Adding Attentiveness to the Neurons in Recurrent Neural Networks

Pengfei Zhang[1 [0000−0002−8303−8930]], Jianru Xue[1⋆ [0000−0002−4994−9343]], Cuiling Lan[2⋆ [0000−0001−9145−9957]], Wenjun Zeng[2 [0000−0003−2531−3137]], Zhanning Gao[1 [0000−0003−2031−2805]], and Nanning Zheng[1 [0000−0003−1608−8257]]

[1] Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
[2] Microsoft Reserach Asia
zpengfei@stu.xjtu.edu.cn, {culan,wezeng}@microsoft.com,
{jrxue,nnzheng}@mail.xjtu.edu.cn, zhanninggao@gmail.com

Abstract. Recurrent neural networks (RNNs) are capable of modeling the temporal dynamics of complex sequential information. However, the structures of existing RNN neurons mainly focus on controlling the contributions of current and historical information but do not explore the different importance levels of different elements in an input vector of a time slot. We propose adding a simple yet effective Element-wise-Attention Gate (EleAttG) to an RNN block (e.g., all RNN neurons in a network layer) that empowers the RNN neurons to have the attentiveness capability. For an RNN block, an EleAttG is added to adaptively modulate the input by assigning different levels of importance, *i.e.*, attention, to each element/dimension of the input. We refer to an RNN block equipped with an EleAttG as an EleAtt-RNN block. Specifically, the modulation of the input is content adaptive and is performed at fine granularity, being element-wise rather than input-wise. The proposed EleAttG, as an additional fundamental unit, is general and can be applied to any RNN structures, *e.g.*, standard RNN, Long Short-Term Memory (LSTM), or Gated Recurrent Unit (GRU). We demonstrate the effectiveness of the proposed EleAtt-RNN by applying it to the action recognition tasks on both 3D human skeleton data and RGB videos. Experiments show that adding attentiveness through EleAttGs to RNN blocks significantly boosts the power of RNNs.

Keywords: Element-wise-Attention Gate (EleAttG) · recurrent neural networks · action recognition · skeleton · RGB video

1 Introduction

In recent years, recurrent neural networks [25], such as standard RNN (sRNN), its variant Long Short-Term Memory (LSTM) [15], and Gated Recurrent Unit (GRU) [3], have been adopted to address many challenging problems with sequential time-series data, such as action recognition [9], machine translation [2],

⋆ Corresponding author.

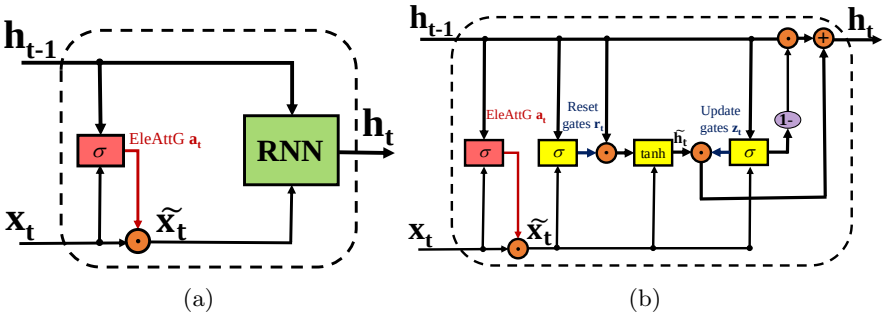


Fig. 1: Illustration of Element-wise-Attention Gate (EleAttG) (marked in red) for (a) a generic RNN block, where the RNN structure could be the standard RNN, LSTM, or GRU and (b) a GRU block which consists of a group of (*e.g.,* N) GRU neurons. In the diagram, each line carries a vector. The brown circles denote element-wise operation, *e.g.,* element-wise vector product or vector addition. The yellow boxes denote the units of the original GRU with the output dimension of N. The red box denotes the EleAttG with an output dimension of D, which is the same as the dimension of the input $\mathbf{x_t}$.

and image caption [39]. They are powerful in exploring temporal dynamics and learning appropriate feature representations.

