From Deterministic to Generative: Multimodal Stochastic RNNs for Video Captioning

Jingkuan Song, Yuyu Guo, Lianli Gao, Xueling Li, Fellow, IEEE, Alan Hanjalic, Fellow, IEEE, and Heng Tao Shen

Abstract—Video captioning, in essential, is a complex natural process, which is affected by various uncertainties stemming from video content, subjective judgment, and so on. In this paper, we build on the recent progress in using encoder–decoder framework for video captioning and address what we find to be a critical deficiency of the existing methods that most of the decoders propagate deterministic hidden states. Such complex uncertainty cannot be modeled efficiently by the deterministic models. In this paper, we propose a generative approach, referred to as multimodal stochastic recurrent neural networks (MS-RNNs), which models the uncertainty observed in the data using latent stochastic variables. Therefore, MS-RNN can improve the performance of video captioning and generate multiple sentences to describe a video considering different random factors. Specifically, a multimodal long short-term memory (LSTM) is first proposed to interact with both visual and textual features to capture a high-level representation. Then, a backward stochastic LSTM is proposed to support uncertainty propagation by introducing latent variables. Experimental results on the challenging data sets, microsoft video description and microsoft research video-to-text, show that our proposed MS-RNN approach outperforms the state-of-the-art video captioning benchmarks.

Index Terms—Recurrent neural network (RNN), uncertainty, video captioning.

I. INTRODUCTION

IN RECENT years, various fields of computer vision have developed rapidly, including image recognition [1]–[3], facial recognition [4], [5], action recognition [6]–[8], and other tasks [9]–[11]. With the explosive growth of online videos over the past decade, video captioning has become a hot research topic. In a nutshell, video captioning is the problem of translating a video into meaningful textual sentences describing its visual content. As such, solving this problem has the potential to help various applications from video indexing and search [12]–[16] to human–robot interaction.

Building on the pioneering work of Kojima et al. [17], a series of studies has been conducted to come up with the first generation of video captioning systems [18]–[20]. Recently, however, the development of these systems has more and more relied on deep neural networks (DNNs) that have been proven effective in both computer vision (e.g., image classification and object detection) and natural language understanding (e.g., machine translation and language modeling), forming two technological pillars of video captioning solutions. In particular, deep convolutional neural networks (CNNs) (e.g., VggNet [21] and ResNet [22]) have been widely deployed to extract representative visual features, while recurrent neural networks (RNNs) (e.g., long short-term memory (LSTM) [23] and gate recurrent unit [24]) have been deployed to translate sequential term vectors to natural language sentences. Despite the significant conceptual and computational complexity of these DNN-based models, their effectiveness has given rise to the so-called encoder–decoder scheme as a popular modern approach for video captioning. In this scheme, typically a CNN is used as an encoder and an RNN as a decoder. This approach has shown better performance than the traditional video captioning methods with hand-crafted features.

Recent efforts toward developing and implementing an encoder–decoder scheme for video captioning have mainly focused on solving the following questions.

1) How to help an encode–decoder framework to more efficiently and effectively bridge the gap between video and language [25]?
2) How to facilitate video captioning using semantic information [26]?
3) How to deploy an attention mechanism to help decide what visual information to extract from video [27], [28]?
4) How to extract attributes/key concepts from sentences to enhance video captioning? [29]–[31].

Numerous approaches have been proposed to address these questions [26]–[28], [32], [33]. However, the above-mentioned approaches have been deterministic without incorporating uncertainties (i.e., both subjective judgment and model uncertainty) into the model calculations at all stages of the modeling. First, in essential,
video captioning is a complex process and involves many factors, such as video itself, description intents, personal characteristics, and experiences. Except for the video content, other factors are inherently random and unpredictable. For example, in Fig. 1, we asked three people to describe two videos separately, and they provided different descriptions for each video. This indicates that video captioning is subjective and uncertain. Second, video captioning models are always abstractions of the natural video captioning processes by leaving out some less important components and keeping only relevant and prominent components, thus modeling uncertainty arises. However, both uncertainties are ignored in the previous work.

Therefore, in this paper, we are focusing on dealing with the above-mentioned uncertainties. All our attempts are to ascertain the true nature about video captioning. We propose a novel approach, namely multimodal stochastic RNN networks (MS-RNNs), which model the uncertainty observed in the data using latent stochastic variables. Our method is inspired by variational autoencoder (VAE) [34], which uses a set of latent variables to capture the latent information. This paper makes the following contributions. 1) We propose a novel end-to-end MS-RNN approach for video captioning. To the best of our knowledge, this is the first approach to video captioning that takes the uncertainty, both subjective judgment and model uncertainty, into consideration. Therefore, for each video, our model can generate multiple sentences to describe it from different aspects. 2) We propose a multimodal LSTM (M-LSTM) layer, which incorporates the features from different information sources (i.e., visual and word) into a set of higher level representation by adjusting the weights on each individual source for improving the video captioning performance. 3) We develop a novel backward stochastic LSTM (S-LSTM) mechanism to model uncertainty in a latent process through latent variables. With S-LSTM, the uncertainty is expressed in the form of probability distribution of latent variables. The uncertainty can be model into a prior distribution by making use of the consistency between prior distribution and posterior distribution. 4) The proposed model is evaluated on two challenging data sets, microsoft video description (MSVD) and Microsoft Research (MSR) video-to-text (MSR-VTT). The experimental results show that our method achieves superior performance in video captioning. Note that our model only utilizes the appearance features of videos, and no attention mechanism is incorporated.

