The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

Reducing the Number of Multiplications in Convolutional Recurrent Neural Networks (ConvRNNs)

Daria Vazhenina Atsunori Kanemura

LeapMind Inc.

Convolutional variants of recurrent neural networks, ConvRNNs, are widely used for spatio-temporal modeling. Although ConvRNNs are suited to model two-dimensional sequences, the introduction of convolution operation brings additional parameters and increases the computational complexity. The computation load can be obstacles in putting ConvRNNs in operation in real-world applications. We propose to reduce the number of parameters and multiplications by substituting some convolutional operations with the Hadamard product. We evaluate our proposal using the task of next video frame prediction and the Moving MNIST dataset. The proposed method requires 38% less multiplications and 21% less parameters compared to the fully convolutional counterpart. In price of the reduced computational complexity, the performance measured by for structural similarity index measure (SSIM) decreased about 1.5%. ConvRNNs with reduced computations can be used in more various situations like in web apps or embedded systems.

1. Introduction

Convolutional recurrent neural networks (ConvRNNs) are widely used because of their ability to model temporal information using 2D input. Donahue et al. [2] proposed to stack CNN and LSTM layers for sequential video processing. Xingjian et al. [6] combined those techniques in ConvLSTM layer and showed its effectiveness on two different tasks with 2D input and temporal dependencies. It allowed to obtain informative representation that improve overall model performance. Representations (or features) of a video are useful in video contents description, activity recognition, and other tasks. ConvRNNs parse video frames sequentially and encode frame-level information.

While lots of work were done on speeding up and improving performance of conventional (i.e. non-convolutional) RNNs, less attention was paid to ConvRNNs. Similar to conventional RNNs, gated variants were proposed for their convolutional conterparts, such as ConvLSTM [6] with three gates, ConvGRU [1] with two gates, and reduced-gate ConvLSTM [3] with one gate, which is variant of non-convolutional JaNet [9]. Gates are sensitive to short-term and long-term patterns in the input and help to overcome vanishing gradient problem. Other improvements for ConvRNNs focused on fitting better to their target task and increased overall model performance, but they resulted in increasing memory footprint, the number of parameters, and the number of floating point multiplications [7, 10].

Recently, for conventional RNNs, Li et al. [4] proposed to reduce the numbers of multiplications and parameters by substituting matrix multiplication between a weight matrix and an input vector to the Hadamard product. It allowed to build a deeper network of up to 21 layers and slightly outperformed state-of-the-art models for three different tasks.

We investigate the influence of substituting convolution with the Hadamard product in ConvLSTM, ConvGRU, and reduced-gate LSTM. It is expected that such substitution would result in a large performance drop. Then, instead of replacing all the convolution operations in a network, we used the Hadamard product only in some parts of a network, in order to keep a good balance between reducing the numbers of parameters and multiplications and performance drops.

As an example of using ConvRNN, we selected the task of next video frame prediction, which learns video sequence representations of individual video frames in an unsupervised manner [8]. The next frame prediction problem is useful because 1) we don't need to obtain lots of labeled data, which is often a difficult task before adopting deep learning, and 2) whereas a system trained for one specific task will learn representations for that specific task, internal representations learned by the model for predicting next frames will be re-usable for other tasks [8]. Also, targets in supervised learning contain much less information than input data, especially in terms of video action recognition, where there is only one label per many frames. That is why learning how to forecast the future of an image sequence requires the prediction model to understand and efficiently encode the content and dynamics for a certain period of time.

Contact: Daria Vazhenina, LeapMind Inc., Tokyo 150-0044, Japan, daria@leapmind.io

2. ConvRNNs

2.1 Basic ConvRNN architecture

The most widely used variant of ConvRNN is ConvLSTM, proposed in [6], where conventional LSTM is modified by replacing matrix multiplication with convolution and changing the shape of input X_t to 2D from 1D. A ConvLSTM cell is described by the following equations:

$$i_t = \sigma(X_t * W_{xi} + h_{t-1} * W_{hi} + b_i), \quad (1)$$

$$f_t = \sigma(X_t * W_{xf} + h_{t-1} * W_{hf} + b_f), \quad (2)$$

$$o_t = \sigma(X_t * W_{xo} + h_{t-1} * W_{h0} + b_o), \quad (3)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(X_t * W_{xc} + h_{t-1} * W_{hc} + b_c), \quad (4)$$

$$h_t = \tanh(c_t \odot o_t), \quad (5)$$

where $*$ means convolutional operation; $W_{..}$ is a set of convolutional kernels; h_t and h_{t-1} are current and previous hidden states, respectively; i_t, f_t, and o_t are input, forget, and output gates, respectively; and c_t is cell output.

