Making Convolutional Networks Recurrent for Visual Sequence Learning
SUPPLEMENTARY MATERIAL

Xiaodong Yang Pavlo Molchanov Jan Kautz
NVIDIA
{xiaodongy,pmolchanov,jkautz}@nvidia.com

In this supplementary material, Section 1 summarizes more implementation details of our networks in the experiments of the three applications including sequential face alignment, dynamic hand gesture recognition, and action recognition. Section 2 provides more comparison results on the hidden state dimensions of traditional RNNs and PreRNN. Section 3 shows the results of different hierarchy of recurrent layers in PreRNN. Finally, we provide the ablation study of structure and initialization of PreRNN in Section 4.

1. More Implementation Details

We employ the stochastic gradient descent with momentum, and set the momentum to 0.9 and weight decay to 5×10^{-4} for all experiments. We apply random scaling, cropping and flipping for data augmentation. We use the following learning rate scheduling schemes to let the networks fully converge.

- In the experiments on sequential face alignment, we set the base learning rate to 1×10^{-4}, and decay the learning rate after 10 epochs by multiplying it with a factor of 0.5 after each additional 5 epochs. Each network is trained for 30 epochs.
- In the experiments on dynamic hand gesture recognition, we set the initial learning rate to 3×10^{-3}, and divide it by 10 if the training loss has not improved over the past 40 epochs. We stop the training when the learning rate lowers to 3×10^{-7}.
- In the experiments on action recognition, the learning rate starts from 3×10^{-4} for the spatial stream and 1×10^{-2} for the temporal stream, and is divided by 2 after every 10 epochs for both streams. We train the networks for 90 epochs and 100 epochs for the spatial stream and temporal stream, respectively.

In all experiments, backbone CNNs are jointly fine-tuned when training RNNs. To implement PreRNN for LSTM and GRU (with gate-dependent input-to-hidden state), we directly use the pre-trained $\boldsymbol{W}_{xy}$ as the input-to-hidden weights for one gate (sharing the same data memory), and copy the values of $\boldsymbol{W}_{xy}$ to initialize the input-to-hidden weights for other gates (localizing different data memory).

2. Dimensionality of Hidden State

In order to compare the performance in a controlled setting, we carefully choose the hidden state dimensions for each basic recurrent structure so that the total number of parameters in traditional RNNs and PreRNN are as close as possible. For PreRNN, the hidden state dimension is determined by the dimensionality of the transformed feedforward layer of a pre-trained CNN, i.e., 4096 for VGG16, 4096 for C3D, and 2048 for ResNet50. Based on this, we calculate how many recurrent parameters to introduce and match them for the traditional RNNs.

Here we show an additional experiment by varying the hidden state dimensions (or network capacities) of the traditional RNNs. Table 1 shows the testing loss (i.e., the Euclidean distance between the predicted facial landmarks and the ground truth) on the 300VW dataset by using 2 recurrent layers. The numbers in parentheses of the second last column indicate the hidden state dimensions of the traditional RNNs that have similar numbers of parameters matching to PreRNN. We observe that each basic recurrent structure tends to be improved along with the increase of hidden state dimensions, but with quite diminished gains after 4096 dimensions. Compared to traditional RNNs, PreRNN loses one hyper-parameter (i.e., the hidden state dimension) to tune. However, PreRNN is found to consistently outperform the traditional RNNs across various hidden state dimensions in Table 1.

	Traditional						PreRNN
	256	512	1024	2048	4096	Similar	
VRNN	0.1414	0.1509	0.1462	0.1398	0.1216	0.1255 (2740)	**0.0819**
LSTM	0.2039	0.1806	0.1669	0.1648	0.1508	0.1523 (5761)	**0.0951**
GRU	0.1471	0.1409	0.1295	0.1292	0.1286	0.1291 (5497)	**0.0866**

Table 1. Testing loss of PreRNN and the traditional RNNs with various hidden state dimensions on the 300VW dataset.

3. Hierarchy of Recurrent Layers

We have transformed either one or both of the two `fc` layers of VGG16 and C3D into recurrent layers by PreRNN. Now we study the effect of further growing the recurrent hierarchy by stacking more traditional recurrent layers (randomly initialized) on top of the transformed `fc6` and `fc7` layers, leading to a 3-layer PreRNN. We have discussed the performance difference between the 1-layer and 2-layer PreLSTM and PreGRU by investigating the internal gating mechanism in the main paper. As shown in Table 2, the 3-layer PreRNN still performs better than traditional RNNs (compare to the results in Table 1), it is however much inferior to the 1-layer and 2-layer PreRNN, due to the fact that the third recurrent layer, which is fully randomly initialized, undermines the generalization capability. This also highlights the advantage of PreRNN in making recurrent layers from the pre-trained feedforward layers of CNNs.

	1 Layer		2 Layers	3 Layers
	`fc6`	`fc7`	`fc6/7`	`fc6/7` + random
VRNN	0.0898	0.0958	**0.0819**	0.1122
LSTM	**0.0822**	0.0882	0.0951	0.1454
GRU	**0.0783**	0.0891	0.0866	0.1077

Table 2. Comparison of the different numbers of recurrent layers in PreRNN on the 300VW dataset, with the testing loss reported.

4. Structure and Initialization

PreRNN primarily benefits from the two factors: (i) the proposed recurrent structure and (ii) the (partial) initialization by pre-trained weights of CNNs. Here we conduct an additional ablation study to analyze the importance of the two ingredients through comparing to the networks that are with the same structure as PreRNN but randomly initialize $\boldsymbol{W}_{xy}$. Table 3 shows their results combining two streams on the first split of UCF101. We observe that PreRNN with random initialization still slightly outperforms traditional RNNs by 0.4% on average. Furthermore, PreRNN using the pre-trained $\boldsymbol{W}_{xy}$ consistently performs better than the randomly initialized with 0.9% improvement on average. This clearly demonstrates that both structure and initialization proposed in PreRNN attribute to the superior performance.

	PreRNN		PreRNN-SIH		Traditional
	pre-trained	random	pre-trained	random	
VRNN	**92.7%**	92.0%	-	-	91.6%
LSTM	**93.2%**	92.8%	**93.5%**	92.7%	92.5%
GRU	**93.7%**	92.8%	**93.3%**	91.8%	92.2%

Table 3. Classification accuracy of PreRNN with different initializations and traditional RNNs on the first split of UCF101.