Making Convolutional Networks Recurrent for Visual Sequence Learning

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

Abstract

Recurrent neural networks (RNNs) have emerged as a powerful model for a broad range of machine learning problems that involve sequential data. While an abundance of work exists to understand and improve RNNs in the context of language and audio signals such as language modeling and speech recognition, relatively little attention has been paid to analyze or modify RNNs for visual sequences, which by nature have distinct properties. In this paper, we aim to bridge this gap and present the first large-scale exploration of RNNs for visual sequence learning. In particular, with the intention of leveraging the strong generalization capacity of pre-trained convolutional neural networks (CNNs), we propose a novel and effective approach, PreRNN, to make pre-trained CNNs recurrent by transforming convolutional layers or fully connected layers into recurrent layers. We conduct extensive evaluations on three representative visual sequence learning tasks: sequential face alignment, dynamic hand gesture recognition, and action recognition. Our experiments reveal that PreRNN consistently outperforms the traditional RNNs and achieves state-of-the-art results on the three applications, suggesting that PreRNN is more suitable for visual sequence learning.

1. Introduction

Recurrent neural networks (RNNs) have achieved excellent performance on a variety of sequential learning problems including language modeling [27], handwriting recognition [16], machine translation [6], speech recognition [17], polyphonic music modeling [8], and intelligent video analytics [10]. A vanilla recurrent neural network (VRNN) [2] extends the conventional feedforward network to handle a variable-length sequence by accumulating the context of previous inputs in its internal state to influence proceeding outputs. However, the range of context that can be accessed in VRNN is limited as the gradients tend to either vanish or explode [2]. Unlike the gradient exploding problem which is relatively easy to address through gradient norm clipping [32], the gradient vanishing problem involves devising more sophisticated gating mechanism. As an earliest attempt in this direction, the long short-term memory (LSTM) [21] adopts a memory cell to maintain the internal state over time and employs gating functions to modulate the information flow into and out of the cell. A simpler alternative to LSTM, the gated recurrent unit (GRU) [6] modifies the functional gates and has been growing increasingly popular. Recently, there have been a number of attempts to understand and further improve these basic recurrent structures for language and speech modeling as seen with the memory network [47], the massive evolutionary architecture search [23], the content based soft attention scheme [1], and the ablation study of removing various gates and connections [18].

RNNs have also been widely applied for visual sequence learning tasks to model dynamic evolutions and provide temporal contexts. However, visual sequences have by nature distinct properties compared to other sequential data. In contrast to language and speech, the processing unit of a visual sequence is in a more structured format such as an image or a short video snippet. Therefore, convolutional neural networks (CNNs) usually serve as the backbone networks to extract semantic features, which RNNs are then built on top. A key advantage of the feature extraction for visual sequences is to exploit the extremely expressive CNN models that are pre-trained on large-scale image or video datasets such as ImageNet [9] and Sports1M [25]. However, it remains an open question how to construct RNNs to better leverage the representational power and generalization ability of these pre-trained CNNs. In addition, visual sequences typically exhibit large redundancy [36] and have diverse temporal dependencies depending on different applications [19, 28, 29]. As a result, it is still poorly understood which recurrent structure and which gating mechanism is best suited.

Our main contributions in this paper are as follows. First, we propose PreRNN, which is an effective approach to make pre-trained CNNs recurrent by directly transforming pre-trained convolutional layers or fully connected layers into recurrent layers. PreRNN is applicable to all three basic recurrent structures (i.e., VRNN, LSTM and GRU) and

can fully take advantage of the strong generalization capability of pre-trained CNNs. Second, a simplified alternative PreRNN-SIH is proposed to refine the gating functions and reduce the number of recurrent parameters. Third, we systematically analyze the internal mechanism of the gating units and demonstrate this can be used as an insightful guidance to better understand and design recurrent architectures for visual sequence learning. Fourth, we extensively evaluate a variety of recurrent convolutional architectures on the three representative visual sequence learning tasks: sequential face alignment, dynamic hand gesture recognition, and action recognition. As summarized in Table 1, our evaluations represent a large diversity of applications, visual sequence types, pre-trained backbone CNNs, and objective functions. To our knowledge, this work provides the first large-scale exploration of different recurrent convolutional networks for visual sequence learning.

2. Notation and Related Work

In this section, we introduce the notation used throughout this paper and summarize the related work. RNNs have been well studied for decades in sequence learning, which mainly includes language modeling [27], machine translation [6] and speech recognition [17]. VRNN [2] contains a recurrent or self-connected hidden state $\boldsymbol{h}_t$, whose activation depends on that of the previous time step:

$$\boldsymbol{h}_t = \mathcal{H}(\boldsymbol{W}_{ih}\boldsymbol{y}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1}), \tag{1}$$

where $\mathcal{H}$ is an activation function, $\boldsymbol{W}_{ih}$ is the input-to-hidden matrix, $\boldsymbol{W}_{hh}$ is the hidden-to-hidden matrix, $\boldsymbol{y}_t$ is the input to this recurrent layer. We omit the bias vector for brevity. In order to enhance the capability to use contextual information, a great amount of efforts have been made to mitigate the gradient vanishing problem for VRNN. Among the most successful variants are LSTM and GRU, which incorporate gating functions into their state dynamics. At each time, LSTM [21] maintains a memory cell $\boldsymbol{c}_t$ and a hidden state $\boldsymbol{h}_t$ that are carefully regulated by the gates:

$$\begin{aligned}
\boldsymbol{i}_t &= \text{sigm}(\boldsymbol{W}_{ii}\boldsymbol{y}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1}),\\
\boldsymbol{f}_t &= \text{sigm}(\boldsymbol{W}_{if}\boldsymbol{y}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1}),\\
\boldsymbol{o}_t &= \text{sigm}(\boldsymbol{W}_{io}\boldsymbol{y}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1}),\\
\tilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{W}_{ic}\boldsymbol{y}_t + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1}),\\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t,\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t).
\end{aligned} \tag{2}$$