II. RELATED WORK

A. Recurrent Neural Networks

RNNs [35] form a directed cycle to connect units. This mechanism allows them to process arbitrary sequential data streams; thus, RNNs have been widely used in computational linguistics and achieved great success. Taking language model as an example, RNNs model a sequential data streams (e.g., a sentence) $s = \{s_1, \ldots, s_T\}$ by decomposing the probability distribution over outputs

$$P(s) = \prod_{t=2}^{T} P(s_t|s_{<t})P(s_1).$$

At each time step, an RNN observes an element and updates its internal states, $h_t = f_{\theta}(h_{t-1}, s_t)$, where $f$ is a deterministic nonlinear function and $\theta$ indicates a set of parameters. The probability distribution over $s_t$ is parameterized as: $P(s_t|s_{<t}) = P_{\theta}(s_t|h_{t-1})$. The RNN language model (RNNLM) [36] parameterized the output distribution by applying a softmax function onto the previous hidden state $h_{t-1}$. To learn the model’s parameters, RNNLM maximizes the log-likelihood by adopting the gradient descent. However, most existing RNNs models propagate deterministic hidden states.

B. Visual Captioning

The study of visual captioning problem has been going on for many years. In 2002, the video captioning system [17] was proposed for describing human behavior; the method first detects visual information (i.e., position of head, direction of head, and positions of hands) to find the position where the person is and the gesture what the person does and then selects appropriate predicate, object, and so on with domain knowledge. Finally, the method applies syntactic rules to generate a whole sentence. Following this work, a series of studies is conducted to utilize such a technique to enhance different multimedia applications [18]–[20]. And there are some works that tackle the problem with the probabilistic graphical model. Farhadi et al. [37] introduce the meaning space, which is represented as triplets of \langle object; action; scene \rangle in the form of a Markov random field, and map the images and
sentence to the meaning space to find the relationship between images and sentences. Rohrbach et al. [38] try to model the relationship between different components of the visual information with a conditional random field and then tackle the captioning problem as a machine translation problem to generate sentences.

Inspired by the recent advances in image classification using CNN networks (e.g., VggNet [21], GoogLeNet [39], and ResNet [22]), and in machine translation utilizing RNN, there have been a few attempts [26], [27], [32], [33], [40]–[42] to address video caption generation by first adopting an efficient CNN network to extract video appearance features and second utilizing an RNN to take video features and the previous predicted words to infer a new word with a softmax. In order to further improve the performance, more complex approaches [26], [27], [33] are proposed from different aspects. Specifically, Yao et al. [27] adopted a spatio-temporal CNN (3-D CNN) for capturing video motion information and a soft attention mechanism to select relevant frame-level CNN (3-D CNN) for capturing video motion information. Pan and a soft attention mechanism to select relevant frame-level CNN (3-D CNN) for capturing video motion information.

To answer question under specific scenarios; and 6) subjective judgment, which occurs as a result of interpretation of data. Without sufficient data, the experts’ judgment will be based on observations and experience. All of these uncertainties are hidden factors affecting the results of video captioning, and we propose to model these uncertainties using latent stochastic variables.

D. Variational Autoencoder

As mentioned earlier, we know that we should find a method to capture the uncertainty in the video captioning problem. But how can we model the uncertainty? VAE [34] model gives us a good way to solve this problem. For capturing the variations in the observed variables x, the VAE model introduces a set of latent random variables z and rewrites the objective function log P(x) as follows:

$$\log P(x) \geq E_Q[\log P(x|z)] - KL[Q(z|x)||P(z)] := \mathcal{L}$$

where $KL[Q||P]$ is the Kullback–Leibler divergence between two distributions $Q$ and $P$, which measures the nonsymmetric difference between two probability distributions. And $Q(z|x)$ is an approximate posterior distribution, which avoids to solve the intractable true posterior distribution. In [34], the VAE model was used to paint the digits, so it needs to decide not just which number is written but the angle, the stroke width, and also abstract stylistic properties, so the model uses a set of latent random variables to capture the latent information. Inspired by this, we also use latent variables with a stochastic layer to capture the uncertainty information in the video captioning. Different with painting digits, the video captioning task needs to generate different sentences based on the content of the video, so our objective function is a conditional probability and we use the loss function introduced in conditional VAE (CVAE) [48], which extend the VAE to dispose conditional probability distribution. And Krishnan et al. [49] compared the different variational models, and they guide us to choose an effective variational model. And there are some works that extend the VAE model to RNN [50]–[52] for generating speech or music signal. All these works inspire us to extend the captioning problem to an uncertainty problem.

III. PROPOSED APPROACH

In this section, we introduce our approach for video captioning, and we follow the conventional encoder–decoder framework. The encoder is based purely on neural networks to generate video descriptions, and the decoder, named MS-RNNs (see Fig. 2), is our major contribution. We first introduce the architecture of our proposed network and then devise the loss function and optimization.

A. Problem Formulation

Given a video v with N frames, we extract their frame-level features, and v can be represented as $v = [v_1, v_2, ..., v_i, ..., v_N]$, where $v_i \in \mathbb{R}^{D_v \times 1}$ and $D_v$ is the dimension of the frame-level features. For each v, we also have a textual sentence a to describe it, and a includes T words, which can be represented as $a = [a_1, a_2, ..., a_t]$. Specifically, $a_t \in \mathbb{R}^{D_a \times 1}$ is the one-hot vector, where $D_a$ is the
This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Fig. 2. End-to-end multimodal RNNs stochastic architecture for video captioning. The S-LSTM is proposed to introduce latent variables to propagate uncertainty. During the training phase, S-LSTM enables the consistency between prior distribution and posterior distribution. Therefore, during the test phase, we only need the learned prior distribution to support video caption generation. It is a common strategy in the VAE model. And we use the B-LSTM to infer the posterior distribution over latent variables, so the B-LSTM layer is removed during the test phase. Our MS-RNN model uses an embedding layer for mapping the one-hot word vectors to low-dimensional vectors \( s \) and then adds the LSTM layer to explore the temporal information between the low-dimensional vectors and generates sentence features \( s' \). We mix the visual feature \( v \) and sentence features \( s' \) with the M-LSTM layer. The latent variables \( z_t \) are generated by the S-LSTM layer, which includes a backward-LSTM and a stochastic layer. Finally, the word probabilities are generated by the softmax layer with latent variables \( z_t \) and mixed features \( l_t \). During the testing phase, the model generates words one by one with the beam search algorithm.

