Array 9 (2021) 100052

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/journals/array/2590-0056/open-access-journal

Empirical autopsy of deep video captioning encoder-decoder architecture

Nayyer Aafaq [*], Naveed Akhtar, Wei Liu, Ajmal Mian

The University of Western Australia, 35 Stirling Hwy, 6009, WA, Australia

A R T I C L E I N F O

Keywords:
CNN architecture
Encoder-decoder
Word embeddings
Language model
Video captioning
Language and vision
Video to text
Natural language processing
Recurrent neural networks

A B S T R A C T

Contemporary deep learning based video captioning methods adopt encoder-decoder framework. In encoder, visual features are extracted with 2D/3D Convolutional Neural Networks (CNNs) and a transformed version of those features is passed to the decoder. The decoder uses word embeddings and a language model to map visual features to natural language captions. Due to its composite nature, the encoder-decoder pipeline provides the freedom of multiple choices for each of its components, *e.g.*, the choices of CNN models, feature transformations, word embeddings, and language models *etc*. Component selection can have drastic effects on the overall video captioning performance. However, current literature is void of any systematic investigation in this regard. This article fills this gap by providing the first thorough empirical analysis of the role that each major component plays in a widely adopted video captioning pipeline. We perform extensive experiments by varying the constituent components of the video captioning framework, and quantify the performance gains that are possible by mere component selection. We use the popular MSVD dataset as the test-bed, and demonstrate that substantial performance gains are possible by careful selection of the constituent components without major changes to the pipeline itself. These results are expected to provide guiding principles for research in the fast growing direction of video captioning.

1. Introduction

Recent years have seen rising research interests in automatic description of images and videos in natural language using deep learning techniques. Recent methods are inspired by the encoder-decoder framework used in machine translation [1]. These techniques use Convolutional Neural Networks (CNNs) as encoders to compute fixed/variable-length vector representations of the input images or videos. A Recurrent Neural Network (RNN), *e.g.*, vanilla RNN [2], Gated Recurrent Units (GRU) [3] or Long Short Term Memory (LSTM) networks [4] are then used as decoders to generate natural language descriptions.

In the encoder-decoder pipeline, visual features are extracted with 2D/3D CNNs from the input videos. These features are then transformed through Mean Pooling (MP) [5–7], Temporal Encoding (TE) [8], and/or Semantic Attributes Learning (SAL) [7,9,10] before feeding to a language model for natural language caption generation. Most of the existing captioning methods [11–16] mainly differ from each other in terms of the adopted visual feature extraction (*i.e.*, CNN models), types of visual feature transformations, language models, and the word embeddings incorporated in the language models. Despite the aforementioned differences, these four core components are common to nearly all techniques that follow the encoder-decoder framework for video captioning.

In Fig. 1, we take a modular decomposition approach and depict the encoder-decoder captioning framework in terms of its four core components. Multiple choices are available to instantiate each component. For instance, one can choose either a 2D-CNN or a 3D-CNN as the *CNN model*. Availability of numerous 2D/3D-CNN models provides further flexibility in the choices of the visual feature extraction component. The choice of models may directly affect the overall performance of the video captioning system. Similarly, the choice of MP or TE for *Feature Transformation* can also have significant effects on the system performance. This modular view of the encoder-decoder pipeline for video captioning is critical to assess how each component contributes to the caption qualities. For many existing methods, it is often unclear whether the performance gain is a result of some novel sophisticated enhancements, or simply due to better component selection. This calls for a systematic investigation to quantify performance gains against various component selections across the pipeline. Such analysis would establish a better understanding of the encoder-decoder framework and in turn help identify the most promising components and their best instances. Moreover, it can guide future research in video captioning to focus more

* Corresponding author.
E-mail address: nayyer.aafaq@research.uwa.edu.au (N. Aafaq).

https://doi.org/10.1016/j.array.2020.100052
Received 5 August 2020; Received in revised form 25 October 2020; Accepted 1 December 2020
Available online 29 December 2020

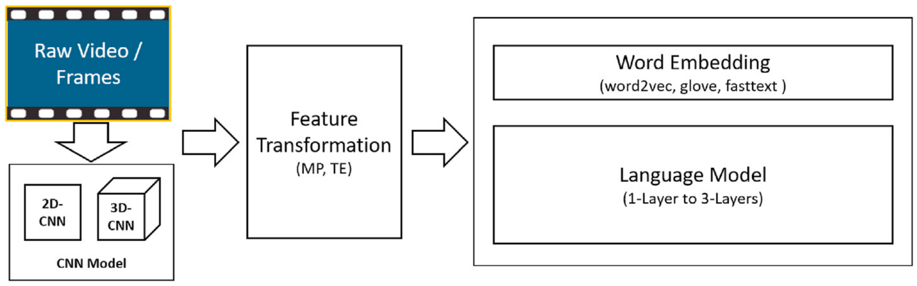

Fig. 1. The encoder-decoder framework of contemporary deep video captioning techniques has four major modules where selections can be made. It is possible to choose from a variety of 2D/3D-CNNs to encode visual features of videos. These features are then transformed to feed into the language model, which can be done by temporal encoding or mean pooling of the features. Multiple choices are also available for selecting the word embeddings that map words in a vocabulary to dense vector representations to be used by the language model. Language models can have different complexity, governed by *e.g.*, the number of network layers. In this paper, we vary the choices for each of the four major components, and analyze its effects on the overall captioning performance of the framework.

on improving the critical components of the framework.

