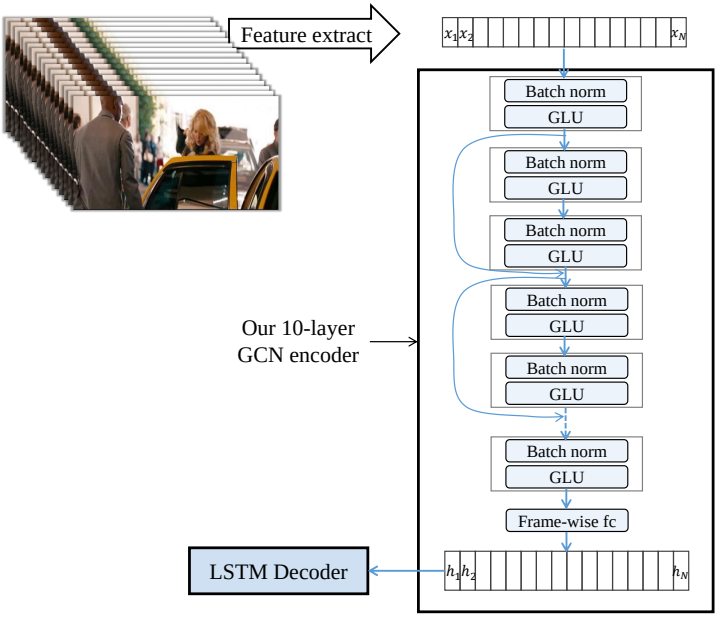


Figure 2: Network overview. We follow the same sequence encoder-decoder framework for video caption generation. Our encoder network (the large black box on the right) is a 10-layer stack of convolutional layers with gated linear units and ResNet-style skip connections.

Given an input video, we extract CNN features using a pretrained CNN model for each video frame. This video sequence input is then embedded into a hidden state $\mathbf{H}$ using our temporal convolutional encoder network. The decoder then takes the hidden representation and produces probabilities of word that would appear in the target video caption. We denote the extracted GoogLeNet features of an input video frames as $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]$ where N is the number of frames, the embedded video frames representation as $\mathbf{H} = [\mathbf{h}_1, ..., \mathbf{h}_N]$, and the final word probability of our attention-based LSTM decoder output as $\mathbf{P} = [\mathbf{p}_1, ..., \mathbf{p}_T]$, which is trained to be aligned with the ground truth labels $\mathbf{y} = [y_1, y_2, ...y_T]$, T is the length of the target caption.

3.1 Convolutional temporal encoder

The main structure of our video encoder is a 10-layer gated convolutional network (GCN), as is shown in Figure 2. Each layer is a batch normalization operation followed with a Gated Linear Unit (GLU) which we describe soon. A skip connection with identity mapping is added for every two layers. The first layer serves as a visual embedding layer that transforms the input CNN feature sequence (2048 GoogLeNet features in our experiments) into the embedding space (see more details see section 4.5), and its input and output are not of the same depth, so the skip connection starts from the second layer, from which all layers outputs embeddings of the same dimensionality. All GCN layers retain the length of the input sequence.

The Gated Linear Unit (GLU) used in each GCN layer is originally proposed in [6] for language modeling. Given a sequence of input $[\mathbf{x}_1, ..., \mathbf{x}_N]$ (video input or output from pre-

vious layer), A GLU computes a linear transformation of the input with a nonlinear gating, both with convolution:

$$o(\mathbf{X}) = (\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{d}) \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{N \times m}$ is the input, $\mathbf{W} \in \mathbb{R}^{k \times m \times n}$, $\mathbf{V} \in \mathbb{R}^{k \times m \times n}$ are kernel matrices of convolutional network, k, m and n are kernel size, input feature dimension and number of filters separately, $\mathbf{b} \in \mathbb{R}^n, \mathbf{d} \in \mathbb{R}^n$ are learnable biases, and σ is the sigmoid function for the gating path. Other activation functions such as ReLU may also apply but sigmoid works better in both [6] and our task. Figure 3 shows an illustration of the GLU unit. The linear path allows gradient to easily pass through the active units, while the gating retains non-linearity of the network. It has shown to have superior performance over more sophisticated LSTM- or GRU-style gating in language modeling [6].

On top of last GCN layer, a frame-wise fully connected layer with dropout [22] is appended to produce the final encoding representation $\mathbf{H}$ of size $N \times n$. The weights of the last fully connected layer are shared across frames so the encoder can handle video sequences of arbitrary length.