Similarly $\boldsymbol{W}_{i\cdot}$ are the input-to-hidden matrices and $\boldsymbol{W}_{h\cdot}$ are the hidden-to-hidden matrices. Here $\boldsymbol{i}_t$, $\boldsymbol{f}_t$ and $\boldsymbol{o}_t$ are respectively the input, forget and output gates, $\tilde{\boldsymbol{c}}_t$ is the new memory state, and $\odot$ is the element-wise product. GRU [6] simplifies LSTM primarily by merging the hidden state and memory cell and combining the forget and input gates into a single update gate:

$$\begin{aligned}
\boldsymbol{r}_t &= \text{sigm}(\boldsymbol{W}_{ir}\boldsymbol{y}_t + \boldsymbol{W}_{hr}\boldsymbol{h}_{t-1}),\\
\boldsymbol{z}_t &= \text{sigm}(\boldsymbol{W}_{iz}\boldsymbol{y}_t + \boldsymbol{W}_{hz}\boldsymbol{h}_{t-1}),\\
\tilde{\boldsymbol{h}}_t &= \tanh(\boldsymbol{W}_{ih}\boldsymbol{y}_t + \boldsymbol{W}_{hh}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})),\\
\boldsymbol{h}_t &= (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t,
\end{aligned} \tag{3}$$

where $\boldsymbol{r}_t$ and $\boldsymbol{z}_t$ are the reset and update gates, and $\tilde{\boldsymbol{h}}_t$ is the candidate hidden state. Note that for the above three basic recurrent structures in Eqs. (1-3) multiple recurrent layers can be stacked on top of each other to perform deep and hierarchical recurrent processing.

In recent years, there has been a growing interest in understanding the properties of RNNs and modifying the functional components to improve upon the three basic recurrent structures. In [8] Chung et al. empirically evaluate GRU with comparison to LSTM and find the two gating strategies to be comparable. Greff et al. in [18] present a large-scale analysis on the importance of different gates and variations of LSTM, and show that the forget gate and output activation function are the most crucial elements. Jozefowicz et al. [23] perform an extensive search of over ten thousand RNN structures to determine whether better structures exist by means of mutating network components. Karpathy et al. [24] investigate the predictions, representations, and error types presented in RNNs. Pascanu et al. in [31] provide a few different ways to build and interpret deep extensions of RNNs. All these studies are conducted in the context of language and audio sequence modeling, while in this paper we focus on visual sequence learning.

RNNs are mostly attached on top of the last layer of pre-trained CNNs in visual sequence learning tasks, as this can harness the strong representational ability of these pre-trained models and capture the long-term temporal contexts. In [29] a few LSTM layers are stacked upon the pre-trained AlexNet [26] and GoogLeNet [41] for action recognition. Donahue et al. [10] also place LSTM after a fully connected layer of the pre-trained ZFNet [51] for activity recognition. Yang et al. [48] merge VRNN with the pre-trained VGG16 [38] and C3D [43] for video classification. VRNN in [28] is employed with the pre-trained C3D to enable online detection and classification of dynamic hand gestures. Peng et al. [30] accompany LSTM with the pre-trained VGG16 for facial landmark detection in videos. In [52] LSTM is combined with the pre-trained AlexNet for video based person re-identification. Tokmakov et al. [42] append GRU on top of the pre-trained network DeepLab [5] for video object segmentation. In contrast to the previous work, we aim to propose a more effective and generalized approach to directly make the pre-trained CNNs recurrent and obtain an in-depth understanding of the internal mechanism for visual sequence learning.

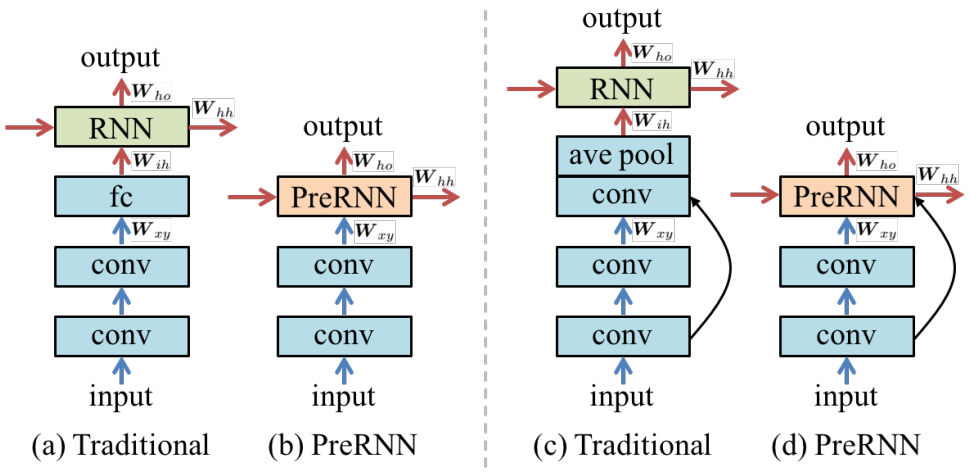

Figure 1. A schematic overview of the traditional RNN and the proposed PreRNN. Each colored arrow represents the corresponding network weights: blue arrows are the weights of pre-trained CNNs and red arrows denote the randomly initialized weights introduced by the recurrent layers. In accordance with different backbone CNN architectures, the traditional RNN in (a, c) stacks the recurrent layer on top of the last `fc` layer or `conv` layer, while our PreRNN in (b, d) makes the pre-trained CNNs recurrent by directly transforming the pre-trained `fc` layer or `conv` layer into the recurrent layer.