B. Encoder

The goal of an encoder is to compute feature vectors that are compact and representative and can capture the most related visual information for the decoder. Specifically, it encodes the input video into a continuous representation, which may be a variable-sized set \( v = \{v_1, v_2, \ldots, v_i, \ldots, v_N\} \). Thanks to the rapid development of deep CNNs, which have made a great success in a large-scale image recognition task [22], object detection [53], and visual captioning [25], high-level features can be extracted from upper or intermediate layers of a deep CNN network. Therefore, a set of well-tested CNN networks, such as the ResNet-152 model [22] which has achieved the best performance in ImageNet Large-Scale Visual Recognition Challenge, can be used as candidate encoders for our framework. With a pretrained deep CNN (ResNet-152 or GoogLeNet in our experiments) on the ImageNet data set, we can apply it to each frame to extract representative frame-level features.

For encoding the sentence, because of the sparsity of one-hot vectors \( a = \{a_1, a_2, \ldots, a_t, \ldots, a_T\} \), like previous works [27] and [28], we process one-hot vector with an "embedding" method. We set a parameter matrix \( U_s \in \mathbb{R}^{D_s \times D_a} \) to map the one-hot vectors \( a \) to \( s \) as follows:

\[
    s = U_s a. \tag{3}
\]

The \( s \in \mathbb{R}^{D_s \times T} \) and \( s = \{s_1, s_2, \ldots, s_i, \ldots, s_T\} \) will be input to the next step. In addition, the end of sentence \( a_{T+1} = <eos> \) is mapped to \( s_{T+1} \).

C. Decoder With MS-RNN

The MS-RNN consists of three core components as shown in Fig. 2: a basic LSTM layer for extracting word-level features, an M-LSTM layer for encoding multiview information (visual and textual features) simultaneously and chronologically, and a backward S-LSTM layer to adequately introduce latent variables.
1) LSTM for Word Features: In our MS-RNN model, we use a basic LSTM layer to take \( s = \{s_1, s_2, \ldots, s_T\} \) as input and output word features \( s' = \{s'_1, s'_2, \ldots, s'_T\} \) with encoded temporal information

\[
s'_t = \text{LSTM}(s_t, s'_{t-1}), \quad t \in \{1, 2, \ldots, T\}
\]

where \( s'_0 = 0 \). More specifically, a standard LSTM unit consists of three gates: a “forget gate” \( f_t \) that decides what information we are going to throw away from an LSTM unit; an “input gate” \( i_t \) that decides what new information we are going to store in the cell state; and an “output gate” \( o_t \) that controls the extent to which the value in memory is used to compute the output activation of the block. A standard LSTM can be defined as

\[
\begin{align*}
    f_t &= \sigma(W_{sf}s_t + W_{hf}s'_{t-1} + b_f) \\
    i_t &= \sigma(W_{si}s_t + W_{hi}s'_{t-1} + b_i) \\
    o_t &= \sigma(W_{so}s_t + W_{ho}s'_{t-1} + b_o) \\
    g_t &= \phi(W_{sg}s_t + W_{hg}s'_{t-1} + b_g) \\
    c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
    s'_t &= o_t \odot \phi(c_t)
\end{align*}
\]

where \( \sigma(\cdot) \) is a sigmoid function, \( \phi(\cdot) \) denotes a hyperbolic tangent function, \( c_t \) is a cell state vector, \( s'_t \) is an output vector, \( g_t \) is a sigmoid gate, \( W_\_ \) is a set of parameters, \( \odot \) denotes the elementwise multiplication, and \( b_\_ \) is a set of bias values. Then, for each word \( s_t \), we extracted its word features as \( s'_t \).

2) Multimodal LSTM Layer: Next, an M-LSTM layer takes \( s' \) and a video-level feature \( \overline{v} \) as inputs to fuse a high-level feature \( l_t \)

\[
l_t = M\_\text{LSTM}(s'_t, \overline{v}, l_{t-1}) \quad t \in \{1, 2, \ldots, T\}.
\]

Here, instead of using advanced but complex temporal or spatial attention mechanism to select a video-level feature, we use the basic mean pooling strategy to obtain one \( \overline{v} \)

\[
\overline{v} = \frac{1}{N} \sum_{i=1}^{N} v_i, \quad v_i \in v.
\]

The motivation is that if our model using the basic way to utilize the visual features can improve the performance of video captioning, the advantages of our MS-RNN are manifest. However, as shown in [28] and [29], the attention mechanism can further boost the performance of video captioning.

M-LSTM is a novel variant of LSTM, and it not only inherits the numerical stability of LSTM but also generates plausible features from multiview sources. We choose LSTM as our basic RNN unit due to the following reasons: 1) it achieved great success in machine translation, speech recognition, and image and video caption [25, 54, 55] and 2) compared with basic RNN units, it is absolutely capable of handling the “long-term dependences” problem.

Given two modalities \( s' = \{s'_1, s'_2, \ldots, s'_t, \ldots, s'_T\} \) and \( \overline{v} \) as the inputs, and two initialized vectors \( l_0 \) and \( c_0 \), an M-LSTM can be used to fuse them and extract a higher level feature. An M-LSTM unit can be described as follows:

\[
l_t = M\_\text{LSTM}(s'_t, \overline{v}, l_{t-1})
= \text{LSTM}([s'_t, \overline{v}], l_{t-1})
\]

where \( [s'_t, \overline{v}] \) is a concatenation vector between \( s'_t \) and \( \overline{v} \). To obtain an abstract concept from two modalities, the M-LSTM needs to first project \( s'_t \) and \( \overline{v} \) into a common feature space, and then, the inside gates can add them together with an activation function. Then, in each time step \( t \), we extracted a higher level feature \( l_t \).