2.2 Next video frame prediction model using ConvRNN

In unsupervised video representation model, ConvRNN cells are combined in composite model described in [8], which has an encoder-decoder pipeline with two different decoders termed a predictor and a reconstructor. In the composite model, several video frames are fed into recurrent encoder and then its final hidden state, so called a learned representation, is used as the initial hidden state in the predictor decoder and the reconstructor decoder, while the two decoders do not share all other parameters (e.g. weight matrices). Learning the two tasks of prediction and reconstruction using the same encoder allows us to improve overall model performance, rather than using different encoders. The loss function is defined to be the sum of one for the reconstructor and the other for the predictor, and we can do backpropagation on it.

Our experiments in this paper are based on the network architecture proposed in [5], where the next frame prediction pipeline has been improved by stacking recurrent layers on the top of the CNN layers in the encoder part and reversing those recurrent and CNN layers in the decoder part, changing the CNN layers into so-called deconvolution layers. This stacking technique reduces the resolution of input image frames with convolutional layers before feeding it to the recurrent ones, thus reducing the number of parameters in the recurrent layers. This significantly reduce mean square error (MSE) of the model for both prediction and reconstruction tasks.

3. Proposed improvements

We investigated and analyzed replacing convolutional operation with Hadamard product in ConvRNN cells. This idea was inspired by IndRNN [4], which reduces the number of multiplications in the vanilla recurrent cell. Since IndRNN was proposed for RNNs without gates, we cannot adopt it for our purpose of improving ConvLSTM, which includes many gates. This reduction of the number of multiplications has not been investigated so far for gated convolutional recurrent units.

Here is the comparison of the expected computational complexity (CC; lower is better) and performance mean square error (MSE; lower is better):

$$CC_{\text{Hadamard}} < CC_{\text{Combi}} \ll CC_{\text{Baseline}},$$
$$MSE_{\text{Hadamard}} \gg MSE_{\text{Combi}} \geq MSE_{\text{Baseline}}$$

where the notation is as follows:

- Baseline (Conv*): The baseline model with standard ConvRNN structure,
- Combi (ConvIndConv*): Proposed combination of convolution and Hadamard product,
- Hadamard (ConvInd*): The model where convolution is replaced with Hadamard product.

Here the asterisk sign "*" means LSTM, GRU, or Janet. That is, "ConvInd*" means ConvIndLSTM, ConvIndGRU, or ConvIndJanet.

As baselines for applying Hadamard product, we used ConvLSTM, ConvGRU, and ConvJanet. Hadamard product was used to calculate gates and cell unit in ConvInd* models:

$$i_t = \sigma(X_t * W_{xi} + h_{t-1} \odot W_{hi} + b_i), \quad (6)$$
$$f_t = \sigma(X_t * W_{xf} + h_{t-1} \odot W_{hf} + b_f), \quad (7)$$
$$o_t = \sigma(X_t * W_{xo} + h_{t-1} \odot W_{h0} + b_o), \quad (8)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(X_t * W_{xc} + h_{t-1} \odot W_{hc} + b_c), \quad (9)$$
$$h_t = \tanh(c_t \odot o_t) \quad (10)$$

and Hadamard product was used for gates calculations only in ConvIndConv* models, so c_t is calculated as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(X_t * W_{xc} + h_{t-1} * W_{hc} + b_c) \quad (11)$$

4. Experimental results

4.1 Database description

We use the moving MNIST dataset, which has been widely used to evaluate video frame prediction models. We used same train and test settings as in [8]. Each sequence in this dataset consists of 20 image frames of size 64×64 with two random moving digits from the MNIST dataset. The performance in next frame prediction was measured by MSE (the low this value the better) and SSIM (the higher this value the better) between prediction and the ground truth [11].

4.2 Performance evaluation

We evaluated three types of ConvRNNs and their ConvInd* and ConvIndConv* variants described in Section 3.. We considered the number of multiplications and the number of parameters for complexity evaluation.

As shown in Fig. 1, the model with ConvLSTM cell showed the best performance in terms of SSIM, while the model with ConvGRU cell was slightly better in terms of MSE. All models with ConvInd* cell showed significant drop in performance about 30% relative to their fully convolutional counterparts. Returning one convolution operation in the ConvIndConv* cell allowed to reduce performance drop to about 5% relative to their fully convolutional counterparts.

Fig. 2 shows that the best performing model with ConvLSTM cell requires the largest numbers of multiplications and parameters. The model with ConvIndConvLSTM cell showed the second best SSIM value and required about 37.5% less multiplications and about 21% less parameters. Its relative drop of SSIM value is 1.54% and MSE is 4.2%. This shows that it is possible to use less parameters and multiplications for gates calculations without big loss in performance.

5. Conclusions

In this work, we compared three gated ConvRNN variants using next video frame prediction task. We showed that it is possible to reduce the number of parameters and multiplications in the ConvRNN architecture with a small drop in the performance of the overall model. ConvInd* models provide significant reduction in the number of parameters and multiplications, but drop in performance of those models is pretty large. Proposed ConvIndConv* models, where convolution operation is kept in the cell computation, allowed to achieve minor drop in performance compared to ConvInd* models, and also kept number of parameters and multiplications smaller compared to Conv* models.