3. Methods

RNNs coupled with pre-trained CNNs are powerful tools to exploit the important temporal connections in visual sequence learning tasks. It is well explored in the literature [11, 34] that CNN models, pre-trained on large-scale image or video datasets, retain strong semantic and generality properties. Prior methods typically introduce a single or a stack of recurrent layers on top of the last layer[1] of a pre-trained CNN and then train the whole network together. It thus requires the entire recurrent layers to be trained from scratch, even though a pre-trained CNN is used for feature extraction. In order to maximize the representational power and generalizing capacity of pre-trained CNNs, we propose PreRNN to directly transform pre-trained convolutional (`conv`) layers or fully connected (`fc`) layers into recurrent layers. This can mitigate the difficulty of training RNNs, as we reuse parts of a pre-trained CNN as a partially pre-trained RNN. It therefore pushes the generalization ability of a pre-trained CNN onto the RNN and ultimately improves the overall performance.

PreRNN is a generic approach that can be applied to various architectures of pre-trained 2D and 3D CNNs. As illustrated in Figure 1(a, b), it transforms CNNs such as VGG [38] and C3D [43] with `fc` layers at the end of the convolutional networks, meanwhile it also converts CNNs such as ResNet [20] and DenseNet [22] with `conv` and global average pooling layers at the end, as depicted in Figure 1(c, d). PreRNN is also able to adapt to all three basic recurrent structures including VRNN, LSTM and GRU. Additionally, an alternative PreRNN-SIH can be used to simplify gating functions and reduce recurrent parameters.

[1]This denotes the last layer after removing the output layer of a pre-trained backbone CNN.

3.1. Transformations for VRNN

To be comprehensive in term of different backbone CNN architectures, we assume that the last `fc` or `conv` layer of a pre-trained CNN has the structure:

$$\boldsymbol{y} = \mathcal{H}(\boldsymbol{W}_{xy} \circ \boldsymbol{x}), \tag{4}$$

where $\boldsymbol{W}_{xy}$ are the pre-trained feedforward weights, $\boldsymbol{x}$ and $\boldsymbol{y}$ are the input and output of this layer, and $\circ$ indicates matrix multiplication for the `fc` layer or convolution operation for the `conv` layer. In order to take advantage of the pre-trained layer, we reformulate this feedforward layer as a recurrent layer using PreRNN. It is straightforward to remodel the `fc` layer through:

$$\boldsymbol{y}_t = \mathcal{H}(\boldsymbol{W}_{xy}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{y}_{t-1}), \tag{5}$$

where $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ are reformed to be the input and hidden state of this recurrent layer at time t. As for the `conv` layer, PreRNN performs the transformation by:

$$\boldsymbol{y}_t = \mathcal{H}(\mathcal{P}(\mathcal{B}(\boldsymbol{W}_{xy} * \boldsymbol{x}_t) + \boldsymbol{\gamma}_t) + \boldsymbol{W}_{hh}\boldsymbol{y}_{t-1}), \tag{6}$$

where $*$ is the convolution operation, $\mathcal{B}$ represents the batch normalization with the pre-computed mini-batch statistics, $\boldsymbol{\gamma}_t$ indicates an optional shortcut connection in residual networks, and $\mathcal{P}$ is the global average pooling.

PreRNN essentially transforms the feedforward weights $\boldsymbol{W}_{xy}$ and output $\boldsymbol{y}$ in Eq. (4) as the input-to-hidden weights

$\boldsymbol{W}_{xy}$ and hidden state $\boldsymbol{y}_t$ in Eqs. (5-6). In comparison to the traditional VRNN in Eq. (1), which introduces two randomly initialized weight matrices, PreRNN in Eqs. (5-6) only brings in a single hidden-to-hidden weight matrix $\boldsymbol{W}_{hh}$ to be trained from scratch, while the input-to-hidden weights $\boldsymbol{W}_{xy}$ inherited from Eq. (4) have been pre-trained and can be just fine-tuned. As a result, PreRNN can fully make use of the robust generalization of a pre-trained CNN and preserve its architecture to the greatest extent.

3.2. Transformations for LSTM and GRU

A prominent feature shared by LSTM and GRU is the additive nature in updating the hidden state from t to $t+1$, i.e., keep the existing state and add changes on top of it through their gating functions. This helps each hidden state unit to remember the existence of a specific feature for a long series of steps, and more importantly, to create shortcut paths to allow the error to be back-propagated easily through multiple steps without vanishing too quickly. Here we aim to extend PreRNN to accommodate the gating functions of LSTM and GRU. For this purpose, we split each gating function to two components and fuse the pre-trained feedforward layer into them.