The structure of recurrent neural networks facilitates the processing of sequential data. RNN neurons perform the same task at each step, with the output being dependent on the previous output, *i.e.*, some historical information is memorized. Standard RNNs have difficulties in learning long-range dependencies due to the vanishing gradient problem [19]. The LSTM [15] or GRU [3] architectures combat vanishing gradients through a gating mechanism. Gates provide a way to optionally let information through or stop softly, which balances the contributions of the information of the current time slot and historical information. There are some variants of RNNs with slightly different designs [19]. Note a gate applies a single scaling factor to control the flow of the embedded information (as a whole) of the input rather than imposing controls on each element of the input. They are not designed to explore the potential different characteristics of the input elements.

Attention mechanisms which selectively focus on different parts of the data have been demonstrated to be effective for many tasks [28,36,49,23,31,46]. These inspire us to develop an Element-wise-Attention Gate (EleAttG) to augment the capability of RNN neurons. More specifically, for an RNN block, an EleAttG is designed to output an attention vector, with the same dimension as the input, which is then used to modulate the input elements. Note that similar to [29], we use an RNN block to represent an ensemble of N RNN neurons, which for example could be all the RNN neurons in an RNN layer. Fig. 1 (a) illustrates the EleAttG within a generic RNN block. Fig. 1 (b) shows a specific case when the RNN structure of GRU is used. The input $\mathbf{x_t}$ is first modulated by the response

of the EleAttG to output $\widetilde{\mathbf{x_t}}$ before other operations are applied to the RNN block. We refer to an RNN block equipped with an EleAttG as EleAtt-RNN block. Depending on the underlying RNN structure used (*e.g.*, standard RNN, LSTM, GRU), the newly developed EleAtt-RNN will also be denoted as EleAtt-sRNN, EleAtt-LSTM, or EleAtt-GRU. An RNN layer with such EleAttG can replace the original RNN layer and multiple EleAtt-RNN layers can be stacked.

We demonstrate the effectiveness of the proposed EleAtt-RNN by applying it to action recognition. Specifically, for 3D skeleton-based human action recognition, we build our systems by stacking several EleAtt-RNN layers, using standard RNN, LSTM and GRU, respectively. EleAtt-RNNs consistently outperform the original RNNs for all the three types of RNNs. Our scheme based on EleAtt-GRU achieves state-of-the-art performance on three challenging datasets, *i.e.,* the NTU [30], N-UCLA [43], and SYSU [16] datasets. For RGB-based action recognition, we design our system by applying an EleAtt-GRU network to the sequence of frame-level CNN features. Experiments on both the JHMDB [18] and NTU [30] datasets show that adding EleAttGs brings significant gain.

The proposed EleAttG has the following merits. First, EleAttG is capable of adaptively modulating the input at a fine granularity, paying different levels of attention to different elements of the input, resulting in faster convergence in training and higher performance. Second, the design is very simple. For an RNN layer, only one line of code needs to be added in implementation. Third, the EleAttG is general and can be added to any underlying RNN structure, *e.g.*, standard RNN, LSTM, GRU, and to any layer.

2 Related work

2.1 Recurrent Neural Networks

Recurrent neural networks have many different structures. In 1997, to address the vanishing gradient problem of standard RNN, Hochreiter *et al.* proposed LSTM, which introduces a memory cell that allows "constant error carrousels" and multiplicative gate units that learn to open and close access to the constant error flow [15]. Gers *et al.* made improvement by adding the "forget gate" that enables an LSTM cell to learn to reset itself (historical information) at appropriate times to prohibit the growth of the state indefinitely [11]. A variant of LSTM is the peephole LSTM, which allows the gates to access the cell [12]. GRU, which was proposed in 2014, is a simpler variant of LSTM. A GRU has a reset gate and an update gate which control the memory and the new input information. Between the LSTMs and GRUs, there is no clear winner [6,19]. For LSTM, a differential gating scheme is proposed in [37] which leverages the derivative of the cell state to gate the information flow. Its effectiveness is demonstrated on action recognition.

In this work, we address the capability of RNNs from a new perspective. We propose a simple yet effective Element-wise-Attention Gate which adaptively modulates the input elements to explore their different importances for an RNN block.

2.2 Attention Mechanisms

Attention mechanisms which selectively focus on different parts of the data have been proven effective for many tasks such as machine translation [28,36], image caption [49], object detection [23], and action recognition [31,46].