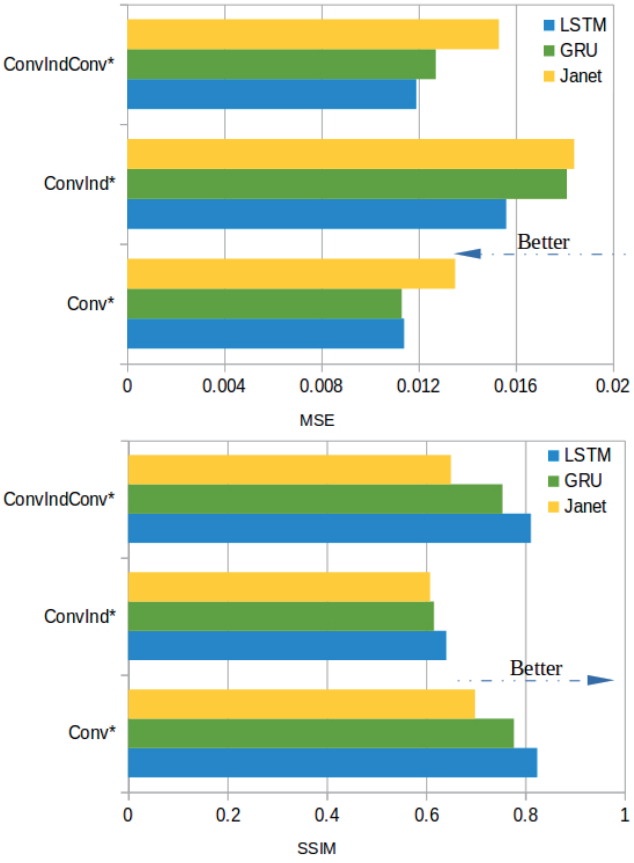

Figure 1: ConvInd* and ConvIndConv* models performance evaluation in terms of MSE and SSIM.

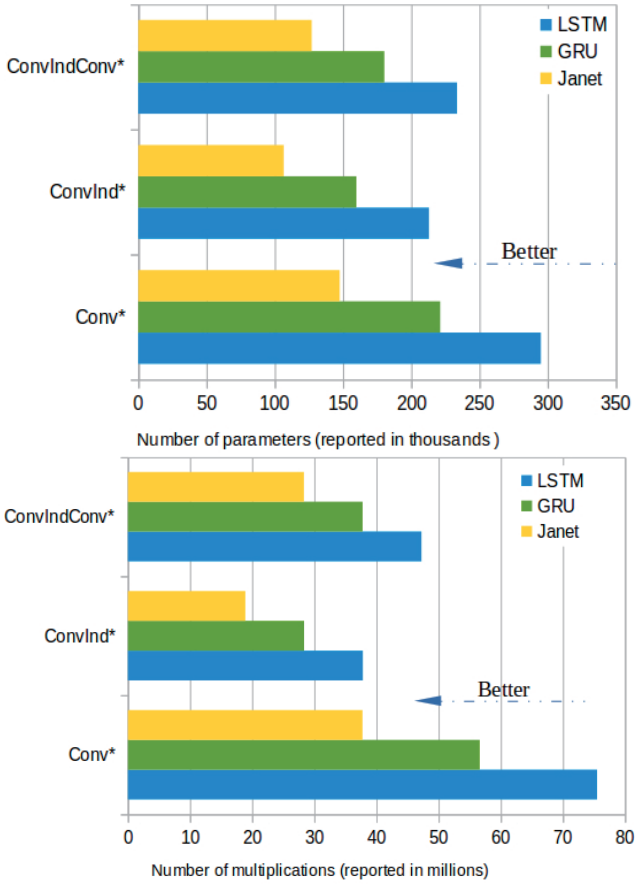

Figure 2: ConvInd* and ConvIndConv* models performance evaluation in terms of number of parameters and multiplications.

References

[1] N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. In *Int. Conf. Learning Representations (ICLR)*, 2016.

[2] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015.

[3] N. Elsayed, A. S. Maida, and M. Bayoumi. Reduced-gate convolutional LSTM using predictive coding for spatiotemporal prediction. *arXiv:1810.07251*, 2018.

[4] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao. Independently recurrent neural network (IndRNN): Building a longer and deeper RNN. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5457–5466, 2018.

[5] B. Sautermeister. Deep learning approaches to predict future frames in videos. Master's thesis, Technishe Universität München, 2016.

[6] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems (NIPS)*, pages 802–810, 2015.

[7] X. Shi, Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W. Wong, and W. Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5617–5627, 2017.

[8] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using LSTMs. In *Int. Conf. Machine Learning (ICML)*, pages 843–852, 2015.

[9] J. van der Westhuizen and J. Lasenby. The unreasonable effectiveness of the forget gate. *arXiv:1804.04849*, 2018.

[10] Y. Wang, Z. Gao, M. Long, J. Wang, and P. S. Yu. PredRNN++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In *Int. Conf. Machine Learning (ICML)*, 2018.

[11] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004.