Delving Deeper into Convolution Networks for Learning Video Representation

Nicolas Ballas[1], Li Yao[1] Chris Pal[1,2] Aaron Courville[1]
[1]MILA, Université de Montréal. [2] Polytechnic Montréal.

1 Introduction

Video analysis and understanding represents a major challenge for computer vision and machine learning research. While previous work has traditionally relied on hand-crafted and task-specific representations, there is a growing interest in designing general video representations that could help solve tasks in video understanding such as human action recognition, video retrieval or video captionning [12].

Two-dimensional Convolutional Neural Networks (CNN) have exhibited state-of-art performance in still image tasks such as classification or detection. However, such models discard temporal information that has been shown to provide important cues in videos [9]. On the other hand, recurrent neural networks (RNN) have demonstrated the ability to understand temporal sequences in various learning tasks such as speech recognition [7] or machine translation [1]. Consequently, Recurrent Convolution Networks (RCN) [6] that leverage both recurrence and convolution have recently been introduced for learning video representation. Such approaches typically extract "visual percepts" by applying a 2D CNN on the video frames and then feed the CNN activations to an RNN in order to characterize the video temporal variation.

Previous works on RCNs has tended to focus on high-level visual percepts extracted from the 2D CNN top-layers. CNNs, however, hierarchically build-up spatial invariance through pooling layers. While CNNs tends to discard local information in their top layers, frame-to-frame temporal variation is known to be smooth. The motion of video patches tend to be restricted to a local neighborhood. For this reason, we argue that current RCN architectures are not well suited for capturing fine motion information. Instead, they are more likely focus on global appearance changes such as shot transitions. To address this issue, we introduce a novel RCN architecture that applies an RNN not solely on the 2D CNN top-layer but also on the intermediate convolutional layers. Convolutional layer activations, or convolutional maps, preserve a finer spatial resolution of the input video from which local spatio-temporal patterns are extracted.

Applying an RNN directly on intermediate convolutional maps, however, inevitably results in a drastic number of parameters characterizing the input-to-hidden transformation due to the convolutional maps size. On the other hand, convolutional maps preserve the frame spatial topology. We propose to leverage this topology by introducing sparsity and locality in the RNN units to reduce the memory requirement. We extend the GRU model [5] and replace the fully-connected RNN linear product operation with a convolution. Our GRU-extension therefore encodes the locality and temporal smoothness prior of videos directly in the model structure.

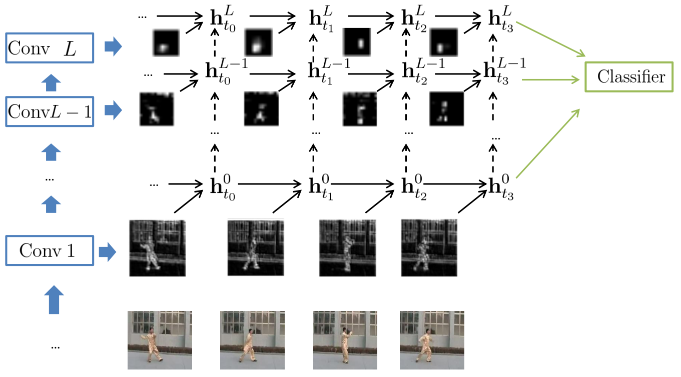

Figure 1: Our approach leverages convolutional maps from different layers of a pretrained-convnet. Each map is given as input to a convolutional GRU-RCN at different time-step. Bottom-up connections may be optionally added between RCN layers.

2 GRU-RCN

Let's consider $(\mathbf{x}^1_t, ..., \mathbf{x}^{L-1}_t, \mathbf{x}^L_t)_{(t=1..T)}$, a set of 2D convolutional maps extracted from L layers at different time steps in a video. We propose to apply L RNNs independently on each convolutional map. We define L RNNs as $\phi^1, ..., \phi^L$, such that $\mathbf{h}^l_t = \phi^l(\mathbf{x}^l_t, \mathbf{h}^l_{t-1})$. The hidden representation of the final time step $\mathbf{h}^1_T, ..., \mathbf{h}^L_T$ are then fed to a classification layer in the case of action recognition, or to a text-decoder RNN for caption generation.