Luong *et al.* examine some simple attention mechanisms for neural machine translation. At each time step, the model infers the attention weights and uses them to average the embedding vectors of the source words [28]. For image caption, Xu *et al.* split an image into L parts with each part described by a feature vector. To allow the decoder which is built by LSTM blocks to selectively focus on certain parts of the image, the weighted average of all the feature vectors using learned attention weights is fed to the LSTM network at every time step [49]. A similar idea is used for RGB-based action recognition in [31]. The above attention models focus on how to average a set of feature vectors with suitable weights to generate a pooled vector of the same dimension as the input of RNN. They do not consider the fine-grained adjustment based on different levels of importance across the input dimensions. In addition, they address attention at the network level, but not RNN block level.

For skeleton-based action recognition, a global context-aware attention is proposed to allocate different levels of importance to different joints of different input frames [27]. Since the global information of a sequence is required to learn the attention, the system suffers from time delay. Song *et al.* propose a spatio-temporal attention model without requiring global information [33]. Before the main recognition network, a spatial attention subnetwork is added which modulates the skeleton input to selectively focus on discriminative joints at each time slot. However, their design is not general and has not been extended to higher RNN layers. In contrast, our proposed enhanced RNN, with EleAttG embedded as a fundamental unit of RNN block, is general, simple yet effective, which can be applied to any RNN block/layer.

2.3 Action Recognition

For action recognition, many studies focus on recognition from RGB videos [40,32,8,50,44]. In recent years, 3D skeleton based human action recognition has been extensively studied and has been attracting increasing attention, thanks to its high level representation [13]. Many traditional approaches focus on how to design efficient features to solve the problems of small inter-class variation, large view variations, and the modeling of complicated spatial and temporal evolution [40,43,38,47,42,45].

For 3D skeleton based human action recognition, RNN-based approaches have been attractive due to their powerful spatio-temporal dynamic modeling ability. Du *et al.* propose a hierarchical RNN model with the hierarchical body partitions as input to different RNN layers [9]. To exploit the co-occurrence of discriminative joints of skeletons, Zhu *et al.* propose a deep regularized LSTM networks with group sparse regularization [52]. Shahroudy *et al.* propose a part-aware LSTM network by separating the original LSTM cell into five sub-cells

corresponding to five major groups of human body [30]. Liu *et al.* propose a spatio-temporal LSTM structure to explore the contextual dependency of joints in spatio-temporal domains [26]. Li *et al.* propose an RNN tree network with a hierarchical structure which classifies the action classes that are easier to distinguish at the lower layers and the action classes that are harder to distinguish at higher layers [24]. To address the large view variation of the captured data, Zhang *et al.* propose a view adaptive subnetwork which automatically selects the best observation viewpoints within an end-to-end network for recognition [51].

For RGB-based action recognition, to exploit the spatial correlations, convolutional neural networks are usually used to learn the features [32,44,50,8]. Some approaches explore the temporal dynamics of the sequential frames by simply averaging/multiplying the scores/features of the frames for fusion [32,44,7]. Some other approaches leverage RNNs to model temporal correlations, with frame-wise CNN features as input at every time slot [50,8].

Our proposed EleAttG is a fundamental unit that aims to enhance the capability of an RNN block. We will demonstrate its effectiveness in both 3D skeleton based action recognition and RGB-based action recognition.

3 Overview of Standard RNN, LSTM, and GRU

Recurrent neural networks are capable of modeling temporal dynamics of a time sequence. They have a "memory" which captures historical information accumulated from previous time steps. To better understand the proposed EleAttG and its generalization capability, we briefly review the popular RNN structures, *i.e.,* standard RNN, Long Short Term Memory (LSTM) [15], and Gated Recurrent Unit (GRU) [3].

For a standard RNN layer, the output response $\mathbf{h_t}$ at time t is calculated based on the input $\mathbf{x_t}$ to this layer and the output $\mathbf{h_{t-1}}$ from the previous time slot

$$\mathbf{h}_t = \tanh\left(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h\right), \tag{1}$$

where $\mathbf{W}_{\alpha\beta}$ denotes the matrix of weights between α and β, $\mathbf{b_h}$ is the bias vector.

The standard RNN suffers from the gradient vanishing problem due to insufficient, decaying error back flow [15]. LSTM allevates this problem by enforcing constant error flow through "constant error carrousels" within the cell unit c_t. The input gate i_t, forget gate f_t and output gate o_t learn to open and close access to the constant error flow. For an LSTM layer, the recursive computations of activations of the units are

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i\right), \\
\mathbf{f}_t &= \sigma\left(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f\right), \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c), \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o\right), \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right),
\end{aligned} \tag{2}$$

where $\odot$ denotes an element-wise product. Note that $\mathbf{i_t}$ is a vector denoting the responses of a set of input gates of all the LSTM neurons in the layer.