3.2.1 Gate-Dependent Input-to-Hidden State

We follow the same principle to convert a pre-trained feedforward layer into a recurrent layer, as we did for transforming VRNN. In Eqs. (2-3) each gate[2] is composed of two components that are the input-to-hidden state and the hidden-to-hidden state. We define the gate-dependent input-to-hidden state for PreRNN as:

$$\boldsymbol{u}_t(g) = \begin{cases} \boldsymbol{W}^p_{ig}\boldsymbol{x}_t & \text{a } \texttt{fc} \text{ layer,} \\ \mathcal{P}(\mathcal{B}(\boldsymbol{W}^p_{ig} * \boldsymbol{x}_t) + \boldsymbol{\gamma}_t) & \text{a } \texttt{conv} \text{ layer,} \end{cases} \tag{7}$$

where g is a gate index, $g = \{i, f, o, c\}$ for LSTM and $g = \{r, z, h\}$ for GRU, $\boldsymbol{u}_t(g)$ is the input-to-hidden state of gate g at time t, and $\boldsymbol{W}^p_{ig}$ is the pre-trained input-to-hidden weights of gate g. Concretely, we convert the pre-trained feedforward weights $\boldsymbol{W}_{xy}$ in Eq. (4) to the input-to-hidden weights for one gate and use the pre-trained values to initialize the input-to-hidden weights for other gates. So we redefine the gating functions of LSTM in Eq. (2) as:

$$\begin{aligned} \boldsymbol{i}_t &= \text{sigm}(\boldsymbol{u}_t(i) + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1}), \\ \boldsymbol{f}_t &= \text{sigm}(\boldsymbol{u}_t(f) + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1}), \\ \boldsymbol{o}_t &= \text{sigm}(\boldsymbol{u}_t(o) + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1}), \\ \tilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{u}_t(c) + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1}), \end{aligned} \tag{8}$$

where only the hidden-to-hidden weights $\boldsymbol{W}_{h\cdot}$ are randomly initialized, and we follow the same updating functions in Eq. (2) to renew the memory cell $\boldsymbol{c}_t$ and hidden state $\boldsymbol{h}_t$. Correspondingly, the gating functions of GRU in Eq. (3) can be redefined as:

$$\begin{aligned} \boldsymbol{r}_t &= \text{sigm}(\boldsymbol{u}_t(r) + \boldsymbol{W}_{hr}\boldsymbol{h}_{t-1}), \\ \boldsymbol{z}_t &= \text{sigm}(\boldsymbol{u}_t(z) + \boldsymbol{W}_{hz}\boldsymbol{h}_{t-1}), \\ \tilde{\boldsymbol{h}}_t &= \tanh(\boldsymbol{u}_t(h) + \boldsymbol{W}_{hh}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})), \end{aligned} \tag{9}$$

and the hidden state $\boldsymbol{h}_t$ is updated in the same manner as in Eq. (3). By fusing the pre-trained feedforward layer into the input-to-hidden state of each gate, PreRNN introduces fewer input-to-hidden parameters and only needs to train the hidden-to-hidden weights from scratch.

3.2.2 Single Input-to-Hidden State

In the aforementioned transformation scheme, each gate learns its own input-to-hidden weights $\boldsymbol{W}^p_{ig}$, though they start from the same initial state $\boldsymbol{W}_{xy}$. In order to simplify the gating functions and fully utilize the pre-trained feedforward layer, we take our idea further and bind all gates to the same input-to-hidden state:

$$\boldsymbol{v}_t = \begin{cases} \boldsymbol{W}_{xy}\boldsymbol{x}_t & \text{a } \texttt{fc} \text{ layer,} \\ \mathcal{P}(\mathcal{B}(\boldsymbol{W}_{xy} * \boldsymbol{x}_t) + \boldsymbol{\gamma}_t) & \text{a } \texttt{conv} \text{ layer,} \end{cases} \tag{10}$$

where $\boldsymbol{v}_t$ is the single input-to-hidden (SIH) state that is adopted by all the gates, and we call this transformation PreRNN-SIH. Compared to the gate-dependent input-to-hidden state in Eq. (7), PreRNN-SIH directly converts the pre-trained feedforward layer to be the unified input-to-hidden state for all the gates. We thus change the gating functions of LSTM in Eq. (2) to:

$$\begin{aligned} \boldsymbol{i}_t &= \text{sigm}(\boldsymbol{v}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1}), \\ \boldsymbol{f}_t &= \text{sigm}(\boldsymbol{v}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1}), \\ \boldsymbol{o}_t &= \text{sigm}(\boldsymbol{v}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1}), \\ \tilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{v}_t + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1}), \end{aligned} \tag{11}$$

where all the gates hinge on the same input-to-hidden state $\boldsymbol{v}_t$. In the same way, the gating functions of GRU in Eq. (3) are reformulated as:

$$\begin{aligned} \boldsymbol{r}_t &= \text{sigm}(\boldsymbol{v}_t + \boldsymbol{W}_{hr}\boldsymbol{h}_{t-1}), \\ \boldsymbol{z}_t &= \text{sigm}(\boldsymbol{v}_t + \boldsymbol{W}_{hz}\boldsymbol{h}_{t-1}), \\ \tilde{\boldsymbol{h}}_t &= \tanh(\boldsymbol{v}_t + \boldsymbol{W}_{hh}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})). \end{aligned} \tag{12}$$

Hence, PreRNN-SIH in Eqs. (11-12) only introduces the hidden-to-hidden weights $\boldsymbol{W}_{h\cdot}$ that need to be trained from scratch. In addition, since the pre-trained feedforward layer is set to be the joint input-to-hidden state for all the gating functions of LSTM and GRU, PreRNN-SIH can therefore significantly reduce the number of recurrent parameters and consequently the computational cost.