3) Backward Stochastic LSTM: In this section, we introduce our backward S-LSTM to take the output of M-LSTM to approximate the posterior distributions over latent variables defined as \( z = \{z_1, z_2, \ldots, z_T\} \), where \( z_t \in \mathbb{R}^{D_z} \). The S-LSTM consists of two units: a backward LSTM unit and a stochastic unit. We define the output of the backward LSTM as \( r_t \).

For the backward LSTM unit in time step \( t \), its output is defined as

\[
r_t = B\_\text{LSTM}(s_{t+1}, l_t, r_{t+1}) \quad t \in \{1, 2, \ldots, T\}
\]

where \( l_t \) is the output of M-LSTM at time step \( t \), \( s_{t+1} \) is the output of embedding layer, and \( r_{t+1} \) is initialized to zero vector. The form of \( B\_\text{LSTM} \) is similar to \( M\_\text{LSTM} \), but it processes sequence with backward direction. We can see that the output of backward LSTM in time step \( t \) depends on the present input \( l_t \) and \( s_{t+1} \), and future output \( r_{t+1} \). This is because in the stochastic units, the posterior distribution of \( z_t \), which is calculated with (15), does not depend on the past outputs and deterministic states but depend on the present and future ones. Therefore, we propose to use the backward LSTM to extract the future information and incorporate it with a stochastic layer to achieve our goal.

Fig. 3 demonstrates the stochastic unit structure. To obtain \( z_t \), we utilize an “reparameterization trick” introduced in [34]. This trick randomly samples a set of values \( \epsilon_t \in \mathbb{R}^{D_z} \) from a standard Gaussian distribution. Therefore, \( \epsilon_t \sim N(0, 1) \). If we assume \( z_t \sim N(\mu_t, \text{diag}(\sigma_t^2)) \), we can use \( z_t = \mu_t + \sigma_t \odot \epsilon_t \) to calculate \( z_t \). Next, we need to solve the problem of how to learn \( \mu_t \) and \( \sigma_t \) for \( z_t \).

In detail, the stochastic unit takes \( l_t \) and \( z_{t-1} \) as input to approximate \( \mu_{p_t} \) and \( \sigma_{p_t} \) by two feedforward networks (i.e., FC\(_{pt1}\) and FC\(_{pt2}\)). In addition, each of them contains two fully
connected layers

\[
\mu_{q_t} = FC_{q1}(\{z_{t-1}, l_t\}) \\
\sigma_{q_t} = \exp(0.5 \times FC_{q2}(\{z_{t-1}, l_t\))
\]  \hspace{1cm} (10)

\[z_{t-1}, l_t\} is a concatenation operation. In addition, the stochastic unit also takes \(r_t\) and \(z_{t-1}\) to approximate \(\mu_{q_t}\) and \(\sigma_{q_t}\) by two feedforward networks (i.e., FC\(_{q1}\) and FC\(_{q2}\))

\[
\mu_{q_t} = FC_{q1}(\{z_{t-1}, r_t\}) \\
\sigma_{q_t} = \exp(0.5 \times FC_{q2}(\{z_{t-1}, r_t\}))
\]  \hspace{1cm} (11)

For training, we set \(z_t = \mu_{q_t} + \mu_{p_t} + \sigma_{q_t} \odot \epsilon_t\), and this method, introduced in [51], can improve the posterior approximation by using the prior mean, while for testing, we set \(z_t = \mu_{p_t} + \sigma_{p_t} \odot \epsilon_t\), and we set \(z_0\) as zero vector at the beginning. To output a symbol \(a_t\), a probability distribution over a set of possible words is obtained using \(U_P\) and \(z_t\)

\[
P(a_{t+1}|z_t, l_t) = \text{softmax} \left( U_P[z_t, l_t] + b \right)
\]  \hspace{1cm} (12)

where \(U_P\) and \(b\) are the parameters to be learned. Next, we can interpret the output of the softmax layer \(P(a_{t+1}|z_t, l_t)\) as a probability distribution over words.

D. Loss Function

Based on the variational inference and CVAE proposed in [48], we define the following loss function:

\[
\log P(a|l) \geq E_Q[\log P(a|z, l)] - KL(Q(z|a, l)||P(z|l)) := \mathcal{L}
\]  \hspace{1cm} (13)

where \(\mathcal{L}\) is the evidence lower bound of the log likelihood. The distribution \(Q(z|a, l)\) is an approximate posterior distribution, which aims to approximate the intractable true posterior distribution. The first term \(E_Q[\log P(a|z, l)]\), which is an expected log likelihood under \(Q(z|a, l)\), is written as

\[
E_Q[\log P(a|z, l)] = E_Q \left[ \sum_{t=1}^{T} \log P(a_{t+1}|z_t, l_t) \right]
\]

\[
= \sum_{t=1}^{T} \log P(a_{t+1}|z_t, l_t).
\]  \hspace{1cm} (14)

Here, we process the concatenation vector \(\{z_t, l_t\}\) with a softmax layer, mentioned by (12), to approximate \(P(a_{t+1}|z_t, l_t)\).