GRU is an architecture that is similar to but much simpler than that of LSTM. A GRU has two gates, reset gate r_t and update gate z_t. When the response of the reset gate is close to 0, the hidden state h'_t is forced to ignore the previous hidden state and reset with the current input only. The update gate controls how much information from the previous hidden state will be carried over to the current hidden state h_t. The hidden state acts in a way similar to the memory cell in LSTM. For a GRU layer, the recursive computations of activations of the units are

$$\begin{aligned}
\mathbf{r}_t &= \sigma\left(\mathbf{W}_{xr}\mathbf{x}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_r\right), \\
\mathbf{z}_t &= \sigma\left(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_z\right), \\
\mathbf{h}'_t &= \tanh\left(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h\right), \\
\mathbf{h}_t &= \mathbf{z_t} \odot \mathbf{h}_{t-1} + (\mathbf{1} - \mathbf{z_t}) \odot \mathbf{h}'_t.
\end{aligned} \tag{3}$$

For all the above designs, we note that the gates can control the information flow. However, the controlling of the flow takes the input $\mathbf{x_t}$ as a whole without adaptively treating different elements of the input differently.

4 Element-wise-Attention Gate for an RNN Block

For an RNN block, we propose an Element-wise-Attention Gate (EleAttG) to enable the RNN neurons to have the attentiveness capability. The response of an EleAttG is a vector $\mathbf{a_t}$ with the same dimension as the input $\mathbf{x_t}$ of the RNNs, which is calculated as

$$\mathbf{a_t} = \phi\left(\mathbf{W_{xa}}\mathbf{x_t} + \mathbf{W_{ha}}\mathbf{h_{t-1}} + \mathbf{b_a}\right), \tag{4}$$

where ϕ denotes the activation function of Sigmoid, *i.e.,* $\phi(s) = 1/(1+e^{-s})$. $\mathbf{W}_{\alpha\beta}$ denotes the matrix of weights between α and β, and $\mathbf{b_a}$ denotes the bias vector. The current input $\mathbf{x_t}$ and the hidden states $\mathbf{h_{t-1}}$ are used to determine the levels of importance of each element of the input $\mathbf{x_t}$.

The attention response modulates the input to have an updated input $\widetilde{\mathbf{x_t}}$ as

$$\widetilde{\mathbf{x_t}} = \mathbf{a_t} \odot \mathbf{x_t}. \tag{5}$$

The recursive computations of activations of the other units in the RNN block are then based on the updated input $\widetilde{\mathbf{x_t}}$, instead of the original input $\mathbf{x_t}$, as illustrated in Fig. 1.

For a standard RNN block with EleAttG (denoted as EleAtt-sRNN), the output responses $\mathbf{h_t}$ at time t are calculated as

$$\mathbf{h}_t = \tanh\left(\mathbf{W}_{xh}\widetilde{\mathbf{x_t}} + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h\right). \tag{6}$$

Similarly, for an EleAtt-GRU block, the recursive computations of activations of the units are

$$\begin{aligned}
\mathbf{r}_t &= \sigma\left(\mathbf{W}_{xr}\widetilde{\mathbf{x_t}} + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_r\right), \\
\mathbf{z}_t &= \sigma\left(\mathbf{W}_{xz}\widetilde{\mathbf{x_t}} + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_z\right), \\
\mathbf{h}'_t &= \tanh\left(\mathbf{W}_{xh}\widetilde{\mathbf{x_t}} + \mathbf{W}_{hh}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h\right), \\
\mathbf{h}_t &= \mathbf{z_t} \odot \mathbf{h}_{t-1} + (\mathbf{1} - \mathbf{z_t}) \odot \mathbf{h}'_t.
\end{aligned} \tag{7}$$

The computations for an EleAtt-LSTM block can be obtained similarly.

Most attention designs use Softmax as the activation function such that the sum of the attention values is 1 [28,36,49,23,31,46,33]. In our design, we relax this sum-to-1 constraint by using the Sigmoid activation function, with response values ranging from 0 to 1. If the sum-to-1 constraint is not relaxed, the attention responses of the k^{th} element will be affected by the changes of other elements' response values even when the levels of importance of this element are the same over consecutive time slots.