To implement the RNN recurrent function ϕ^l, we propose to leverage Gated Recurrent Units [5]. GRU models input to hidden-state and hidden to hidden transitions using fully connected units. However, convolutional map inputs are 3D tensors (spatial dimension and input channels). Applying a GRU directly can lead to a drastic number of parameters. Let N_1, N_2 and O_x be the input convolutional map spatial size and number of channels. Applying a GRU directly would require input-to-hidden parameters $\mathbf{W}^l$, $\mathbf{W}^l_z$ and $\mathbf{W}^l_r$ to be of size $N_1 \times N_2 \times O_x \times O_h$ where O_h is the dimensionality of the GRU hidden representation. To tackle this issue, we replace the fully-connected units in GRU with convolution operations. Our recurrent units therefore have sparse connectivity and share their parameters across different input spatial and temporal locations:

$$\begin{aligned}
\mathbf{z}^l_t &= \sigma(\mathbf{W}^l_z * \mathbf{x}^l_t + \mathbf{U}^l_z * \mathbf{h}^l_{t-1}), && (1)\\
\mathbf{r}^l_t &= \sigma(\mathbf{W}^l_r * \mathbf{x}^l_t + \mathbf{U}^l_r * \mathbf{h}^l_{t-1}), && (2)\\
\tilde{\mathbf{h}}^l_t &= \tanh(\mathbf{W}^l * \mathbf{x}^l_t + \mathbf{U} * (\mathbf{r}^l_t \odot \mathbf{h}^l_{t-1}), && (3)\\
\mathbf{h}^l_t &= (1 - \mathbf{z}^l_t)\mathbf{h}^l_{t-1} + \mathbf{z}^l_t \tilde{\mathbf{h}}^l_t, && (4)
\end{aligned}$$

where $*$ denotes a convolution operation. In this formulation, Model parameters $\mathbf{W}, \mathbf{W}^l_z, \mathbf{W}^l_r$ and $\mathbf{U}^l, \mathbf{U}^l_z, \mathbf{U}^l_r$ are 2D-convolutional kernels. Our model results in hidden recurrent representation that preserves the spatial topology, $\mathbf{h}^l_t = (\mathbf{h}^l_t(i,j))$ where $\mathbf{h}^l_t(i,j))$ is a feature vector defined at the location (i,j). To ensure that the spatial size of the hidden representation remains fixed over time, we use zero-padding in the recurrent convolutions.

Using convolution parameters $\mathbf{W}^l$, $\mathbf{W}^l_z$ and $\mathbf{W}^l_r$ have a size of $k_1 \times k_2 \times O_x \times O_h$ where $k_1 \times k_2$ is the convolutional kernel spatial size (usually 3×3), chosen to be significantly lower than convolutional map size $N_1 \times N_2$. The candidate hidden representation $\tilde{\mathbf{h}}_t(i,j)$, the activation gate $\mathbf{z}_k(i,j)$ and the reset gate $\mathbf{r}_k(i,j)$ are defined based on a local neigborhood of size $(k_1 \times k_2)$ at the location(i,j) in both the input data $\mathbf{x}_t$ and the previous hidden-state $\mathbf{h}_{t-1}$. In addition, the size of receptive field associated with $\mathbf{h}^l(i,j)_t$ increases in the previous representation $\mathbf{h}^l_{t-1}, \mathbf{h}^l_{t-2}$... as we go back further in time. Our model is therefore capable of characterizing spatio-temporal patterns with high spatial variation in time.

We can also extend our model by adding bottom-up connection. We precondition each GRU-RNN on the output of the previous GRU-RNN at the current time step: $\mathbf{h}^l_t = \phi^l(\mathbf{h}^l_{t-1}, \mathbf{h}^{l-1}_t, \mathbf{x}^l_t)$. Adding this extra-connection brings more flexibility and gives the opportunity for the model to leverage representations with different resolutions. Adding this extra-connection brings more flexibility and gives the opportunity for the model to leverage representations with different resolutions.

3 Experimentation

3.1 Action Recognition

We evaluate our approach on the UCF101 dataset [10]. We report results on the dataset UCF101 first split. We follow the training procedure and evalution introduced by the two-stream framework [9].

We extract visual "percept" using VGG-16 CNNs that consider either RGB or flow inputs. VGG-16 CNNs are pretrained on ImageNet and fine-tuned on the UCF-101 dataset, following the protocol in Wang et al. [14]. We then extract the convolution maps from *pool2*, *pool3*, *pool4*, *pool5* layers and the fully-connected map from layer *fc-7*. Those features maps are given as inputs to our RCN models.

Method	RGB	Flow
VGG-16	78.0	85.4
VGG-16 RNN	78.1	84.9
GRU-RCN	79.9	**85.7**
Stacked-GRU RCN	78.3	-
Bi-directional GRU-RCN	**80.7**	-
Two-Stream [9]	72.8	81.2
Two-Stream + LSTM [6]	71.1	76.9
Improved Two-Stream [14]	**79.8**	**85.7**
C3D one network [12], 1 million videos as training	82.3	-
C3D ensemble [12], 1 million videos as training	**85.2**	-

Table 1: Classification accuracy on the UCF101 split 1.