[2]For notational simplicity, we also call LSTM's new memory state $\tilde{\boldsymbol{c}}_t$ and GRU's candidate hidden state $\tilde{\boldsymbol{h}}_t$ gate here.

Applications	Sequences	CNNs	Datasets	Objectives
Sequential Face Alignment	Color	VGG16 [38]	300VW [7]	ℓ_2
Hand Gesture Recognition	Color & Depth	C3D [43]	NVGesture [28]	CTC [15]
Action Recognition	Color & Flow	ResNet50 [20]	UCF101 [39]	NLL

Table 1. Summary of the diverse experiments conducted in this paper in terms of applications, visual sequence types, pre-trained backbone CNNs, benchmark datasets, and objective functions.

	Traditional		PreRNN			PreRNN-SIH		
	1 layer	2 layers	`fc6`	`fc7`	`fc6/7`	`fc6`	`fc7`	`fc6/7`
VRNN	0.704	0.716	0.757	0.742	**0.763**	-	-	-
LSTM	0.718	0.671	**0.769**	0.754	0.746	0.743	0.746	0.719
GRU	0.722	0.698	**0.772**	0.755	0.761	0.768	0.748	0.762

Table 2. AUC of the traditional RNNs and our proposed PreRNN and PreRNN-SIH on the 300VW dataset.

4. Applications

In this section, we exemplify our proposed methods for visual sequence learning with three applications: sequential face alignment, dynamic hand gesture recognition, and action recognition. As summarized in Table 1, our evaluations represent a large diversity of visual sequences, pre-trained backbone CNNs, benchmark datsets, and objective functions. To compare the performance of each basic recurrent structure in a controlled setting, we carefully choose the hidden state dimensions of different recurrent models so that the total number of parameters in each case is as close as possible (see supplementary material for more details). We train the models using mini-batch stochastic gradient descent with momentum, and implement our networks in Theano and PyTorch on an NVIDIA DGX-1. In the following, we use PreVRNN, PreLSTM and PreGRU to indicate the three basic recurrent structures created by the proposed PreRNN, and denote the ones constructed by the traditional RNNs as TraVRNN, TraLSTM and TraGRU.

4.1. Sequential Face Alignment

We start from the video based face alignment, which is fundamental to many applications such as face recognition, expression analysis, facial animation capturing, etc. We experiment on the benchmark dataset 300VW [7], which contains 114 videos and 218,595 frames in total, with 68 annotated facial landmarks per frame. We follow the same experimental setting as [19] to split the dataset into 80% for training and 20% for testing. A Faster R-CNN [35] based face detector is used as a preprocess to detect the facial region on each frame.

We employ the pre-trained VGG16 [38] on ImageNet [9] as the backbone CNN and the ℓ_2 loss as our objective function, and change the output layer to 136 units corresponding to the locations of 68 facial landmarks. We use the same evaluation metric, i.e., area under the curve (AUC) for quantitative performance comparison. AUC is the area under the cumulative error distribution curve (see Figure 3), which describes the proportion of frames with the normalized point-to-point error less than a given threshold.

Since there are two `fc` layers in VGG16, we first transform both of them (`fc6` and `fc7`) into recurrent layers with PreRNN. As a comparison, we follow the traditional RNNs to build two recurrent layers on top of `fc7` in VGG16. As shown in Table 2, PreRNN with `fc6/7` significantly outperforms the traditional RNNs with 2 layers for the three basic recurrent structures.

Next we investigate the internal mechanism of traditional RNNs and PreRNN to better understand the source of their performances and shortcomings, similar to the analysis for language modeling in [24]. Specifically, we look into the distributions of gate activations and define a gate unit to be left or right saturated if its activation is less than 0.1 or more than 0.9, or unsaturated otherwise. We then infer the gating mechanism through the saturation plots or the activation histograms as shown in Figure 2. Our consistent finding is that the activations in the first layer of PreLSTM lie in the more saturated regime (closer to the diagonal) than those of TraLSTM. This implies that PreLSTM is more capable to utilize the temporal context, e.g., the multiple frequently right saturated forget gate units (bottom right of the forget gate plot) correspond to the memory cells that remember their values for long durations. Conversely, the activations of TraLSTM are dispersed in the more unsaturated regime, indicating that the integrated temporal information decays rapidly. We make a similar observation for the first layer of PreGRU where the left saturated (0.0-0.1) and right saturated (0.9-1.0) bins dominate the distribution, whereas the activations of TraGRU gather in the unsaturated bins.