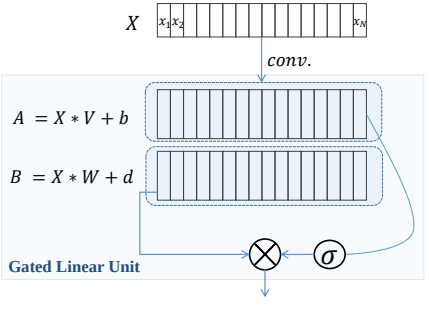


Figure 3: Illutration of the gated linear unit (GLU).

3.2 Caption decoder

In the decoding phase, a single-layer LSTM is used to generate words. Between the encoder and the decoder is a soft *attention* layer that uses the hidden states output by encoder to generate a feature vector $\mathbf{u}_t$ at each timestep t in the decoder. The feature vectors $\mathbf{u}_t$ can be seen as what the decoder takes from the entire video (as a weighted sum of the video sequence, where the weights vary across timesteps). The vector $\mathbf{u}_t$ at timestep t is computed as follows:

$$\mathbf{u}_t = \sum_{i=1}^{N} \alpha_i^t \mathbf{h}_i, \tag{2}$$

where weight α_i^t satisfy the constraint $\sum_{i=1}^{N} \alpha_i^t = 1$ and is updated at every timestep t:

$$\alpha_i^t = \frac{\exp(e_i^t)}{\sum_{j=1}^{N} \exp(e_j^t)}, \tag{3}$$

where

$$e_i^t = \mathbf{v}^\intercal \phi(\mathbf{U}_a \mathbf{s}_{t-1} + \mathbf{W}_a \mathbf{h}_i + \mathbf{b}_a). \tag{4}$$

Here, ϕ is the hyperbolic tangent function tanh, $\mathbf{s}_{t-1}$ is the hidden state of the decoder at previous time step, $\mathbf{U}_a, \mathbf{W}_a, \mathbf{v}, \mathbf{b}_a$ are network parameters.

Except for $\mathbf{u}_t$, the LSTM decoder at each timestep t also expects input $\mathbf{y}_{t-1}$ and $\mathbf{s}_{t-1}$ which are word prediction and hidden state of the previous timestep. With all the inputs, the LSTM states are updated as:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{iy}\mathbf{E}\mathbf{y}_{t-1} + \mathbf{W}_{is}\mathbf{s}_{t-1} + \mathbf{W}_{iu}\mathbf{u}_t + \mathbf{b}_i), && (5)\\
\mathbf{f}_t &= \sigma(\mathbf{W}_{fy}\mathbf{E}\mathbf{y}_{t-1} + \mathbf{W}_{fs}\mathbf{s}_{t-1} + \mathbf{W}_{fu}\mathbf{u}_t + \mathbf{b}_f), && (6)\\
\mathbf{o}_t &= \sigma(\mathbf{W}_{oy}\mathbf{E}\mathbf{y}_{t-1} + \mathbf{W}_{os}\mathbf{s}_{t-1} + \mathbf{W}_{ou}\mathbf{u}_t + \mathbf{b}_o), && (7)\\
\mathbf{g}_t &= \phi(\mathbf{W}_{gy}\mathbf{y}_{t-1} + \mathbf{W}_{gs}\mathbf{s}_{t-1} + \mathbf{W}_{gu}\mathbf{u}_t + \mathbf{b}_g), && (8)\\
\mathbf{c}_t &= \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \mathbf{g}_t, && (9)\\
\mathbf{s}_t &= \mathbf{o}_t \otimes \phi(\mathbf{c}_t), && (10)
\end{aligned}$$

where $\otimes$ is the element-wise product, $\mathbf{W}_{*x}$ and $\mathbf{W}_{*h}$ are learnable matrix, $\mathbf{E}$ is the learnable word embedding matrix, $\mathbf{b}_*$ is the bias vector.