Note that in our design, an EleAttG is shared by all neurons in an RNN block/layer (see (5) and (6) for the standard RNN block, (5) and (7) for the GRU block). Theoretically, each RNN neuron (instead of block) can have its own attention gate at the cost of increased computation complexity and a larger number of parameters. We focus on the shared design in this work.

5 Experiments

We perform comprehensive studies to evaluate the effectiveness of our proposed EleAtt-RNN with EleAttG by applying it to action recognition from 3D skeleton data and RGB video, respectively.

To demonstrate the generalization capability of EleAttG, we add EleAttG to the standard RNN, LSTM, and GRU structures, respectively.

For 3D skeleton based action recognition, we use three challenging datasets, *i.e.,* the NTU RGB+D dataset (NTU) [30], the Northwestern-UCLA dataset (N-UCLA) [43], and the SYSU Human-Object Interaction dataset (SYSU)[16]. The NTU is currently the largest dataset with diverse subjects, various viewpoints and small inter-class differences. Therefore, in-depth analyses are performed on the NTU dataset. For RGB-based action recognition, we take the CNN features extracted from existing, pre-trained models without finetuning on our datasets as the input to the RNN based recognition networks and evaluate the effectiveness of EleAttG on the NTU and the JHMDB datasets [18]. We conduct most of our experiments based on GRU here, as it has simpler structure than LSTM and better performance than standard RNN.

5.1 Datasets

NTU RGB+D Dataset (NTU) [30]. NTU is currently the largest RGB+D+Skeleton dataset for action recognition, including 56880 videos of in total

more than 4 million frames. There are 60 action classes performed by different subjects. Each subject has 25 body joints and each joint has 3D coordinates. Three cameras placed in different positions are used to capture the data at the same time. We follow the standard protocols proposed in [30] including the Cross Subject (CS) and Cross View (CV) settings. For the CS setting, 40 subjects are equally split into training and testing groups. For the CV setting, the samples of cameras 2 and 3 are used for training while those of camera 1 are for testing.

Northwestern-UCLA dataset (N-UCLA) [43]. N-UCLA is a small RGB+D+Skeleton dataset including 1494 sequences which records 10 actions performed by 10 subjects. 20 joints with 3D coordinates are provided in this dataset. Following [43], we use samples from the first two cameras as training data, and the samples from the third camera as testing data.

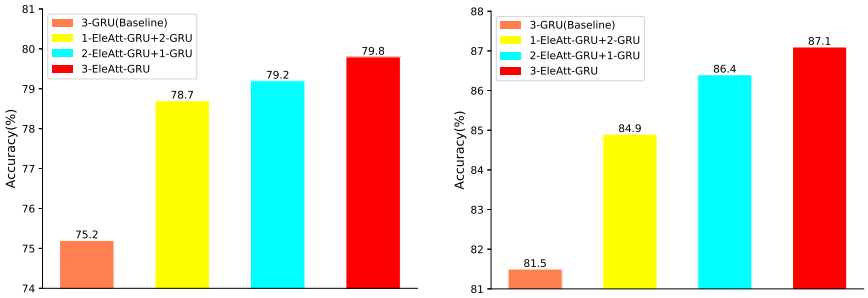
SYSU Human-Object Interaction dataset (SYSU) [16]. SYSU is a small RGB+D+Skeleton dataset, including 480 sequences performed by 40 different subjects. It contains 12 actions. A subject has 20 joints with 3D coordinates. We follow the standard protocols proposed in [16] for evaluation. They include two settings. For the Cross Subject (CS) setting, half of the subjects are used for training and the others for testing. For the Same Subject (SS) setting, half of the sequences of each subject are used for training and others for testing. The average performance of 30-fold cross validation is reported.

JHMDB dataset (JHMDB) [18]. JHMDB is an RGB-based dataset which has 928 RGB videos with each video containing about 15-40 frames. It contains 21 actions performed by different actors. This dataset is challenging where the videos are collected on the Internet which also includes outdoor activities.

5.2 Implementation Details

We perform our experiments on the deep learning platform of Keras [4] with Theano [1] as the backend. For the RNN networks, Dropout [34] with the probability of 0.5 is used to alleviate overfitting. Gradient clipping similar to [35] is used by constraining the maximum amplitude of the gradient to 1. Adam [21] is used to train the networks from end-to-end. The initial learning rate is set to 0.005 for 3D skeleton-based action recognition and 0.001 for RGB-based action recognition. During training, the learning rate will be reduced by a factor of 10 when the training accuracy does not increase. We use cross-entropy as the loss function.