In this work, we present the first systematic analysis of the encoder-decoder framework components with the aim of revealing the contribution of each component on the quality of the generated captions. Our analysis is performed by studying the effects of popular choices for each component while keeping the remaining components fixed. We also include the choices of important hyper-parameters in our analysis. The main contributions of this paper are as follows:

1. We evaluate and quantify the role of CNN architecture employed to extract features in the video captioning framework. We analyze 5 CNN models encompassing varying depths and structures. We observe that networks with stronger expressive ability perform better. We empirically demonstrate that the choice of CNN model in the video captioning framework can lead to performance gain up to 16.61% in METEOR (see Table 5 for complete details).
2. We analyze and quantify the performance gain achieved by employing temporal encoding over inferior mean pooling strategy to represent the whole video. It is further observed that temporal encoding applied on 2D-CNN, depending upon the technique, may even outperform most widely used C3D which is computationally much more expensive.
3. We further explore and quantify the effects of pre-trained word embeddings utilized in language models and report that *FastText* outperforms the contemporary popular word embeddings *e.g.*, *Word2Vec* [17] or *GloVe* [18].
4. Lastly, we analyze the effects of language model depth and various hyper-parameters choices *e.g.*, internal state size, number of frames, fine tuning the pre-trained word embeddings, and dropout regularization in video captioning framework.

2. Related work

2.1. Classical methods

Early works on video captioning [19–21] follow the template based approaches. These methods involve handcrafted features of limited pre-defined entities *i.e.*, subject, object, action, and place. The entities in a video are detected [22] separately by employing classical methods for each. Subsequently, to describe the detected components in natural language, most important entities are fit into a pre-defined sentence template. Typically, a sentence template comprises of three components namely lexicons, grammar and template rules. *Lexicon* represents vocabulary, *template rules* are user-defined rules to help select appropriate lexicons and *grammar* defines linguistic rules to ensure that generated sentence is syntactically correct. Due to advancement in deep neural networks, the encoder-decoder based framework was firstly introduced to overcome the limitations of classical approaches [5].

2.2. Encoder-decoder methods

Contemporary methods in video captioning rely on neural networks based framework with encoder-decoder architecture being the most popular [1,5,8,9,23–26]. The encoder (*i.e.*, 2D/3D-CNN), first encodes the video frames and decoder (*i.e.*, vanilla-RNN, GRU, LSTM), decodes the visual information into natural language sentences. These methods train the encoder and decoder into an end-to-end manner. This framework exploits the power of CNNs and RNNs in video representation and sequential learning respectively. Later methods, improve the performance by replacing the mean pooling with temporal encoding and introducing attention [8,9,27,28] in the encoder-decoder architecture.

Most recent methods incorporate various adjustments in the encoder-decoder framework. Wang et al. [29] proposed multi-model memory to model the long-term visual-textual dependency. Chen et al. [30] proposes a frame picking module to select a compact frame subset to represent the video. This enables the encoder-decoder architecture more applicable to process the real world videos. GRU-EVE [25] employs Short Fourier Transform on the encoder output to enrich the video representation with temporal information. Zhang et al. [31] models salient objects with their temporal dynamics to improve the architecture performance. Zheng et al. [32] propose syntax aware action targetting module to explicitly learn actions. In order to understand the strength of each technique employing encoder-decoder architecture, it is important to understand the role of each module in encoder-decoder. For that matter, we put together the performance contribution of each module, forming the basis of encoder-decoder architecture, towards overall framework performance.

3. Setup for analysis

3.1. Components

We first introduce the setup used in our empirical analysis of the video captioning framework. For evaluation, we divide the framework into four core components, namely *CNN model* - that encodes visual features of videos, *feature transformation* - that transforms visual features to be used as inputs by the language model component, *word embeddings* - that provides numerical representation of words in the vocabulary, and the *language model* component, which decodes the visual features into natural language descriptions. Extensive experiments are carried out by varying the methods for each component of the framework and analyze the captioning performance of the overall pipeline.

3.2. Evaluation metrics

We measure the performance in terms of most commonly used evaluation metrics in the contemporary captioning literature, namely Bilingual Evaluation Understudy (BLEU) [33], Recall Oriented Understudy for Gisting Evaluation (ROUGE) [34], Metric for Evaluation of Translation with Explicit Ordering (METEOR) [35], and Consensus based Image

Description Evaluation (CIDEr) [36]. These metrics are known to comprehensively evaluate the quality of automatically generated captions. We briefly discuss each metric in the below text. For details on each metric, pros and cons, and for their comparisons, we refer the readers to original papers and survey paper [37].

BLEU measures *n–gram* based exact matches of the words as well as their order in the reference and generated sentences. Hence, with higher number of reference sentences BLEU is more likely to score high. Therefore, more realistic usage of BLEU would be at a corpus level. Formally BLEU is computed as;

$$\log(\mathrm{BLEU}) = \min\left(1-\frac{l_{gt}}{l_c}, 0\right) + \sum_{k=1}^{K} w_k \log p_k,$$

where l_{gt}/l_c represent the ratio between the lengths of ground truth and the candidate descriptions, w_k are positive weights, and p_k is the geometric average of the modified *n–gram* precisions. The first term in above equation computes brevity penalty that penalizes descriptions shorter than the ground truth description (refer to original paper for further details).

ROUGE$_L$ computes the recall score of the generated sentences using *n-â€"grams*. It computes the common subsequences between the candidate and reference sentences.Unlike, words match may not be consecutive however should be in sequence.

METEOR addresses many of the BLEU shortcomings. For example, instead of exact word match, it incorporates synonym matching and performs semantic matching. It is also found to be more robust and closely correlated to human judgments [38]. To compute the METEOR score, first *uni–gram* based recall R and precision P scores are computed as;

$$R = m_{cg}/m_{gt}, P = m_{cg}/m_{ct},$$

where m_{cg}, in both equations, refer to the number of *uni–grams* co-occurring in both candidate and ground truth sentences, m_{gt} is the total number of *uni–grams* in the ground truth sentences and m_{ct} corresponds to total number of *uni–grams* in the candidate sentences. METEOR score for a sentence is computed using F-score and penalty p such as;

$$F_{score} = \frac{10PR}{R+9P}; \quad p = \frac{1}{2}\left(\frac{N_{ch}}{N_{uni}}\right)^2; \quad M = F_{score}(1-p),$$

where N_{ch} and N_{uni} represents the number of chunks and number of *uni–grams* grouped together respectively. N_{ch} includes adjacent *uni–grams* in candidate as well as reference sentences. The ratio $\frac{N_{ch}}{N_{uni}}$ will be 1 in case of generated sentence is an exact match to the reference sentence. In order to compute corpus level score, aggregated values of the constituent components *i.e.*, P, R, and p are taken. Moreover, highest METEOR score is selected if there are multiple reference sentences against a generated sentence.