After getting $\mathbf{s}_t$, we learn a linear embedding transforming $\mathbf{s}_t$ to a V dimensional vector where V is the vocabulary size and apply softmax on it to get the probability of each word. Between $\mathbf{s}_t$ and the linear embedding layer, there are an optional *deep output* layer [20] which takes $\mathbf{u}_t, \mathbf{s}_t, \mathbf{y}_{t-1}$ as input and outputs a vector $\mathbf{z}_t$:

$$\mathbf{z}_t = \phi(\mathbf{W}_{zu}\mathbf{u}_t + \mathbf{W}_{zs}\mathbf{s}_t + \mathbf{W}_{zy}\mathbf{E}\mathbf{y}_{t-1} + \mathbf{b}_z). \quad (11)$$

Here, $\mathbf{W}_{zu}, \mathbf{W}_{zs}, \mathbf{W}_{zy}, \mathbf{b}_z$ are all learnable parameters. We adopt Maxout [9] to calculate $\mathbf{z}_t$ as the work of Pan et al. [17] does. Finally, the linear embedding layer transforms $\mathbf{z}_t$ into the V dimensional vector and computes a softmax probability $p_{t,v}$ for each word v at timestep t. The final training objective sums over all training instances and words:

$$\max_{\Theta} \sum_{t=1}^{T} \sum_{v=1}^{V} \mathbb{1}(y_t = v) \cdot log(p_{t,v}) \quad (12)$$

where y_t is the t-th label of the ground-truth video captions from training data.

4 Experiments

This section describes the experimental results of the proposed model on two standard video captioning benchmark datasets: the Microsoft Video Description Corpus (MSVD) and the Montreal Video Annotation Dataset (M-VAD).

4.1 Data sets

MSVD: The Microsoft Video Description Corpus (MSVD) [4] is a widely used dataset for video captioning which contains 1,970 video clips. Each video has multiple descriptions annotated by human. Each description is a single sentence. Although the data set contains multi-lingual descriptions, we only use the English descriptions as previous works [27, 28, 29], which result in a collection of clip-sentence pairs that contains about 80,000 pairs in total. We follow the same strategy used in [11] to split the data into training, validation and testing set.

M-VAD: The Montreal Video Annotation Dataset (M-VAD) [24] is a large-scale movie description dataset that contains 46,589 video clips in total with each video clip labeled by only one description. The dataset is built using descriptive video services (DVS) which generates the description in a semi-automatic way. We follow the standard split provided by [24], that is, 36,921 video clips for training, 7,417 clips for validation and 4,951 clips for test.

4.2 Pre-processing

We use pretrained inception-v3 model (GoogLeNet [23]) to extract static frame features[1]. The features are extracted from the last pooling layer the model. We extract feature for every 10 video frames and get a list of the frame features. For MSVD, each video's frame features list has a maximum length of 80. For MVAD, the maximum length is set to 40, since the average clip length of MVAD is smaller. For video clips that is too short to get the features at maximum length, we pad them with zeros.

To preprocess ground truth descriptions, we remove all punctuation marks and convert the remaining characters to lower case. Then, we tokenize sentences using PTBTokenizer provided in the Stanford CoreNLP tools [15]. We using a special token $\langle$UNK$\rangle$ to replace all word that appear less than two times. We need two extra special tokens $\langle$BOS$\rangle$ and $\langle$EOS$\rangle$. This yields a vocabulary size of $5,427$ for MSVD and $9,614$ for MVAD. We use one-hot vectors to represent words and learn a word embedding at decoding phase. The parameters of word embedding are shared across all timesteps. In training, the ground truth sentence is used with a start token $\langle$BOS$\rangle$ inserted at the beginning and an end token $\langle$EOS$\rangle$ appended in the end. In testing, we only input $\langle$BOS$\rangle$ at the first time step. For every timestep, we choose the word with the maximum probability from the output of previous timestep.

4.3 Baselines

For each dataset, we compare our model with two baseline methods: a standard LSTM encoder-decoder network and a stacked LSTM network. The stacked network uses 4 layers of LSTM encoder. We also tried deeper networks but the 4-layer stacked network worked the best. We also compare our model with two self variants: "GCN w/o skip connection" is our model with the ResNet-style skip connections removed, and "GCN w/o gating" is our model without the gating mechanism in the GLU. For the latter, because the GLU is a linear path with gating, therefore, when we remove the gating, the linear output is passed to a ReLU activation to retain nonlinearity. All of the above baseline method use the same decoder as our final model. We also compare our model with a number of related methods, including state-of-the-art video captioning models: the HRNE model [17] and the hierarchical boundary-aware encoder [1].

4.4 Evaluation metrics

There are several evaluation metrics used in the domain of visual captioning, such as BLEU [19], METEOR [7], ROUGE-L [14] and CIDEr [26]. Vedantam et al. [26] evaluate these four metrics on the image description task. Their results indicate that METEOR is always better than other three metrics when the number of references are small. Thus, METEOR become our first choice naturally. For fair compare, We utilize the Microsoft COCO Evaluation Server [6] to compute all scores in our experiment as previous works. Evaluation results are displayed in Table 1-2.