The second term \(KL(Q(z|a, l)||P(z|l))\), namely KL term, is the Kullback–Leibler divergence, which measures the nonsymmetric difference between two probability distributions (i.e., \(Q(z|a, l)\) and \(P(z|l)\)). And in this paper, we choose the variational model introduced in [49] to factorize the posterior distribution. The posterior and prior distributions are factorized as follows:

\[
Q(z|a, l) = \prod_{t=1}^{T} Q(z_t|z_{t-1}, a_{t+1}, l_{t+1})Q(z_0|a_0, l_0)\geq 0)
\]  \hspace{1cm} (15)

\[
P(z|l) = \prod_{t=1}^{T} P(z_t|z_{t-1}, l_t)P(z_0|l_0).
\]  \hspace{1cm} (16)

For approximating \(Q(z_t|z_{t-1}, a_{t+1}, l_{t+1})\) and \(P(z_t|z_{t-1}, l_t)\), we first use a backward LSTM layer to encode \(s_{t+1}\) [we have encoded \(a_{t+1}\) to \(s_{t+1}\) mentioned in (3)] and \(l_t\) to \(r_t\), and then utilize the method, mentioned in Section III-C3, to approximate the means and the variances of \(Q(z_t|z_{t-1}, a_{t+1}, l_{t+1})\) and \(P(z_t|z_{t-1}, l_t)\). So, we can use the following function to calculate the Kullback–Leibler divergence at the \(t\)th time step:

\[
KL(Q_t||P_t) = \sum_{t=1}^{D_t} \log \frac{Q(z_t|z_{t-1}, a_{t+1}, l_{t+1})}{Q(z_t|z_{t-1}, l_t)} \frac{P(z_t|z_{t-1}, l_t)}{Q(z_t|z_{t-1}, a_{t+1}, l_{t+1})}
\]

\[
= \sum_{t=1}^{D_t} \log \frac{\sigma_{q_t}}{\sigma_{p_t}} + \frac{\sigma_{q_t}^2 + (\mu_{q_t} - \mu_{p_t})^2}{2\sigma_{p_t}^2} - \frac{1}{2}.
\]  \hspace{1cm} (17)

For the whole sentence generation, we calculate the global Kullback–Leibler divergence \(KL(Q(z|l)||P(z|l))\) by

\[
KL(Q(z|a, l)||P(z|l)) = \sum_{t=1}^{T} KL(Q_t||P_t).
\]  \hspace{1cm} (18)

In this paper, we maximize the above-proposed loss function to learn all the parameters. More specifically, we use a back-propagation through time algorithm to compute the gradients and conduct the optimization with ADADelta [56].

IV. EXPERIMENT

We evaluate our model on two standard video captioning benchmark data sets: the widely used MSVD [58] and the large-scale MSR-VTT [59].

MSVD: This data set consists of 1970 short video clips collected from YouTube, with an average length of about 9 s. In addition, this data set contains about 80,000 clip-description pairs labeled by Amazon Mechanical Turkers (AMT). In other words, each clip has multiple sentence descriptions. In total, all the descriptions contain nearly 16,000 unique vocabularies. Following previous works [27], [33], and [34], we split this data set into a training, a validation, and a testing data set with 1200, 100, and 670 video clips, respectively.

MSR-VTT: This data set was proposed by Xu et al. [59] in 2016. They aim to provide a new large-scale video benchmark for supporting video understanding, especially for the task of translating videos into text. In total, this data set contains 10k Web video clips and 200k clip-sentence pairs in total. Each clip is annotated with 20 natural sentences by 1327 AMT workers. This data set is collected from a commercial video search engine, and so far, it covers the most comprehensive categories and diverse visual content, representing the largest data set in terms of sentences and vocabularies. We run our experiments on their updated version with sentence quality control. This data set is divided into three subsets: 65% for training, 5% for validating, and 30% for testing.

A. Evaluation Metrics

To evaluate the performance of our model, we utilize the following four evaluation metrics: bilingual evaluation under study [59], Metric for Evaluation of Translation with Explicit ORdering (METEOR) [60], consensus-based image description evaluation (CIDER) [61], and Recall-Oriented Understudy
Fig. 4. Demonstration of our results, which are generated by repeatedly inputting each video five times into our trained model on the MSVD data set. Our model is able to generate different captions based on the different hidden stochastic variables.

for Gisting Evaluation [62]. In addition, Microsoft COCO evaluation server [63] has implemented these metrics, so we directly call such evaluation functions to test the performance of video captioning.

B. Experimental Settings

1) Video Appearance Feature Extraction: The experimental results obtained by Xu et al. [59] show that applying different pooling methods (i.e., single frame, meaning pooling, and soft attention) obtains different performance. Both mean pooling and soft attention perform significantly better than the single frame. The soft attention performs slightly better than mean pooling with 0.6% BULE@4 and 0.6% METEOR increase, but it involves more operations. Therefore, we apply a mean pooling to a set of frame-level features to generate a representative video-level feature. In addition, we follow previous work [27] to uniformly sample $K = 28$ frames from each clip for controlling video frames duplication. Deep CNNs achieved
a great success in image feature extraction. Therefore, in this paper, we, respectively, use the ResNet-152 and GoogLeNet, the two state-of-the-art CNNs, to extract video-frame level features to analyze our model.

About GoogLeNet, Szegedy et al. [39] introduced an inception module, an optimal local sparse structure in convolutional vision networks, and stacked these modules to construct a 22-layer inception network. The inception module is made up of 1 × 1 convolutions, 3 × 3 convolutions, 5 × 5 convolutions, and 3 × 3 max pooling layers. They used asynchronous stochastic gradient descent (SGD) with 0.9 momentum and decreased the learning rate by 4% every 8 epochs to learn the parameters in GoogLeNet. About ResNet, He et al. [22] introduced a deep residual learning framework, which is constructed by building blocks, to solve the degradation problem of training accuracy. A building block is made up of three convolutions layer: 1 × 1, 3 × 3, and 1 × 1 convolutions. They trained the entire network by SGD with backpropagation. They set the learning rate as 0.1 and divided it by 10 when the error plateaus. In this paper, we used the 152-layer ResNet with 5.7% validation top-5 error on ImageNet.