Lastly, CIDEr$_D$ evaluates the consensus between a generated and reference sentences. For that matter, it first performs stemming to all the candidate and reference words followed by measuring the *n–gram* based co-existence frequency using Term Frequency Inverse Document Frequency (TF-IDF) [39]. score is computed as;

$$\mathrm{CIDEr}_n(g_i, S_i) = \frac{1}{m}\sum_j \frac{\kappa^n(g_i).\kappa^n(s_{ij})}{||\kappa^n(g_i)||.||\kappa^n(s_{ij})||},$$

where $\kappa^n(g_i)$ represents *n–grams* of length n, $\kappa^n(g_i)$ denote magnitude of $\kappa^n(g_i)$. Same is the case for $\kappa^n(s_{ij})$. CIDEr uses higher order *n–grams* to capture the grammatical properties and semantics of the text. It combines scores of different *n–grams* using the following equation:

$$\mathrm{CIDEr}(g_i, S_i) = \sum_{n=1}^{N} w_n \mathrm{CIDEr}_n(g_i, S_i).$$

With slight modifications in the original metric, CIDEr-D is the most popular variant in the image and video captioning evaluation. Firstly, it removes the stemming to ensure correct form of words are used. Secondly, by clipping *n–grams* count, it ensures high score is not produced by repeating high confidence words even if the sentence does not make sense.

We used the Microsoft COCO server [40] to compute our results. To clearly analyze the contribution of each component in the overall pipeline, we follow the strategy of freezing all other components when evaluating a particular module. The metric has also been found to be more robust to distractions *e.g.*, scene or person changes [38].

3.3. Dataset

We perform experiments on the popular video captioning dataset MSVD [41]. This dataset comprises 1,970 YouTube short video clips, primarily containing single action/event in a video. Each clip duration varies from 10 to 25 s. Each video is associated with multiple human annotated captions. On average, there are 41 captions per video clip. For bench-marking, we follow the data split of 1,200, 100, and 670 videos for training, validation and testing respectively. This is a widely employed protocol for evaluation using MSVD dataset [8,9,29].

3.4. Training details

In our experiments, we employ 5 popular CNN architectures (*i.e.*, VGG16, VGG19, Inception-v3, InceptionResNet-v2, and C3D) in the encoder to extract the visual features. The last '*fc*2' and '*avgpool*' layer of the 2D-CNNs and '*fc*6' layer of 3D-CNN are considered as the extraction layers. All the 2D-CNNs are pretrained on ImageNet [42] whereas 3D-CNN on sports 1 M dataset [43]. For the 2D-CNN, we process all frames and for 3D-CNN 16 frame clips as inputs with an 8 frame overlap are selected. For decoder training, we add '*start*' and '*end*' tokens to deal with the dynamic length of the captions. We set maximum length of sentence to 30 words and truncate/zero pad in case of length of the sentence exceeds or fall short respectively. We employ RMSProp algorithm and set the learning rate 2×10^{-4} to train the models. We select the batch size of 60 and train the models for 50 epochs. We use sparse cross entropy loss to train our model. We use NVIDIA Titan XP 1080 GPU in our experiments. TensorFlow framework is used for the development.

4. Analysis of framework components

4.1. Preliminary

In this section we briefly describe the vanilla encoder-decoder architecture which is backbone of most video captioning methods. Encoder encodes the input video frames by employing pre-trained convolutional neural network(s). The extracted features from encoder are then utilized as input to RNN decoder to generate the caption. Formally, for a given video

$$V = \{f_1, f_2, \ldots, f_N\},$$

where N represents the number of frames in video V. To generate the features for all frames from video V, deep Convolutional Neural Networks (CNNs) pretrained for object recognition and detection are employed. These deep CNN models generate a feature representation for each frame f. Formally:

$$F_V = \varphi(V) = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_N\},$$

where φ denote pretrained CNN model and $\mathbf{f}_i \in \mathbb{R}^m$ represent the feature vector for i^{th} frame. The visual representation F_V may further be enriched by employing additional modalities such as attention, temporal modeling, semantic attributes, audio and last but not the least objects and action modeling.

$$\mathbf{F}_{encd} = \Psi(F_V),$$

where Ψ represents the additional modalities applied and $\mathbf{F}_{encd}$ denote the encoder output. The encoder output is utilized to initialize the initial hidden state of the language model. The word at time step t is generated based on the previous hidden state and sequence of words generated till time step t. This process continues until it gets the end token of the sentence. Formally, the decoder takes $\mathbf{F}_{encd}$ as input and outputs the sentence (*i.e.*, word sequence) represented as:

$$S = \{w_1, w_2, \dots, w_T\},$$

where the decoder Φ, *i.e.* a language model in this case, generates word distribution w_t at any time step:

$$w_t = \Phi(\mathbf{F}_{encd} | (w_1, w_2, \dots, w_{t-1})).$$

4.2. CNN selection

Convolutional Neural Networks (CNNs) can be readily applied to images and videos. In deep learning based encoder-decoder framework for captioning, CNNs dominate the encoder part. Due to the significance of a encoder role, the choice of CNN models can affect the overall captioning performance significantly. Hence, we first analyze the five most commonly used CNN models in captioning, namely; C3D [44], VGG-16 [45], VGG-19 [45], Inception-v3 [46], and InceptionResNet-v2 [47]. Among these models, C3D - a popular example of 3D-CNN, is a common choice [23,48] because it can not only process individual frames, but also short video clips. This is possible due to its ability to process tensors with an extra time dimension.