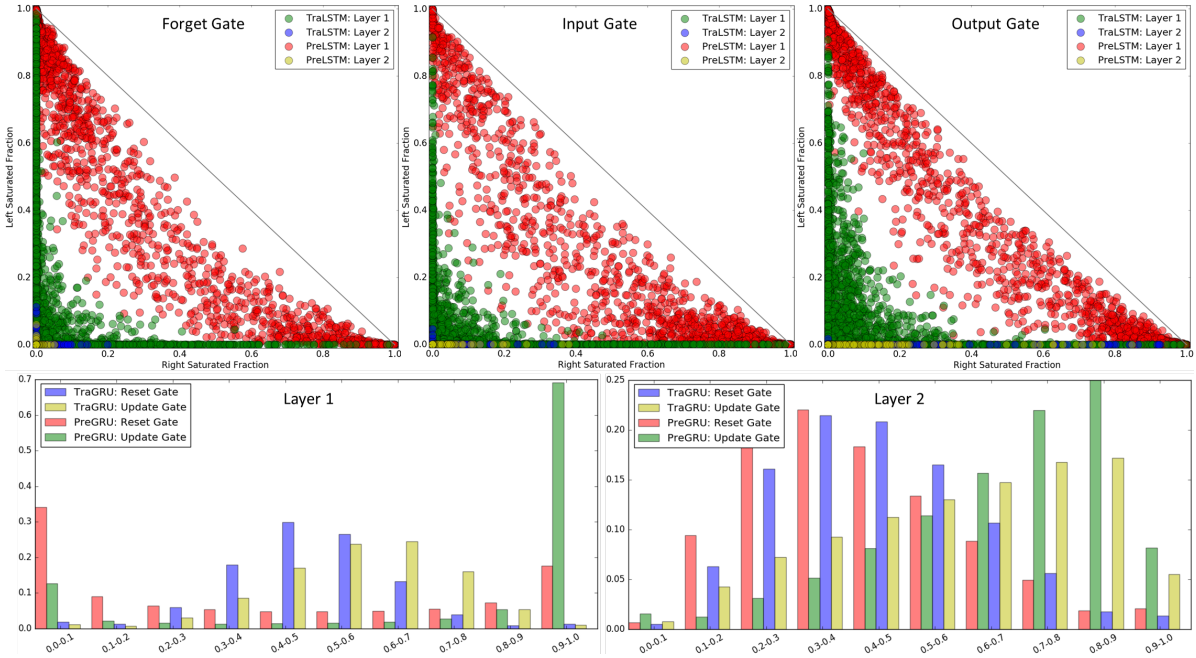

Figure 2. Examples of the gate activation distribution for LSTM and GRU. Top: saturation plots of the fraction of times that each gate unit is left or right saturated for LSTM. Bottom: activation histograms over 10 bins for GRU. This figure is best viewed on screen.

It is also interesting to note that the activations in the second layer of both TraLSTM and PreLSTM concentrate near the origin in the saturation plots, where the gate units are rarely left or right saturated. This suggests that the second recurrent layer virtually functions in a feedforward fashion and the preceding hidden state is barely used. A similar phenomenon is also shown in the activation histogram for the second layer of TraGRU and PreGRU. We take this observation as a guidance to determine the hierarchy of recurrent layers: we transform only one `fc` layer (either `fc6` or `fc7`) into a recurrent layer and leave the other one as a feedforward layer for PreRNN, and correspondingly build only one recurrent layer for the traditional RNNs. Table 2 clearly shows the improvements of PreRNN with `fc6` and the traditional RNNs with 1 layer over their 2-layer counterparts for both LSTM and GRU.

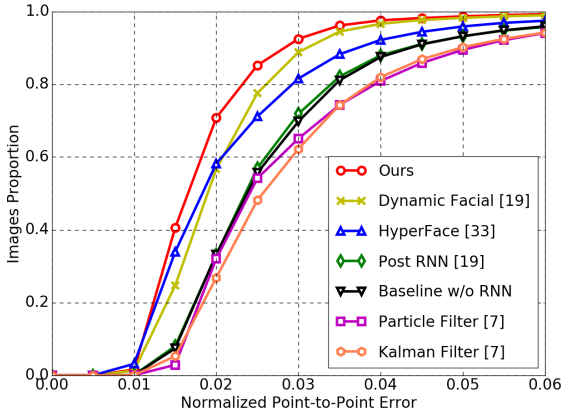

Figure 3. Comparison of our approach with the state-of-the-art methods on the 300VW dataset.

PreRNN-SIH substantially reduces the recurrent parameters as shown in Table 3, yet, it still outperforms traditional RNNs and compares favorably with PreRNN. As for comparing the basic recurrent structures, GRU performs slightly better than LSTM, which further moderately outperforms VRNN. We finally show the cumulative error distributions of our approach and the competing algorithms in Figure 3, where ours (PreGRU) outperforms the other methods.

4.2. Dynamic Hand Gesture Recognition

Our second application is the online dynamic hand gesture recognition, which is a natural and important form for human computer interaction. This is a challenging task and requires to simultaneously detect and classify the inprogress gestures from unsegmented input streams. We experiment with the public benchmark dataset NVGesture [28], which contains 25 hand gesture categories and 1,532 videos captured with multiple sensors. Our experiments are based on the color and depth modalities. We comply with the standard evaluation protocol to split the dataset by subject into 70% for training and 30% for testing.

We leverage on the method developed in [28] to use the C3D [43] pre-trained on Sports1M [25] as the base CNN and the connectionist temporal classification (CTC) [15] as the loss function. CTC is an objective function proposed for

	VGG16			C3D			ResNet50
	fc6	fc7	fc6/7	fc6	fc7	fc6/7	
LSTM	0.18×	0.57×	0.27×	0.40×	0.57×	0.47×	0.84×
GRU	0.20×	0.60×	0.30×	0.43×	0.60×	0.50×	0.86×

Table 3. Number of recurrent parameters used by PreRNN-SIH compared to those by traditional RNNs or PreRNN based on the different pre-trained backbone CNNs.

	Traditional		PreRNN			PreRNN-SIH		
	1 layer	2 layers	fc6	fc7	fc6/7	fc6	fc7	fc6/7
VRNN	83.3%	80.8%	81.9%	82.9%	**84.4%**	-	-	-
LSTM	81.3%	81.3%	81.7%	81.9%	82.7%	80.0%	81.7%	**84.2%**
GRU	81.9%	82.5%	82.1%	81.0%	83.1%	**84.4%**	79.8%	83.8%

Table 4. Classification accuracy of the traditional RNNs and our proposed PreRNN and PreRNN-SIH on the NVGesture dataset.

speech recognition to label unsegmented audio sequence. It is applied in this task to support predicting gestures from the unsegmented color and depth streams.