Like most previous works on video captioning [25], [27], [33], we extract the video-frame level features with pretrained deep CNNs and store these features on disks first and then feed them into the MS-RNN model both for training and testing. We did not fine-tune or retrain these deep CNNs but directly extract features from the pool5 layer with parameters shared at GitHub. The results show that ResNet-152 features perform well.

2) Sentence Preprocessing: For the MSVD data set, we tokenize it by first converting all words to lowercases and second utilizing the WordPunct function from NLTK toolbox to tokenize sentences and remove punctuations. As a result, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a 23,662 size vocabulary from its training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set. For the MSR-VTT data set, after tokenization, we obtain a vocabulary with 13,010 words from the MSVD training data set.

In addition, in our experiments, we use ADADELTA, which can dynamically adjust the learning rate, to learn parameters and set the beam search size as 5. Empirically, we set all the M-LSTM unit sizes as 512, all the B-LSTM unit sizes as 512, the dimension of latent variables as 256, and the word embedding size as 512. Our objective function (13) is optimized over the whole training video sentence pairs with mini-batch 64 in size of MSVD and 256 in size of MSR-VTT. We stop training our model until 500 epochs are reached or until the evaluation metric does not improve on the validation set at the patience of 20. In addition, we multiply the KL term by a scalar, which starts at 0.01 and linearly increases to 1 over the first 20 epochs.

4) Testing Details: During testing, our model takes the video and a begin-of-sentence tag (bos) as the inputs to generate sentences to describe the input video. After the parameters are learned, we perform the generation with beam search [64]. All experiments are conducted on Ubuntu 14.04 with an Intel(R) Core(TM) i7-5930K CPU, a GeForce GTX TITAN Z GPU, and 64-GB memory cards. And Theano [65] library is utilized to construct models.

In addition, our model incorporates latent variables for ascertaining the true nature about video caption and has potential to describe video from different aspects. Thus, we have repeatedly input the test videos into our trained model five times. Each time we obtain a performance showing in Table I. Finally, we obtain an average performance. Moreover, Fig. 4 shows some output examples.

| C. Results on MSVD Data Set |

In this paper, we propose to utilize the probability distribution of latent variables to depict uncertainty; thus, for each time, our model may generate different descriptions. In this section, we run the testing five times and report the results in Table I. The performance of each testing is quite stable and reasonable. By checking the generated sentences (see Fig. 4), we can see that our model can describe a video from various aspects, and likely in real life, human provides various sentences to describe one video to fit their intents.

D. Component Analysis

In this paper, we design two core components: an M-LSTM layer and an S-LSTM layer, which affect the performance of our algorithm. In this section, we study their performance variance with the following two settings:

1) only using M-LSTM for video captioning (M);
2) incorporating M-LSTM and S-LSTM for video captioning (M+S).

In this subexperiment, we first conduct the experiments on the MSVD data set and use ResNet to extract frame features.

Table II lists the results, which demonstrate that our MS-RNN model with both M-LSTM and S-LSTM outperforms M-LSTM only on all evaluation metrics, with a 1.3% M, 3.3% C, and 1% RL performance increase.

In Fig. 4, we show some example sentences generated by our approach, with only M-LSTM and with both M-LSTM

---

1https://github.com/KaimingHe/deep-residual-networks
2https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet
and S-LSTM, respectively. From Fig. 4, we have the following observations.

1) Both M-LSTM and M-LSTM+S-LSTM are able to generate accurate descriptions for a video. In addition, the results generated by M-LSTM+S-LSTM are generally better than the M-LSTM method, which is consistent with the results reported in Table II.

2) M-LSTM is deterministic, and it can only generate one sentence, while M-LSTM+S-LSTM can produce different sentences.

3) In general, M-LSTM+S-LSTM can provide more specific, comprehensive, and accurate descriptions than M-LSTM. For example, in the top-left example, M-LSTM generates “a women is playing a guitar,” while M-LSTM+S-LSTM provides “a girl is singing” and “a women is playing a guitar.” From the middle bottom, we can see that M-LSTM provides a wrong description “cucumber,” while M-LSTM+S-LSTM generates “vegetables” and a set of verbs “slicing, chopping, and cutting.”

4) Our MS-RNN model may produce duplicate and comprehensive results, which is consistent with the nature of video captioning.

5) The last column shows some wrong examples. For the top-right example, both the methods provide wrong descriptions, “cutting a cucumber” and “slicing a carrot.” This is mainly because the MSVD data set contains many videos about cooking and few videos about folding paper, which leads to an overfitting problem. In addition, the right middle is also inaccurate. This is because both our models only take video appearance features as inputs and ignores the motion features. For the right bottom example, our model does not correctly identify the number of objects in some cases.

### Table II

<table>
<thead>
<tr>
<th>Model</th>
<th>B@1</th>
<th>B@2</th>
<th>B@3</th>
<th>B@4</th>
<th>M</th>
<th>C</th>
<th>RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>82.7</td>
<td>71.8</td>
<td>62.7</td>
<td>52.2</td>
<td>32.5</td>
<td>71.5</td>
<td>69.2</td>
</tr>
<tr>
<td>M+S</td>
<td>82.9</td>
<td>72.6</td>
<td>63.5</td>
<td>53.3</td>
<td>33.8</td>
<td>74.8</td>
<td>70.2</td>
</tr>
</tbody>
</table>

### Table III

Comparing the quality of sentence generation on different video spatial representations on the MSVD data set. (V), (G), and (R) stands for the VGGNet, GoogleNet, and ResNet, respectively. This experiment is conducted on the MSVD data set. All the values are reported as percentage (%).