While performing these experiments to compare different CNN models, we fix all components of the pipeline, except the visual features. For the remaining components, the popular Mean Pooling is used to transform the extracted visual feature into a fixed length vector to represent a complete video; word2vec word embeddings are used for a $2-$layer GRU as the language model. The results of this set of experiments are summarized in Fig. 2. We can see a significant variation in captioning performance due to changes in the CNN models, ascertaining that better visual features (obtained from more sophisticated models) lead to better video captions. Hence, a careful selection of CNN model is critical for effective video captioning. Interestingly, the spatial visual features of 2D-CNN (*VGG16, VGG19, Inc-V3,* and *Inc-Res-V2*) are able to outperform the spatio-temporal features of *C3D* for the video captioning task, indicating that the extra dimension of 3D-CNNs may not be particularly effective in this case.

Moreover, we observe that the effect of CNN architecture in the video captioning framework follows the similar trend as seen in image classification tasks. It is generally believed that deeper networks tend to perform better, we see that VGG-Nets from 16 to 19 layers slightly improve the performance of captioning framework. However, after a certain depth is reached in the CNN, increase in number of layers do not increase the classification accuracy of the model. The problem was resolved by introducing residual blocks that resulted in networks with stronger expressive ability and hence performed better in classification tasks. Similarly, we see the same trend in our experiments where we achieve better performance by employing these networks in the video captioning framework.

4.3. Features transformation

Most existing video captioning methods [5–7,9] perform mean pooling to combine individual frame features into a feature vector for the whole video. However, this practice is bound to inferior performance as mean pooling can result in significant loss of temporal information and the order of events in videos. Such information often plays a crucial role in video understanding for humans. Inspired by this observation, we conduct another series of experiments that compares a temporal encoding strategy to the mean pooling strategy for feature transformation in video captioning.

For temporal encoding, we follow our previous work [25] and compute Short Fourier Transform [49] of the frame level features of the video. These features are combined in a hierarchical manner that captures the local, intermediate and high level temporal dynamics in the video. Interested readers are referred to our work [25] for exact details of the temporal feature transformation. The core insight relevant to our analysis here is that instead of compromising on the temporal information through mean pooling, we capture high fidelity temporal dynamics over whole videos with our temporal encoding. Similar to the other sections of this paper, we fix all the remaining components of the underlying framework when analyzing the transformation strategy. The results of our experiments are summarized in Fig. 3. It is evident that the models employing temporal encoded features, have outperformed all of

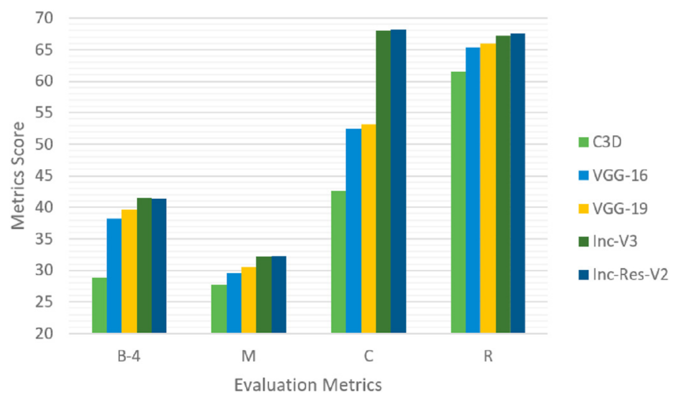

Fig. 2. Performance of five 2D/3D CNNs architectures (C3D, VGG-16, VGG-19, Inception-V3 (Inv-V3), and Inception-ResNet-V2 (Inv-Res-V2)) when used as visual encoder in the captioning framework. Results are achieved by using Mean Pooling for feature transformation, word2vec as word embedding, and a 2-layer GRU as the language model.

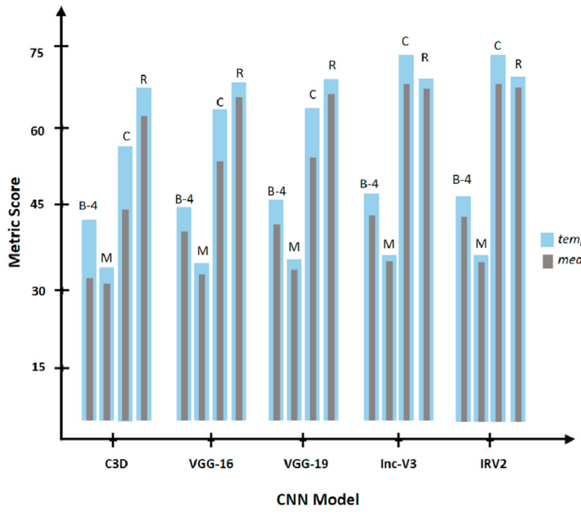

Fig. 3. Performance comparison of five 2D/3D CNNs models with two types of feature transformations *i.e.*, Mean Pooling and Temporally Encoding. It is evident that under all circumstances, temporally encoded features outperform the mean pooled features across all networks and among all metrics.

their mean pooling based counterparts across all evaluation metrics. Considering the widespread use of mean pooling as the feature transformation strategy in video description, these results are significant. This experiment clearly establishes the supremacy of temporal encoding over mean pooling for the encoders in video description. This temporal encoding does not require any training, as it is applied after the features are extracted, therefore little computational overhead is introduced.

It is also evident from the results that for temporal encoding, the performance of different models show a similar behavior relative to each other, which is also the case for the mean pooled features. For instance, with temporally encoded features, the best performing architecture still remains the best and vice versa is also true. The temporal encoding is providing a significant positive offset to the performance.

4.4. Word embeddings in language model

In this encoder-decoder framework, a word embedding is a vector representation for each word in the available vocabulary for video caption generation. Word embeddings are much more powerful low-dimensional representations for words as compared to the sparse one-hot vectors. More importantly, unlike one-hot vectors, word embeddings can be learned for the captioning tasks. In captioning literature, two methods are commonly used to compute these vectors. The first approach is to learn the vectors from the training dataset while the language model is trained. In this case, one can initialize the embedding vectors randomly and compute the embeddings tailored to the captioning task. However, such vectors often fail to capture rich semantics due to the fact that captioning corpus size is often small for the purpose of training a language model. The second way to obtain these vectors is to use pre-trained embeddings that are learned for a different task and select those according to the vocabulary of the current task.