For 3D skeleton-based action recognition, similar to the classification network design in [51], we build our systems by stacking three RNN layers with EleAttGs and one fully connected (FC) layer for classification. We use 100 RNN neurons in each layer. Considering the large difference on the sizes of the datasets, we set the batch size for the NTU, N-UCLA, and SYSU datasets to 256, 32, and 32, respectively. We use the sequence-level pre-processing method in [51] by setting the body center in the first frame as the coordinate origin to make the system invariant to the initial position of human body. To improve the robustness to view variations at the sequence level, we can perform data augmentation by randomly rotating the skeleton around the X, Y and Z axes by various degrees



(a) NTU-CS (b) NTU-CV

Fig. 2: Effectiveness of proposed EleAttGs on the three layered GRU network for 3D skeleton based human action recognition on the NTU dataset. "m-EleAtt-GRU+n-GRU" denotes that the first m layers are EleAtt-GRU layers and the remaining n layers are the original GRU layers.

ranging from -35 to 35 during training. For the N-UCLA and SYSU datasets, we use the RNN models pre-trained on the NTU dataset to initialize the baseline schemes and the proposed schemes.

For RGB-based action recognition, we feed an RNN network with the features to further explore temporal dynamics. Since our purpose is to evaluate whether the proposed EleAttG can generally improve recognition accuracy, we extract CNN features using some available pre-trained models without finetuning for the specific dataset or task. For the JHMDB dataset, we use the TSN model from [44,48] which was trained on the HMDB dataset [22] to extract a 1024 dimensional feature for each frame. For the NTU dataset which has more videos, we take the ResNet50 model [14,5] which has been pre-trained on ImageNet as our feature extractor (2048 dimensional feature for each frame) considering the ResNet50 model is much faster than the TSN model. The implementation details of the RNN networks are similar to that discussed above. For the NTU dataset, we stack three EleAtt-GRU layers, with each layer consisting of 512 GRU neurons. For the JHMDB dataset, we use only one GRU layer (512 GRU neurons) with EleAttG to avoid overfitting, considering that the number of video samples is much smaller than that of the NTU dataset. The batch size is set to 256 for the NTU dataset and 32 for the JHMDB dataset.

5.3 Effectiveness of Element-wise-Attention-Gates

Effectiveness on GRU network. Fig. 2 shows the effectiveness of EleAttG on the GRU network. Our final scheme with three EleAtt-GRU layers ("3-EleAtt-GRU") outperforms the baseline scheme "3-GRU(Baseline)" significantly, by **4.6%** and **5.6%** for the CS and CV settings, respectively. The performance increases as more GRU layers are replaced by the EleAtt-GRU layers.

Generalization to other input signals. The proposed RNN block with EleAttG is generic and can be applied to different types of source data. To demon-

Table 1: Effectiveness of proposed EleAttGs in the GRU network for RGB-based action recognition on the NTU and JHMDB datasets.

Dataset	NTU		JHMDB			
	CS	CV	Split1	Split2	Split3	Average
Baseline-GRU	61.3	66.8	60.6	59.2	62.9	60.9
EleAtt-GRU	63.3	70.6	64.5	59.2	65.0	62.9

Table 2: Effectiveness of EleAttGs on three types of RNN structures on the NTU dataset. "EleAtt-X" denotes the scheme with EleAttGs based on the RNN structure of X.

RNN structure	Scheme	CS	CV
Standard RNN	Baseline(1-sRNN)	51.6	57.6
	EleAtt-sRNN	**61.6**	**67.2**
LSTM	Baseline(3-LSTM)	77.2	83.0
	EleAtt-LSTM	**78.4**	**85.0**
GRU	Baseline(3-GRU)	75.2	81.5
	EleAtt-GRU	**79.8**	**87.1**

strate this, we use CNN features extracted from RGB frames as the input of the RNNs for RGB based action recognition. Table 1 shows the performance comparisons on the NTU and JHMDB dataset respectively. The implementation details have been described in Section 5.2. We can see that the "EleAtt-GRU" outperform the "Baseline-GRU" by about 2-4% on the NTU dataset, and 2% on the JHMDB dataset. Note that the performance is not optimized since we have not used the fine-tuned CNN model on this dataset for this task.