[1] we also experimented with ResNet [12] features. The score is less as good as that with GoogLeNet features.

Method	*C*	*B*@1	*B*@2	*B*@3	*B*@4	*METEOR*
S2VT-RGB(VGG) [28]	-	-	-	-	-	29.8
SA-GoogleNet + 3D-CNN [29]	51.7	-	-	-	41.9	29.6
LSTM-E (VGG + C3D) [18]	-	78.8	66.0	55.4	45.3	31.0
LSTM2-ATT(SVO) [31]	-	**82.4**	**71.8**	**62.5**	**52.0**	32.3
HRNE (G) [17]	-	78.4	66.1	55.1	43.6	32.1
HRNE (G) w/ attention(G) [17]	-	79.2	66.3	55.1	43.8	**33.1**
Boundary-aware (ResNet+C3D) [1]	-	-	-	-	42.5	32.4
Basic RNN (LSTM)	65.9	76.7	63.9	53.6	42.9	31.5
Stacked RNN (LSTM 4 layers)	53.6	72.7	59.3	49.6	39.6	29.0
GCN w/o skip connection	68.9	77.5	64.4	54.3	43.5	31.9
GCN w/o *gating*	68.5	77.6	64.6	54.4	43.9	31.9
GCN (G)	**72.4**	78.8	66.5	56.4	46.0	**33.1**

Table 1: Experiment results on the MSVD dataset

Method	*METEOR*
S2VT-RGB(VGG) [28]	5.6
SA-GoogleNet + 3D-CNN [29]	4.1
HRNE [17]	5.8
HRNE w/ attention [17]	6.8
Boundary-aware encoder (ResNet+C3D) [1]	**7.3**
Basic RNN (LSTM)	6.4
Stacked RNN (LSTM 4 layers)	6.8
GCN w/o skip connection	6.7
GCN w/o *gating*	6.3
GCN:final	7.1

Table 2: Experiment results on the M-VAD dataset

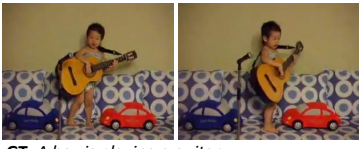
4.5 Implementation details

Our model is implemented on Google Tensorflow. We adopt the following default parameter settings to train models on both datasets. All GCN layers except for the first layer have 512 kernels of size 5×512 with a stride of 1 and zero padding that preserves the sequence length. The first GCN layer has 512 kernels of size $5 \times n$, where $n = 2048$ is the dimensions of the input GoogLeNet feature. In other words, The first GCN layer projects the input feature dimension from 2048 to 512, for all the remaining layers the embedded feature dimension remains unchanged. The LSTM decoder has hidden states of size 512. In our experiments, we trained our models using ADAM [13] with learning rate 2×10^{-4}, decaying parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$. We apply gradient clipping at norm 5. We employ dropout [22] with rate 0.5 at the input and output of all LSTM timesteps. The mini-batch size is set to 128.

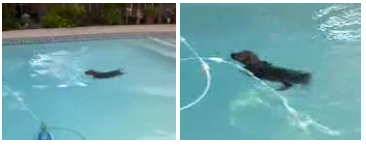
The word embedding size is set to 512 as well and it uses uniform initialization in range $(-\sqrt{3}, \sqrt{3})$. All other learnable matrices are initialized with the initialization method proposed by Glorot et al. [8]. All biases are initialized to zero except for the biases in the forget gate of LSTM which is initialized to 1. We let the training stop at 50 epochs and apply early-stopping criteria based on validation performance.

4.6 Results and Conclusion

Table 1 and Table 2 display the evaluation results on the two benchmark datasets. On the MSVD dataset, our model achieved the best METEOR score over all others, and even better



GT: *A boy is playing a guitar.*
Basic LSTM: a boy is playing a guitar.
HRNE: A man is playing a guitar.
BA encoder: A boy is playing guitar.
Ours: A boy is playing a guitar.



GT: *A dog is swimming in a pool.*
basic LSTM: two dogs are swimming in a pool.
HRNE: A dog is swimming.
Ours: two dogs are swimming in a pool.

GT: *A woman dips a shrimp in batter.*
Basic LSTM: a man is mixing ingredients in a bowl.
HRNE: A woman is cooking.
BA encoder: A woman is adding ingredients to a bowl of food.
Ours: A woman is cooking shrimp.