We follow both of the aforementioned methods to compute embeddings in our analysis. For the first method, random initialization is performed. For the second, we obtain the most commonly used four pre-trained word embeddings in the contemporary video description literature, namely Word2Vec [17], two variants of Glove (*i.e.*, `glove6B` and `glove840B`) [18] and FastText [50]. In our analysis, we select the best performing CNN model from our experiments in Section 4.3 that uses temporal encoding for feature transformation, *i.e.*, Inception-ResNet-V2. The results of our experiments for word embeddings are summarized in Fig. 4. From the figure, we can conclude that FastText currently provides much more effective word embeddings for video captioning than the other techniques. Moreover, learning the embeddings with random initialization is still a useful option for the MSVD dataset. This is true to the extent that `glove6B` shows comparable performance to our randomly initialized learned word embeddings.

We attribute the superior performance of FastText to its ability to generate vectors for out of vocabulary words according to the contextual

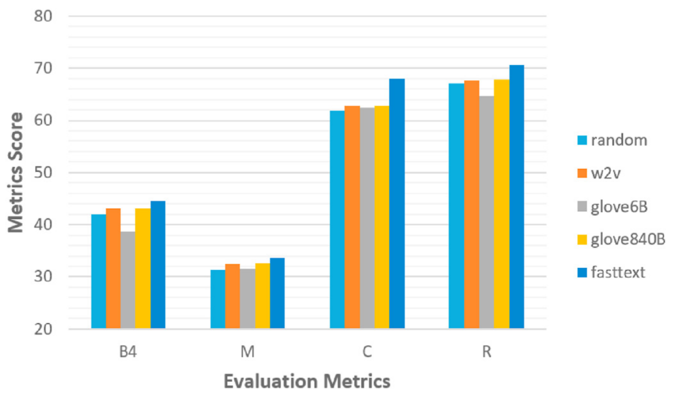

Fig. 4. Performance of four popular pre-trained word embeddings and the learned embedding with random initialization.

vectors. The other word embeddings do not have this property. For instance, with 9,914 words of corpus vocabulary size in FastText, 8,846 tokens are extracted from the pre-trained embeddings and the embeddings for the remaining 1,068 tokens are generated using character n-grams of out-of-vocabulary words. The resulting vectors are then merged to produce the final embedding vector. This strategy is certainly better than random initialization of the out-of-vocabulary words. With FastText at the top, `glove840B` and Word2Vec performs almost at par. Among all the pre-trained embeddings, `glove6B` proved to be the weakest.

4.5. Language model depth selection

In language models, given the type and size of data, depth of the model plays the pivotal role in effective learning. Where lower layers of a model learn to represent the *syntactic information* (parts of speech, grammatical role of words in each sentence etc.), *semantic information* (meaning of the words, contextual information) is better captured at the higher layers. As each layer learns different type of information, depth of models becomes important for effective language modelling. However, the modelling performance may start to deteriorate at a certain depth due to the data size limitation.

In our experiments, Gated Recurrent Units (GRUs) based language models are used. Long Short Term Memory (LSTM) networks are also popular for language modeling, however, there is a consensus in the literature that the performance and behavior of the two do not deviate significantly for the same task [51,52]. In our analysis, we vary the number of layers in the language model to observe the performance change. Our empirical evaluation shows that two-layers language model performs best under our settings (type and size of dataset, encoder-decoder framework), as compared to a one or a three layer model. The results are summarized in Table 1. Increasing layers from one to two generally improves the captioning performance. However, increasing the layers further to three does not result in performance gain. In fact, it slightly deteriorates the scores across all metrics. The experiments are performed using visual features of Inception-ResNet-V2 that are transformed with temporal encoding [25] for the language model.

4.6. Hyperparameter settings

Appropriate hyper-parameter setting and model fine-tuning are well-known for their role in achieving the improved performance with deep networks. Here, we provide a study of a few important hyper-parameters relevant to the captioning task under the encoder-decoder framework. The reported results and findings can serve as guidelines for the community for training effective captioning models.

State Size: In the language model, deciding a suitable state size is critical. We tested captioning performance for various state sizes, *i.e.*, 512, 1024, 2048, and 4096. These results are reported in Table 2 and Fig. 5. We find a direct relation between the state size and the model performance. We compute Pearson's correlation between state sizes and each metric as shown in Table 4. It is evident from the results that there is significant correlation between state size and all the metrics. The relationship is even stronger in the lower n–grams of BLEU metric. It can be observed that the model performance enhances gradually when we change the state size from 512 until 2048. Further increase in the state size results in negligible or no improvement in the performance of the BLEU-4, METEOR, CIDEr, and ROUGE$_L$ metrics. However, lower *n–grams* of BLEU metric (B1, B2, B3) show slight improvements in such cases.

Table 1
Results of depth variation in GRU-based language model.

Model Depth	B-4	M	C	R
1 layer	49.6	34.9	75.8	71.3
2 layers	47.9	35.0	78.1	71.5
3 layers	47.7	34.6	77.4	70.8

Table 2
Results on the state size choices of the GRU language model.