<table>
<thead>
<tr>
<th>Model</th>
<th>B@1</th>
<th>B@2</th>
<th>B@3</th>
<th>B@4</th>
<th>M</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM-E(V) [27]</td>
<td>74.9</td>
<td>60.9</td>
<td>50.6</td>
<td>40.2</td>
<td>29.5</td>
<td>-</td>
</tr>
<tr>
<td>h-RNN(V) [34]</td>
<td>77.3</td>
<td>64.5</td>
<td>54.6</td>
<td>44.3</td>
<td>31.1</td>
<td>62.1</td>
</tr>
<tr>
<td>SA(G) [28]</td>
<td>79.1</td>
<td>63.2</td>
<td>51.2</td>
<td>40.6</td>
<td>29.0</td>
<td>-</td>
</tr>
<tr>
<td>MFA-LSTM(R) [30]</td>
<td>81.3</td>
<td>69.8</td>
<td>60.5</td>
<td>50.4</td>
<td>32.2</td>
<td>69.8</td>
</tr>
<tr>
<td>MS-RNN(G)</td>
<td>80.3</td>
<td>68.4</td>
<td>57.7</td>
<td>48.0</td>
<td>31.0</td>
<td>65.6</td>
</tr>
<tr>
<td>MS-RNN(R)</td>
<td>82.9</td>
<td>72.6</td>
<td>63.5</td>
<td>53.3</td>
<td>33.8</td>
<td>74.8</td>
</tr>
</tbody>
</table>

### Table IV

Comparing the computational cost. The top model uses only M-LSTM, while the bottom model integrates M-LSTM and S-LSTM. The time costs of average forward-backpropagation and average forward are calculated during one update (batch size = 64). The testing duration is calculated with the beam search algorithm (beamwidth = 5) on the MSVD testing data set.

<table>
<thead>
<tr>
<th>Model</th>
<th>AVG Forward-Backpropagation</th>
<th>AVG Forward</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>187.3 ms</td>
<td>15.4 ms</td>
<td>71.41 s</td>
</tr>
<tr>
<td>M+S</td>
<td>362.4 ms</td>
<td>35.4 ms</td>
<td>74.50 s</td>
</tr>
</tbody>
</table>

### E. Comparison of Computational Cost

In this section, we compare computational time cost between M-LSTM and M-LSTM+S-LSTM and show the results in Table III. The training time cost of M-LSMT+S-LSTM is longer than M-LSTM, and the testing time cost is close to that of M-LSTM. The results are reasonable because there are three LSTM layers in MS-RNN during the training phase, but two LSTM layers during the testing phase.

### F. Comparison Results on MSVD Data Set

In this section, we conduct experiments to examine how different video representations work on video captioning, as well as comparing our model with existing approaches. In addition, all the approaches in these subexperiments only take one type video representation extracting from VggNet (V), GoogleNet (G), or ResNet (R). We conduct our experiments on the MSVD data set.

Table IV lists the experimental results. From Table IV, we have following observations.

1) With only appearance features, our MS-RNN (R) model achieves the best performance on all evaluation metrics. Compared with the state-of-the-art method MFA-LSTM (R), our model achieves significantly better performance with 1.6%, 2.8%, 3%, 2.9%, 1.6%, and 5% increase on B@1, B@2, B@3, B@4, M, and C, respectively.

2) For video captioning task, the RestNet-based video representation performs better than both VggNet-based and GoogleNet-based video features. Specifically, our model RestNet feature performs better than GoogleNet features. For the whole experimental results, the approaches (SCN-LSTM and MFA-LSTM) with ResNet-based features perform better than the methods with GoogleNet or VggNet-based features.

3) Compared with the methods using attention mechanisms, e.g., temporal attention [27], our MS-RNN (R) achieves even better results with 3.8%, 9.4%, 12.3%, 12.7%, and 4.8% increase on B@1, B@2, B@3, B@4, and M by using a simple mean pooling strategy. This indicates the advantages of our proposed MS-LSTM.

We also compare our methods with the others using multiple features. Specifically, in this section, we compare our model using only appearance features with six state-of-the-art methods: LSTM-E(V+C) [26], spatial attention (SA) (G+3-DCNN) [27], HRNE-AT(G+C) [32], h-RNN(V+C) [33],...
TABLE VI
EXPERIMENT RESULTS ON THE MSR-VTT DATA SET. SA-LSTM RUNS
EMPLOY SOFT ATTENTION OVER THE FRAME-LEVEL FEATURES
EXTRACTED FROM DEEP NETWORK, WHILE MP-LSTM AND OUR
METHOD UTILIZE MEAN POOLING OVER THE FRAME-LEVEL
VIDEO FEATURES

<table>
<thead>
<tr>
<th>Model</th>
<th>R@4</th>
<th>M</th>
<th>C</th>
<th>RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP-LSTM(VGG-19)</td>
<td>34.8</td>
<td>24.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MP-LSTM(C3D)</td>
<td>35.4</td>
<td>24.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MP-LSTM(VGG-19+C3D)</td>
<td>35.8</td>
<td>25.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SA-LSTM(VGG-19)</td>
<td>35.6</td>
<td>25.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SA-LSTM(C3D)</td>
<td>36.1</td>
<td>25.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SA-LSTM(VGG-19+C3D)</td>
<td>36.6</td>
<td>25.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFA-LSTM(R+C)</td>
<td>39.2</td>
<td>26.9</td>
<td>44.6</td>
<td>60.1</td>
</tr>
<tr>
<td>MS-RNN(R)</td>
<td>39.8</td>
<td>26.1</td>
<td>40.9</td>
<td>59.3</td>
</tr>
</tbody>
</table>

MFA-LSTM(R+C) [29], and SCN-LSTM(R+C) [66], which make use of both appearance and motion video features. Here, V and R are short for VggNet and ResNet, which are used to extract appearance features. 3-D and C are short for 3-DCNN and C3D, which are used to generate video motion features.