GT: *A mango is being sliced.*
Basic LSTM: A woman is peeling a potato.
HRNE: A person is preparing an egg.
Ours: A woman is peeling a mango.

GT: *A basketball player is doing a hook shot.*
Basic LSTM: A cat is jumping into a bag.
HRNE: A man is doing a dance.
Ours: A man is talking on a room.

GT: *A woman dials a cell phone.*
Basic LSTM: A girl is putting on her hair.
HRNE: A girl is talking.
Ours: a girl is putting her face.

Figure 4: Qualitative results and comparison.

performance over the boundary-aware encoder [1] which uses richer feature input (ResNet + C3D feature) while our model only uses plain GoogLeNet feature. An exception is the LSTM2-ATT network [31], which delivers higher score in BLEU@k than existing work as well as ours. A likely reason is its sophisticated feature design compared to ours. The HRNE model uses the same feature input as out model, so it is a fairer comparison, for which our model slightly outperform on the BLEU metrics. On the M-VAD dataset, our model outperformed the HRNE model with a considerable margin (7.1 versus 6.8), but did less as good as the boundary-aware encoder which, again, utilized more powerful feature input than our model. The numbers also show the skip connection and the gating scheme both play an important role in the model. When the skip connection is removed, the METEOR score dropped from 33.1 to 31.9 on MSVD and from 7.1 to 6.7 on M-VAD; when gating is removed, METEOR score on M-VAD dropped from 7.1 to 6.3.

Figure 4 shows the results on a set of example videos. For all the examples, our captioning results are meaningful and comparable even to the ground-truth labels. This demonstrates the effectiveness of our convolutional encoder, and opens up the possibility to use new and perhaps more powerful convolutional architectures for sequence modeling, such as the convolutional encoder-decoder networks.

While the current models work reasonably well in generating human language-like word fragments, the results do reveal a few flaws, for example, the model can count the object number wrong (the "dog swimming" example) or misunderstand multi-person interaction scenarios ("playing basketball" instead of "talking") or generate plausible but incorrect phrases ("putting her face" in the last example), suggesting the need of language models or visual encoding schemes beyond state-of-the-art.

5 Acknowledgement

We thank all the anonymous reviewers. This project is supported by China NSF program under grant No. U1509206 and ZJNSF grant under No. Q15F020006.

References

[1] Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. Hierarchical boundary-aware neural encoder for video captioning. *arXiv preprint arXiv:1611.09312*, 2016.

[2] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*, 2016.

[3] David L. Chen and William B. Dolan. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, HLT '11, pages 190–200, Stroudsburg, PA, USA, 2011. ISBN 978-1-932432-87-9.

[4] David L Chen and William B Dolan. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 190–200. Association for Computational Linguistics, 2011.

[5] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[6] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*, 2016.

[7] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *In Proceedings of the Ninth Workshop on Statistical Machine Translation*. Citeseer, 2014.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[9] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C Courville, and Yoshua Bengio. Maxout networks. *ICML (3)*, 28:1319–1327, 2013.

[10] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013. URL http://arxiv.org/abs/1303.5778.

[11] Sergio Guadarrama, Niveda Krishnamoorthy, Girish Malkarnenkar, Subhashini Venugopalan, Raymond Mooney, Trevor Darrell, and Kate Saenko. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2712–2719, 2013.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain, 2004.

[15] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[16] Joe Yue-Hei Ng, Matthew J. Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. *CoRR*, abs/1503.08909, 2015. URL http://arxiv.org/abs/1503.08909.

[17] Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[18] Yingwei Pan, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui. Jointly modeling embedding and translation to bridge video and language. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[19] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[20] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[21] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. A hybrid convolutional variational autoencoder for text generation. *CoRR*, abs/1702.02390, 2017. URL http://arxiv.org/abs/1702.02390.

[22] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[23] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[24] Atousa Torabi, Christopher J. Pal, Hugo Larochelle, and Aaron C. Courville. Using descriptive video services to create a large data source for video annotation research. *CoRR*, abs/1503.01070, 2015. URL http://arxiv.org/abs/1503.01070.

[25] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014. URL http://arxiv.org/abs/1412.0767.

[26] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575, 2015.

[27] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014.

[28] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence-video to text. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4534–4542, 2015.

[29] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4507–4515, 2015.

[30] Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[31] Mihai Zanfir, Elisabeta Marinoiu, and Cristian Sminchisescu. Spatio-temporal attention models for grounded video captioning. In *Asian Conference on Computer Vision*, pages 104–119, 2016.