State Size	B1	B2	B3	B4	M	C	R
4096	77.3	62.9	51.6	40.7	30.8	59.2	66.7
2048	77.1	62.8	51.5	41.0	31.3	61.9	67.6
1024	74.9	59.4	47.5	36.6	30.0	55.9	65.6
512	71.8	54.9	42.8	32.0	28.4	46.5	62.4

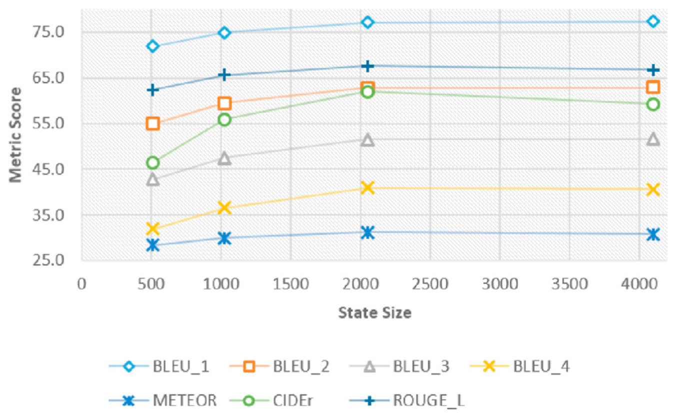

Fig. 5. Performance evaluation of language model using four state sizes. Each trend line show each metric used to compute the captions score.

Table 3
Results on number of frames vs captioning quality.

# Frames	B1	B2	B3	B4	M	C	R
16–F	65.4	46.7	34.0	23.0	24.5	31.6	57.2
All–F	69.6	52.1	39.6	28.8	27.7	42.6	61.6

Number of Frames: Frame selection can be treated as a function of time for any video. Though smaller number of frames reduces the computation cost of the model however it potentially loses some important spatio-temporal information. In this set of experiments, we first process significantly low number of frames, *i.e.*, using every sixteenth (16^{th}) frame in the video. In the next experiment, we process all frames of the video. A significant gain in model performance is observed in the case where all the frames are processed. These experiments are performed using C3D model (Tran et al., 2015) for visual encoding. The two experiments follows the same settings except in terms of the number of frames used. The results of these experiments are shown in Table 3. As evident from the results, a significant improvement in model performance across all metrics can be observed when all frames are used for captioning.

Fine Tuning Pre-Trained Word Embeddings: It is also observed in our experiments that using pre-trained word embeddings often outperform random initialization based learned embeddings.

We also experimented by fine tuning the model for 10 epochs on the pre-trained word embeddings. It was observed that in this case, the performance on BLEU and CIDEr metrics improved slightly with the fine tuning. However, performance on ROUGE$_L$ metric remained negligible. METEOR metric value showed mixed behaviour with no regular patterns.

Dropout in Recurrent Layers: Dropout is a technique used in neural networks to prevent overfitting of the model during training. In recurrent networks *e.g.*, in GRU, input and recurrent connections to GRU units are probabilistically excluded from activation and weight updates while training the network. Using dropout is typically effective for training language models with large dataset. However, with the MSVD dataset, the caption corpus is rather small (~48K captions with ~9K unique tokens), dropout therefore does not have a significant effect on language model performance for this dataset, or the datasets of similar scale. We employed dropout in the recurrent layers of language model. However, it was observed that application of dropout did not improve the performance. In fact, it sometimes resulted in slight deterioration of the model performance. Based on the observed behavior, we can confidently recommend to avoid the use of recurrent dropout in a GRU language model, given the training data of MSVD size (or comparable) and model complexity of 2 – 3 GRU layers.

5. System level discussion and analysis

With Section 4 focusing on 'ablation study' of individual components, in this section, we further discuss and analyze the results of the pipeline as a whole, at the system level. First, we discuss the results in terms of Min – Max improvements in captioning score for each metric, respectively, as shown in Table 5. Here, 'Min' denotes the minimum percentage gain in the performance which is computed as the difference between the lowest score and the second lowest score in our experiments. 'Max' denotes the percentage gain achieved by comparing the lowest and the highest values achieved. The Min – Max ranges provide an estimate of the performance gain that is possible by varying the selection of component variants.

In Table 5, first two rows depict improvements by selecting superior or inferior CNNs (in terms of their original results on ImageNet classification accuracies). These results are obtained using mean pooling strategy over the frame level features of the corresponding networks. When compared across 2D/3D CNNs (first row) we see a drastic obtainable improvement in the model performance *i.e.*, up to 44 % in BLEU and 60 % for CIDEr metric, if we choose the right visual feature encoding model. Similarly, when comparing among 2D CNNs only (second row), we see there are significant performance variations. These variations only resulted from varying the CNN model. Hence, we can conclusively argue that superior CNNs (with better representation power) can result in significant performance improvement for the captioning techniques.

The evaluation results for the word vector representations are shown in row 3 of Table 5. We can observe that there are a few instances of significant performance variations across all metrics when we use different word embeddings. Among the used popular embeddings, FastText performs the best and glove6B the weakest. Note that the results also include the learned word vectors obtained during language model

Table 5
Percentage improvements (Min – Max) achievable with careful selection of components. First row compared performance including 2D/3D networks. Row 2 demonstrates performance variation with 2D networks only. Subsequent rows show improvements due to word embeddings and the depth of language model.

	B-4 (%)	M (%)	C (%)	R (%)
CNN (2D/3D)	32.64–44.10	6.86–16.61	23.0–60.09	6.17–9.74
CNN (2D only)	3.93–8.64	3.38–9.12	1.53–30.15	0.92–3.36
Word Vectors	8.27–14.99	0.64–7.03	0.81–9.69	0.60–5.37
Depth of Language Model	0.42–3.98	0.87–1.16	2.11–3.03	0.71–0.99

Table 4
Results of Pearson's correlation of state sizes with each metric.

	B1	B2	B3	B4	M	C	R
Pearson's Correlation	0.8110	0.8028	0.8107	0.7947	0.6897	0.6635	0.6714
p-Value	0.1889	0.1972	0.1893	0.2053	0.3103	0.3365	0.3286

Table 6
Percentage improvement achieved in BLEU (B-4), METEOR (M), CIDEr (C), and ROUGE$_L$ (R) metrics when mean pooled visual features are replaced with temporally encoded features of corresponding networks.