In Table 1, we evaluate three architectures, GRU-RCN, Stacked GRU-RCN, adding bottom-up connection, and Bi-direcitonal GRU-RCN that runs two GRU-RCN, one forward and one backward in time. We compare our approaches with two different baselines, VGG-16 [14] and VGG-16 RNN that applies a GRU-RNN on top of *fc-7* representation. GRU-RCN outperforms the baselines, showing the benefit of delving deeper into a CNN. Stacked-GRU RCN performs significantly lower than GRU-RCN. We argue that bottom-up connection, increasing the depth of the model, combined with the lack of training data make the Stacked-GRU RCN learning difficult. The Bi-directional GRU-RCN performs the best among the GRU-RCN. Bi-directional GRU-RCN obtains a gain 3.4% in term of performances, relatively to the baselines that focus only the VGG-16 top layer with RGB inputs.

We also investigate the combination of the RGB and flow streams following [14]. Fusion the VGG-16 model baseline achieve an accuracy of 89.1. Combining the RGB Bi-directional GRU-RCN with the flow GRU-RCN achieves a performance gain of 1.9% over baseline, reaching 90.8. Our model is on part with [14] that obtain state-of-art results using both RGB and flow streams which obtains 90.9.

3.2 Video Captioning

We evaluate our approach on the captioning problem using YouTube2Text dataset [4]. To perform video captioning, we use the encoder-decoder framework with soft-attention based decoder [15]. As for encoder, we compare both VGG-16 CNN and Bi-directional GRU-RCN.

Table 2 reports the performance of our proposed method using automatic BLEU, METEOR and CIDER metrics. All models are early-stopped based on the negative-log-likelihood (NLL) of the validation set. We then select the model that performs best on the validation set according to the metric at consideration.

The first two lines of Table 2 compare the performances of the VGG-16 and Bi-directional GRU-RCN encoder. Results clearly show the superiority of the Bi-Directional GRU-RCN Encoder as it outperforms the VGG-16 Encoder on all three metrics. In particular, GRU-RCN Encoder obtains a performance gain of 10% compared to the VGG-16 Encoder according to the BLEU metric. Combining our GRU-RCN Encoder that focuses on *action* with a GoogleNet Encoder that captures visual *entities* further improve the performances.

	YouTube2Text		
Model	BLEU	METEOR	CIDEr
VGG-16	0.3700	0.2640	0.4330
Bi-dir GRU-RCN	0.4100	0.2850	0.5010
GoogleNet	0.4128	0.2900	0.4804
GoogleNet + Bidir GRU-RCN	**0.4963**	**0.3170**	**0.6801**
GoogleNet + HRNE [8]	0.436	**0.321**	-
VGG + p-RNN [16]	0.443	0.311	-
VGG + C3D + p-RNN [16]	**0.499**	**0.326**	-
Soft-attention [15]	0.4192	0.2960	0.5167
Venugopalan *et al.* [13]	0.3119	0.2687	-
+ Extra Data (Flickr30k, COCO)	0.3329	0.2907	-
Thomason *et al.* [11]	0.1368	0.2390	-

Table 2: Performance of different variants of the model on YouTube2Text for video captioning.

Our GoogleNet + Bi-directional GRU-RCN approach significantly outperforms Soft-attention [15] that relies on a GoogLeNet and cuboids-based 3D-CNN Encoder, in conjunction to a similar soft-attention decoder. This result indicates that our approach is able to offer more effective representations. According to the BLEU metric, we also outperform other approaches using more complex decoder schemes such as spatial and temporal attention decoder [16] or a hierarchical RNN decoder [8] Our approach is on par with [16], without the need of using a C3D-encoder that requires training on large-scale video dataset.

Acknowledgments

The authors would like to acknowledge the support of the following agencies for research funding and computing support: NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR. We would also like to thank the developers of Theano [2, 3] , for developing such a powerful tool for scientific computing.

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

[2] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. NIPS Workshop, 2012.

[3] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *SciPy*, 2010.

[4] D. Chen and W. Dolan. Collecting highly parallel data for paraphrase evaluation. In *ACL*. Association for Computational Linguistics, 2011.

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv*, 2014.

[6] J. Donahue, L. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv*, 2014.

[7] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, 2014.

[8] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. *arXiv*, 2015.

[9] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

[10] K. Soomro, A. Roshan Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv*, 2012.

[11] J. Thomason, S. Venugopalan, S. Guadarrama, K. Saenko, and R. Mooney. Integrating language and vision to generate natural language descriptions of videos in the wild. In *COLING*, 2014.

[12] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3d: generic features for video analysis. *arXiv*, 2014.

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

[14] L Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *arXiv*, 2015.

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

[16] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu. Video paragraph captioning using hierarchical recurrent neural networks. *arXiv*, 2015.