We transform the fc layers of C3D (fc6 and or fc7) into recurrent layers with PreRNN and PreRNN-SIH. As a comparison, we construct recurrent layers after fc7 for the traditional RNNs. Table 4 demonstrates that both PreRNN and PreRNN-SIH outperform the traditional RNNs, especially for LSTM and GRU. PreRNN-SIH yields superior performance and also significantly reduces the number of parameters required by recurrent layers by more than half, as shown in Table 3.

In addition to improving the classification accuracy, PreRNN is also found to converge faster than the traditional RNNs during training. Figure 4 demonstrates the training curves of different networks. As shown in this figure, PreVRNN greatly expedites the training process and reduces overfitting. PreLSTM also exhibits faster convergence than TraLSTM (see the slope at the early training stage). We attribute the faster convergence of our approach to fusing the pre-trained feedforward layers into recurrent layers so that our RNNs are partially pre-trained and therefore they can accelerate the convergence.

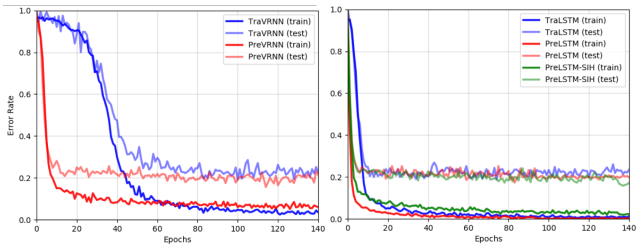

Figure 4. Comparison of the training processes between the traditional RNNs and our proposed PreRNN and PreRNN-SIH for VRNN (left) and LSTM (right).

Comparing the basic recurrent structures, we observe that VRNN is on a par with LSTM and GRU on this dataset. We hypothesize this is due to approximately 20% of the gesture categories, e.g., "thumb up" and "index finger", being almost still and lacking a strong temporal dependency. We then compare our approach with the state-of-the-art methods in Table 6. We achieve superior results on each individual modality and the combination of them, and significantly outperform the baseline by 4.0%. Notably, our approach using the single depth modality already produces an accuracy that is better than other competitors by a clear margin, and even consistently outperforms the ones that combine more modalities, highlighting the advantage of PreRNN to make use of the temporal connections.

4.3. Action Recognition

We also apply our approach to model the dynamic motions for action recognition, which plays an important role in surveillance event detection, content based video search, health care monitoring, etc. We experiment on the public benchmark dataset UCF101 [39], which consists of 101 action classes and 13,320 videos in total. We use the standard three training and testing splits as in [39] to perform our evaluations, and report results on the first split for our internal comparisons.

We follow the original two-stream method [37] to adopt two separate CNNs to operate on the spatial (color) and temporal (optical flow) streams. Our temporal network employs the pre-computed optical flow [50] stacking with 10 frames. CNNs of each stream in our approach are then equipped with RNNs to capture the temporal dynamics over the whole video sequence. We use the ResNet50 model [20] pre-trained on ImageNet [9] as the backbone CNN and the negative log likelihood (NLL) as the loss function.

	Traditional			PreRNN			PreRNN-SIH		
	Color	Flow	Comb	Color	Flow	Comb	Color	Flow	Comb
VRNN	82.9%	83.6%	91.6%	83.8%	84.6%	**92.7%**	-	-	-
LSTM	83.4%	84.0%	92.5%	85.3%	84.8%	93.2%	85.0%	84.6%	**93.5%**
GRU	83.6%	83.8%	92.2%	84.3%	85.2%	**93.7%**	84.9%	84.7%	93.3%

Table 5. Classification accuracy of the traditional RNNs and our proposed PreRNN and PreRNN-SIH on the first split of UCF101 dataset.

Method	Modality	Accuracy
C3D [43]	Color	69.3%
R3DCNN [28]	Color	74.1%
Ours	Color	**76.5%**
SNV [49]	Depth	70.7%
C3D [43]	Depth	78.8%
R3DCNN [28]	Depth	80.3%
Ours	Depth	**84.4%**
Two-Stream [37]	Color + Flow	65.6%
iDT [45]	Color + Flow	73.4%
R3DCNN [28]	Five Modalities	83.8%
Baseline (w/o RNN)	Color + Depth	81.0%
Ours	Color + Depth	**85.0%**

Table 6. Comparison of our approach with the state-of-the-art methods on the NVGesture dataset, which consists of five modalities including color, depth, optical flow, IR image and IR disparity.

Method	Accuracy
Dynamic Image Nets [3]	76.9%
Long-Term Recurrent ConvNet [10]	82.9%
Composite LSTM Model [40]	84.3%
C3D [43]	85.2%
iDT [45]	86.4%
Two-Stream ConvNet [37]	88.0%
Multilayer Multimodal Fusion [48]	91.6%
Long-Term ConvNets [44]	91.7%
Two-Stream Fusion [14]	92.5%
Spatiotemporal ResNets [12]	93.4%
Inflated 3D ConvNets [4]	93.4%
Temporal Segment Networks [46]	**94.2%**
Spatiotemporal Multiplier Nets [13]	**94.2%**
Baseline (w/o RNN)	91.7%
Ours	**94.3%**

Table 7. Comparison of our approach with the state-of-the-art methods on the average of three splits of UCF101 dataset.