	B-4 (%)	M (%)	C (%)	R (%)
C3D	40.97	11.91	30.75	9.42
VGG-16	13.09	7.77	20.04	4.59
VGG-19	12.59	6.86	18.80	4.70
Inception-V3	10.84	4.04	8.82	2.98
Inception-ResNet-V2	10.14	4.33	8.80	3.25

training with random initialization. Moreover, we also experimented with fine tuning of the pre-trained embeddings for 10 epochs for the captioning task. However, we observed that fine tuning does not result in any drastic performance gain. We noticed that the performance of word2vec and glove840B mostly remain at par with each other. Compared to the visual feature encoder selection, we can see the performance gain by the informed selection of word embeddings are not negligible either. However, the right CNN model does have a dominant effect on the performance gain as compared to the word embedding selection.

The last row of Table 5 provides language model depth analysis. Relative to the gain obtainable by varying other components in the pipeline, altering the depth from *1-layer* to *3-layers*, does not boost the performance significantly. The metric scores generally improve when model depth is varied from $1 - \text{to} - 2$ *layers*. However, further increase in the depth degrades the model performance. We can confidently claim that under the employed popular pipeline, *2-layers* GRU network performs better as compared to the single or three layers RNNs.

In Table 6, we report the maximum percentage gains caused by the feature transformation techniques in our experiments. Each row reports the metric score improvement resulted when mean pooled features are replaced with the temporal encoding features [25] of same CNN network. As can be observed, there is significant improvement in the model performance across all metrics and all networks with the temporal encoding. The largest performance gain results in the case of C3D. We conjecture that a major reason behind this phenomenon is that there are always less number of clips as compared to the number of frames in videos. C3D exploits clips which reduces the number of unit data samples containing distinct pieces of information for the task at hands. The temporal encoding strategy is able make up for this discrepancy. Moreover, spatial feature capturing with a 2D-CNN followed by temporal encoding result in more discriminative video-level features as compared to the spatio-temporal features of 3D-CNNs.

Based on the evaluations performed with all components of the captioning framework, we can order the components in terms of their importance/contribution to the overall captioning performance. To that end, in our experiments, the most significant contribution comes from the feature transformation technique *i.e.*, Temporal Encoding. The second significant performance variation is possible through the selection of appropriate CNN model. At the third position in terms of contribution to captioning quality, we can place the word embedding vectors. The number of network layers in captioning model had less significant role to play in our experimental results. A simple 2-layer GRU seems a reasonable baseline choice for the captioning models. Similar to the network layers, other hyper-parameters choices also contribute to the captioning performance, as mentioned in Sec. 4.6. However, assuming a reasonable default hyper-parameter settings, their role is far less significant than the variation in the major components of the pipeline.

6. Conclusion

In this paper, we empirically evaluate and quantify the performance contribution of encoder-decoder architecture components towards video captioning framework. For that matter we decompose the architecture into four core components namely, CNN architecture (*i.e.*, employed for visual feature extraction), features transformation technique (*e.g.*, mean pooling, temporal encoding), word embeddings (utilized in language model), and language model itself. This allows us to carry out a comprehensive and fair ablation study at both the component level and the system level on a most popular MSVD dataset. Various model hyper-parameters (*e.g.*, depth of language model, it's internal state size, and dropout in recurrent layers) are also included in our empirical study. Exhaustive experiments are carried out to capture and quantify the performance gain by each constituent component of the overall video captioning framework. In particular, we use 5 popular CNNs (*i.e.*, C3D, VGG-16, VGG-19, Inception-V3, and Inc-ResNet-V2), 2 feature transformation algorithms (*i.e.*, mean pooling, temporal encoding), 5 pre-trained word embeddings (*i.e.*, learned, word2vec, glove6B, glove840B, and fasttext) and an RNN language model with depth of 1, 2, and 3 layers. It is demonstrated that with a well-informed selection of the components in the encoder-decoder based video captioning framework, a significant performance gain can be achieved. In our experiments, the best performing framework comprises Inception-ResNet-V2 as the visual encoder, followed by temporal encoding for feature transformation, fasttext word embeddings and a 2 layers language model.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work is supported by Australian Research Council Grant ARC DP19010244. The GPU used for this work was donated by NVIDIA Corporation.

References

[1] Donahue J, Anne Hendricks L, Guadarrama S, Rohrbach M, Venugopalan S, Saenko K, Darrell T. Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2015. p. 2625–34.
[2] Elman JL. Finding structure in time. Cognit Sci 1990;14:179–211.
[3] Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. 2014. arXiv preprint arXiv:1406.1078.
[4] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9:1735–80.
[5] Venugopalan S, Xu H, Donahue J, Rohrbach M, Mooney R, Saenko K. Translating videos to natural language using deep recurrent neural networks. 2014. arXiv preprint arXiv:1412.4729.
[6] Pan Y, Mei T, Yao T, Li H, Rui Y. Jointly modeling embedding and translation to bridge video and language. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 4594–602.
[7] Pan Y, Yao T, Li H, Mei T. Video captioning with transferred semantic attributes. In: IEEE CVPR; 2017.
[8] Yao L, Torabi A, Cho K, Ballas N, Pal C, Larochelle H, Courville A. Describing videos by exploiting temporal structure. In: Proceedings of the IEEE international conference on computer vision; 2015. p. 4507–15.
[9] Gan Z, Gan C, He X, Pu Y, Tran K, Gao J, Carin L, Deng L. Semantic compositional networks for visual captioning. In: IEEE CVPR; 2017.
[10] Yao X, Han J, Cheng G, Qian X, Guo L. Semantic annotation of high-resolution satellite images via weakly supervised learning. IEEE Trans Geosci Rem Sens 2016;54:3660–71.
[11] Pei W, Zhang J, Wang X, Ke L, Shen X, Tai YW. Memory-attended recurrent network for video captioning. In: The IEEE conference on computer vision and pattern recognition (CVPR); 2019.
[12] Pan B, Cai H, Huang DA, Lee KH, Gaidon A, Adeli E, Niebles JC. Spatio-temporal graph for video captioning with knowledge distillation. 2020. arXiv preprint arXiv:2003.13942.
[13] Yan C, Tu Y, Wang X, Zhang Y, Hao X, Zhang Y, Dai Q. Stat: spatial-temporal attention mechanism for video captioning. In: IEEE transactions on multimedia; 2019.
[14] Liu S, Ren Z, Yuan J. Sibnet: sibling convolutional encoder for video captioning. In: IEEE transactions on pattern analysis and machine intelligence; 2020.