The experimental results are shown in Table V. Although our model only uses appearance features, it performs better than the existing methods on B@2 (72.6%), B@3 (63.5%), B@4 (53.3%), and M (33.8%) and achieves comparable results on B@1 (82.9%) and C (74.8%).

G. Comparison Results on MSR-VTT Data Set

In this section, we compare our method with MP-LSTM [40] and SA-LSTM [27] on the MSR-VTT data set. In addition, to obtain the appearance features, the MP-LSTM and our MS-RNN are based on the mean pooling strategy, while SA-LSTM is based on a soft-attention mechanism. In theory, soft attention is more complex than mean pooling but usually provides better visual features. The experimental results are shown in Table VI, and we have the following observations.

MS-RNN:

1) gains a promising performance with 39.8% B@4, 26.1% M, 40.9% C, and 59.3% RL on the MSR-VTT data set.
2) Overall with the same visual input (VGG-19, VGG-19+C3D, or C3D), SA-LSTM performs better than MP-LSTM. However, SA is based on the soft attention. In other words, in theory, SA-LSTM takes better visual features as inputs. Compared with MP-LSTM, our MS-RNN (R) outperforms MP-LSTM (VGG-19+3C3D) with 4% B@4 and 0.8% M increase. Compared with SA-LSTM, our MS-RNN (R) outperforms SA-LSTM(VGG-19+C3D) with 3.2% B@4. Compared with MFA-LSTM(R+C), our model achieves comparable results on B@4, M, and RL by using single feature (R).

V. CONCLUSION AND FUTURE WORK

In this paper, we propose an MS-RNN framework for video captioning. This paper has shown how to extend the modeling capabilities of RNN by approximating both prior distribution and true posterior distribution with a nonlinear latent layer (S-LSTM). In addition, MS-RNN achieves the state-of-the-art performance with only mean video appearance features and is comparable with the counterparts, which take both video appearance and motion features. Last but not least, the proposed model can be applied to a wide range of video analysis applications.

In the future, we will integrate the state-of-the-art attention mechanism [27] with our model to further improve the video captioning performance. Moreover, the motion feature will be considered.

REFERENCES

Z. Wang et al. received the Ph.D. degree in information technology from The University of Queensland, Brisbane, QLD, Australia, in 2014. He was with Columbia University, New York, NY, USA, as a Post-Doctoral Research Scientist, from 2016 to 2017, and the University of Trento, Trento, ON, Canada, as a Research Fellow, from 2014 to 2016. He is currently a Professor with the University of Electronic Science and Technology of China, Chengdu, China. His current research interests include large-scale multimedia retrieval, image/video segmentation, and image/video annotation using hashing, graph learning, and deep learning techniques.

Dr. Song is a Guest Editor of IEEE TRANSACTIONS ON MULTIMEDIA and the Area Chair of ACM Multimedia 2018.
Yuyu Guo is currently pursuing the master’s degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. He is currently involved in image/video understanding and image/video captioning.

Lianli Gao received the Ph.D. degree in information technology from The University of Queensland, Brisbane, QLD, Australia. She is currently an Associate Professor in computer science with the Future Media Center and the School of Information Technology and Electrical Engineering, University of Electronic Science and Technology of China, Chengdu, China. Her current research interests include machine learning, deep learning, and computer vision.

Alan Hanjalic (M’99–SM’08–F’16) is currently a Professor and the Head of the Multimedia Computing Group, Delft University of Technology, Delft, The Netherlands. His current research interests include multimedia search, recommender systems, and social media analytics.

Dr. Hanjalic was the Chair of the Steering Committee of the IEEE TRANSACTIONS ON MULTIMEDIA. He was a Keynote Speaker at the IEEE International Workshop on Multimedia Signal Processing 2013, the International Multimedia Modeling Conference 2012, and the Pacific-Rim Conference on Multimedia 2007. He has also been a General or Program (Co-)Chair of the organizing committees of multimedia conferences, such as ACM Multimedia, ACM International Conference on Content-Based Image and Video Retrieval/ACM International Conference on Multimedia Retrieval, and IEEE International Conference on Multimedia and Expo. He has been a member of the Editorial Board of several scientific journals in the multimedia field, including the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE TRANSACTIONS ON AFFECTIVE COMPUTING, the ACM Transactions on Multimedia, and the International Journal of Multimedia Information Retrieval. He is an Associate Editor-in-Chief of the IEEE Multimedia Magazine.

Xuelong Li (M’02–SM’07–F’12) is currently a Full Professor with the Xi’an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi’an, China.

Heng Tao Shen received the B.Sc. degree (Hons.) and the Ph.D. degree from the Department of Computer Science, National University of Singapore, Singapore, in 2000 and 2004, respectively. He then joined The University of Queensland, Brisbane, QLD, Australia, as a Lecturer, a Senior Lecturer, a Reader, and became a Professor in 2011. He is currently a Professor of the National Thousand Talents Plan, the Dean of the School of Computer Science and Engineering, and the Director of the Center for Future Media, University of Electronic Science and Technology of China, Chengdu, China. He is also an Honorary Professor with The University of Queensland. His current research interests include multimedia search, computer vision, artificial intelligence, and big data management. He has made continuous contributions to big data indexing and retrieval and developed the first real-time near-duplicate video retrieval system. He has published over 200 peer-reviewed papers, among which over 140 appeared in Chinese Computing Federation ranked publication venues, such as ACM Multimedia, IEEE Conference on Computer Vision and Pattern Recognition, International Conference on Computer Vision, AAAI Conference on Artificial Intelligence, International Joint Conference on Artificial Intelligence, The ACM Special Interest Group on Management of Data, International Conference on Very Large Data Bases, International Conference on Data Engineering, ACM Transactions on Information Systems, IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and VLDB Journal.