We transform the last `conv` layer of ResNet50 into a recurrent layer with PreRNN and PreRNN-SIH. As defined in Eqs. (6, 7, 10), we fuse the pre-trained weights and mini-batch statistics of `res5c-branch2c` as well as the shortcut connection from `res5b` into the recurrent layer. As a comparison, traditional RNNs build a recurrent layer on top of `pool5` in ResNet50. Table 5 demonstrates that PreRNN and PreRNN-SIH both outperform traditional RNNs by up to 1.9% and 1.4% on the color and optical flow streams, respectively. Combining the two streams through simple averaging softmax scores boosts the classification results for all methods. Apart from improving the accuracy, PreRNN-SIH reduces the recurrent parameters by around 15%, as shown in Table 3. In comparison, among the three basic recurrent structures, LSTM produce similar results to GRU, which both outperform VRNN.

We finally compare with the most recent competing algorithms in Table 7, where our approach achieves the state-of-the-art classification accuracy 94.3%, which is 2.6% improvement over the baseline. In particular, we note that [13, 46] also produce the competitive results. However, [13] employs a more powerful CNN (i.e., ResNet152) to the temporal stream, and [46] relies on two more input modalities (i.e., warped flow in iDT and difference images). More importantly, these methods are specifically designed for action recognition, while our approach is generic for various visual sequence learning problems and can be potentially combined with other methods to obtain further gains.

5. Conclusion

In this paper, we have proposed PreRNN and PreRNN-SIH to make pre-trained CNNs recurrent for visual sequence learning by directly transforming pre-trained feed-forward layers into recurrent layers. Our approach fits for all basic recurrent structures and various architectures of CNNs. Extensive experiments on three applications find PreRNN and PreRNN-SIH to produce consistently better results than traditional RNNs, in addition to a significant reduction of recurrent parameters by PreRNN-SIH. This clearly shows that our method is not just geared to a particular dataset but is generally applicable to different visual sequence learning tasks. We also provide the insight of understanding the internal gating mechanism and demonstrate that this can be used to improve the design of recurrent architecture. In the future work, we intend to explore and apply our method to more visual sequence learning problems such as sequential human pose estimation, semantic video segmentation, and multi-frame optical flow estimation.

References

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

[2] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *TNN*, 1994.

[3] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, 2016.

[4] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *CVPR*, 2017.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015.

[6] K. Cho, B. Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv:1412.3555*, 2014.

[7] G. Chrysos, E. Antonakos, P. Snape, A. Asthana, and S. Zafeiriou. A comprehensive performance evaluation of deformable face tracking "in-the-wild". *IJCV*, 2017.

[8] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Workshop*, 2014.

[9] J. Deng, W. Dong, R. Socher, J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[10] 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. In *CVPR*, 2015.

[11] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

[12] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016.

[13] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal multiplier networks for video action recognition. In *CVPR*, 2017.

[14] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016.

[15] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006.

[16] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *TPAMI*, 2009.

[17] A. Graves, R. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013.

[18] K. Greff, R. Srivastava, B. Steunebrink, and J. Schmidhuber. LSTM: A search space odyssey. *TNNLS*, 2016.

[19] J. Gu, X. Yang, S. De Mello, and J. Kautz. Dynamic facial analysis: From Bayesian filtering to recurrent neural network. In *CVPR*, 2017.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[21] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[22] G. Huang, Z. Liu, K. Weinberger, and L. Maaten. Densely connected convolutional networks. In *CVPR*, 2017.

[23] R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In *ICML*, 2015.

[24] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. In *ICLR Workshop*, 2016.

[25] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[26] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[27] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

[28] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. Online detection and classification of dynamic hand gestures with recurrent 3D convolutional neural networks. In *CVPR*, 2016.

[29] Y. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[30] X. Pang, R. Feris, X. Wang, and D. Metaxas. A recurrent encoder-decoder network for sequential face alignment. In *ECCV*, 2016.

[31] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks. In *ICLR*, 2014.

[32] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013.

[33] R. Ranjan, V. Patel, and R. Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *TPAMI*, 2017.

[34] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *CVPR Workshop*, 2014.

[35] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[36] K. Schindler and L. Van Gool. Action snippets: How many frames does human action recognition require. In *CVPR*, 2008.

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

[38] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[39] K. Soomro, A. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402*, 2012.

[40] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015.

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[42] P. Tokmakov, K. Alahari, and C. Schmid. Learning video object segmentation with visual memory. In *ICCV*, 2017.

[43] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: Generic features for video analysis. In *ICCV*, 2015.
[44] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *TPAMI*, 2017.
[45] H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A robust and efficient video representation for action recognition. *IJCV*, 2015.
[46] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Val Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.
[47] J. Weston, S. Chopra, and A. Bordes. Memory networks. In *ICLR*, 2015.
[48] X. Yang, P. Molchanov, and J. Kautz. Multilayer and multimodal fusion of deep neural networks for video classification. In *ACM Multimedia*, 2016.
[49] X. Yang and Y. Tian. Super normal vector for human activity recognition with depth cameras. *TPAMI*, 2017.
[50] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *DAGM*, 2007.
[51] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.
[52] Z. Zhou, Y. Huang, W. Wang, L. Wang, and T. Tan. See the forest for the trees: Joint spatial and temporal recurrent neural networks for video-based person re-identification. In *CVPR*, 2017.