Generalization on various RNN structures. The proposed EleAttG is generic and can be applied to various RNN structures. We evaluate the effects of EleAttGs on three representative RNN structures, *i.e.,* the standard RNN (sRNN), LSTM, GRU respectively and show the results in Table 2. Compared with LSTM and GRU, the standard RNN neurons do not have the gating designs which control the contributions of the current input to the network. The EleAttG can element-wise control the contribution of the current input, which remedies the lack of gate designs to some extent. The gate designs in LSTM and GRU can only control the information flow input-wise. In contrast, the proposed EleAttGs are capable of controlling the input element-wise, adding the attentiveness capability to RNNs. We can see that the adding of EleAttGs enhances performance significantly. Note that for sRNN, we build both the Baseline(1-sRNN) and our scheme using one sRNN layer rather than three as those for LSTM and GRU, in considering that the three layered sRNN baseline converges to a poorer performance, i.e., 33.6% and 42.8% for the CS and CV settings, which may be caused by the gradient vanishing of sRNN.

Table 3: Performance comparisons on the NTU dataset in accuracy (%).

Method	CS	CV
Skeleton Quads [10]	38.6	41.4
Lie Group [38]	50.1	52.8
Dynamic Skeletons [16]	60.2	65.2
HBRNN-L [9]	59.1	64.0
Part-aware LSTM [30]	62.9	70.3
ST-LSTM + Trust Gate [26]	69.2	77.7
STA-LSTM [33]	73.4	81.2
GCA-LSTM [27]	74.4	82.8
URNN-2L-T [24]	74.6	83.2
Clips+CNN+MTLN [20]	79.6	84.8
VA-LSTM [51]	79.4	87.2
Baseline-GRU	75.2	81.5
EleAtt-GRU	79.8	87.1
EleAtt-GRU(aug.)	**80.7**	**88.4**

Comparisons with state-of-the-arts on skeleton based action recognition. Table 3, 4 and 5 show the performance comparisons with state-of-the-art approaches for the NTU, N-UCLA and SYSU datasets, respectively. "Baseline-GRU" denotes our baseline scheme which is built by stacking three GRU layers while "EleAtt-GRU" denotes our proposed scheme which replaces the GRU layers by the proposed GRU layers with EleAttGs. Implementation details can be found in Section 5.2. "EleAtt-GRU(aug.)" denotes that data argumentation by rotating skeleton sequences is performed during training. We achieve the best performances in comparison with other state-of-the-art approaches on all the three datasets. Our scheme "EleAtt-GRU" achieves significant gains over the baseline scheme "Baseline-GRU", of 4.6-5.6%, 4.7%, and 2.4-2.8% on the NTU, N-UCLA, and SYSU datasets, respectively.

Visualization of the responses of EleAttG. To better understand the learned element-wise attention, we observe the responses of the EleAttG in the first GRU layer for the skeleton based action recognition. In the first layer, the input (with dimension of $3 \times J$) at a time slot corresponds to the J joints with each joint represented by the X, Y, and Z coordinate values. The physical meaning of the attention responses is clear. However, in a higher layer, the EleAttG modulates the input features on each element which is more difficult to interpret and visualize. Thus, we perform visualization based on the attention responses of the first GRU layer in Fig. 3 for the actions of *kicking* and *touching the neck*. Actually, the absolute response values cannot represent the relative importances across dimensions very well. The statistical energies of the different elements of the original input are different. For example, the foot joint which is in general far away from the body center has a higher average energy than that of the body center joint. We can imagine that there is a static modulation $\overline{a_i}$ on the i^{th} element of the input, which can be calculated by the average energy before

Table 4: Performance comparisons on the N-UCLA dataset in acc. (%).

Method	accuracy
HOJ3D [47]	54.5
AE [41]	76.0
VA-LSTM [51]	70.7
HBRNN-L [9]	78.5
Baseline-GRU	84.3
EleAtt-GRU	89.0
EleAtt-GRU(aug.)	**90.7**

Table 5: Performance comparisons on the SYSU dataset in acc. (%).