[15] Wang X, Chen W, Wu J, Wang YF, Yang Wang W. Video captioning via hierarchical reinforcement learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2018. p. 4213–22.

[16] Wang B, Ma L, Zhang W, Liu W. Reconstruction network for video captioning. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2018. p. 7622–31.

[17] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems; 2013. p. 3111–9.

[18] Pennington J, Socher R, Manning C. Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing. EMNLP); 2014. p. 1532–43.

[19] Kojima A, Tamura T, Fukunaga K. Natural language description of human activities from video images based on concept hierarchy of actions. IJCV 2002;50:171–84.

[20] Das P, Xu C, Doell RF, Corso JJ. A thousand frames in just a few words: lingual description of videos through latent topics and sparse object stitching. In: IEEE CVPR; 2013. p. 2634–41.

[21] Krishnamoorthy N, Malkarnenkar G, Mooney RJ, Saenko K, Guadarrama S. Generating natural-language video descriptions using text-mined knowledge. In: AAAI; 2013. p. 2.

[22] Yao X, Han J, Zhang D, Nie F. Revisiting co-saliency detection: a novel approach based on two-stage multi-view spectral rotation co-clustering. IEEE Trans Image Process 2017;26:3196–209.

[23] Yu H, Wang J, Huang Z, Yang Y, Xu W. Video paragraph captioning using hierarchical recurrent neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 4584–93.

[24] Cheng G, Yang C, Yao X, Guo L, Han J. When deep learning meets metric learning: remote sensing image scene classification via learning discriminative cnns. IEEE Trans Geosci Rem Sens 2018;56:2811–21.

[25] Aafaq N, Akhtar N, Liu W, Gilani SZ, Mian A. Spatio-temporal dynamics and semantic attribute enriched visual encoding for video captioning. In: IEEE CVPR; 2019. URL: http://arxiv.org/abs/1902.10322.

[26] Park JS, Darrell T, Rohrbach A. Identity-aware multi-sentence video description. In: Proceedings of the ECCV; 2020.

[27] Li X, Zhao B, Lu X, et al. Mam-rnn: multi-level attention model based rnn for video captioning. In: IJCAI; 2017. p. 2208–14.

[28] Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, Zemel R, Bengio Y. Show, attend and tell: neural image caption generation with visual attention. In: International conference on machine learning; 2015. p. 2048–57.

[29] Wang J, Wang W, Huang Y, Wang L, Tan T. M3: multimodal memory modelling for video captioning. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2018. p. 7512–20.

[30] Chen Y, Wang S, Zhang W, Huang Q. Less is more: picking informative frames for video captioning. In: Proceedings of the European conference on computer vision. ECCV); 2018. p. 358–73.

[31] Zhang J, Peng Y. Object-aware aggregation with bidirectional temporal graph for video captioning. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2019. p. 8327–36.

[32] Zheng Q, Wang C, Tao D. Syntax-aware action targeting for video captioning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2020. p. 13096–105.

[33] Papineni K, Roukos S, Ward T, Zhu WJ. Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on ACL; 2002. p. 311–8.

[34] Lin CY. Rouge: a package for automatic evaluation of summaries. In: Text summarization branches out: proceedings of the ACL-04 workshop; 2004 [Barcelona, Spain].

[35] Banerjee S, Lavie A. Meteor: an automatic metric for mt evaluation with improved correlation with human judgments. In: Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization; 2005. p. 65–72.

[36] Vedantam R, Lawrence Zitnick C, Parikh D. Cider: consensus-based image description evaluation. In: IEEE CVPR; 2015.

[37] Aafaq N, Mian A, Liu W, Gilani SZ, Shah M. Video description: a survey of methods, datasets, and evaluation metrics. ACM Comput Surv 2019b;52:115.

[38] Kilickaya M, Erdem A, Ikizler-Cinbis N, Erdem E. Re-evaluating automatic metrics for image captioning. 2016. arXiv preprint arXiv:1612.07600.

[39] Robertson S. Understanding inverse document frequency: on theoretical arguments for idf. J Doc Oct 2004;60:503–20.

[40] Chen X, Fang H, Lin TY, Vedantam R, Gupta S, Dollár P, Zitnick CL. Microsoft coco captions: data collection and evaluation server. 2015. arXiv preprint arXiv: 1504.00325.

[41] Chen DL, Dolan WB. Collecting highly parallel data for paraphrase evaluation. In: ACL: human language technologies-volume 1. ACL; 2011. p. 190–200.

[42] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, et al. Imagenet large scale visual recognition challenge. Int J Comput Vis 2015;115:211–52.

[43] Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Fei-Fei L. Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2014. p. 1725–32.

[44] Tran D, Bourdev L, Fergus R, Torresani L, Paluri M. Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the IEEE international conference on computer vision; 2015. p. 4489–97.

[45] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: ICLR; 2015.

[46] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 2818–26.

[47] Szegedy C, Ioffe S, Vanhoucke V, Alemi AA. Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-first AAAI conference on artificial intelligence; 2017.

[48] Krishna R, Hata K, Ren F, Fei-Fei L, Carlos Niebles J. Dense-captioning events in videos. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 706–15.

[49] Oppenheim AV. Discrete-time signal processing. Pearson Education India; 1999.

[50] Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information. TACL 2017:135–46.

[51] Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. 2014. arXiv preprint arXiv:1412.3555.

[52] Jozefowicz R, Zaremba W, Sutskever I. An empirical exploration of recurrent network architectures. In: International conference on machine learning; 2015. p. 2342–50.