Method	CS	SS
LAFF [17]	54.2	-
DS [16]	75.5	76.9
ST-LSTM[26]	76.5	-
VA-LSTM [51]	76.9	77.5
Baseline-GRU	82.1	82.1
EleAtt-GRU	84.9	84.5
EleAtt-GRU(aug.)	**85.7**	**85.7**

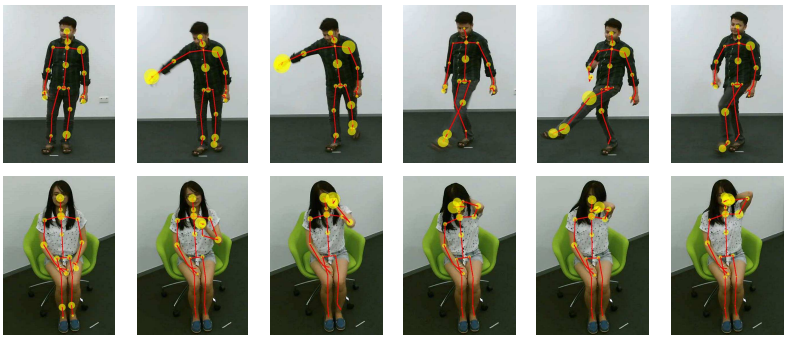
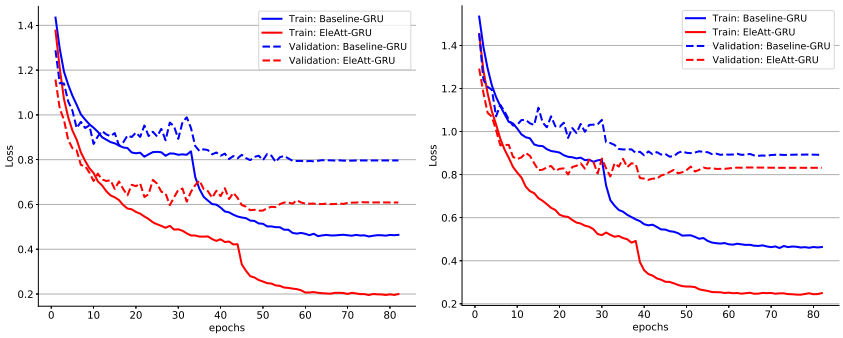


Fig. 3: Visualization based on the attention responses of the first GRU layer for the actions of *kicking* and *touching neck*. For each joint, the size of the yellow circle indicates the learned level of importance.

and after the modulation. For the i^{th} element of an sample j with attention value $a_{i,j}$, we use the relative response value $\widehat{a_{i,j}} = a_{i,j}/\overline{a_i}$ for visualization to better reflect the importances among joints. Note that the sum of the relative responses for the X, Y, and Z of a joint is utilized for visualization. For the action of *touching neck* which is highly concerned with the joints on the arms and heads, the relative attention on those joints are larger. For *kicking*, the relative attention on the legs is large. These are consistent with a human's perception.

5.4 Discussions

Convergence of Learning. Fig. 4 shows the loss curves for the training set and validation set during the training process for the proposed EleAtt-GRU and the baseline Baseline-GRU, respectively. By adding the EleAttGs, the convergence becomes faster and the final loss is much lower. EleAtt-GRU is consistently better than the baseline. The modulation of input can control the information



(a) NTU-CS (b) NTU-CV

Fig. 4: Loss curves during training on the NTU dataset for the proposed scheme "EleAtt-GRU" and the baseline scheme "Baseline-GRU".

flow of each input element adaptively and make the subsequent learning within the neurons much easier.

Relaxing the sum-to-1 constraint on EleAttG responses. Unlike other works [49,33,27], we do not use Softmax, which enforces the sum of attention responses to be 1, as the activation function of EleAttG. Instead, we use the Sigmoid activation function to avoid introducing mutual influence of elements. We show the experimental comparisons between the cases with the sum-to-1 constraint (*w/constraint*) by using Softmax, and our case without such constraint (*wo/constraint*) by using Sigmoid in Table 6. "EleAttG-n^{th}" denotes that the n^{th} GRU layer uses the GRU with EleAttG while the other layers still use the original GRU. "Baseline" denotes the baseline scheme with three GRU layers. We can see *wo/constraint* always performs better than that with constraint *w/constraint*. Adding EleAttG with constraint on the second or the third layer even decreases the accuracy by about 2.4-3.2% in comparison with the baselines.

Number of parameters versus performance. For an RNN block, the adding of an EleAttG increases the number of parameters. Taking a GRU block of N neurons with the input dimension of D as an example, the numbers of parameters for the original GRU block and the proposed EleAttG-GRU block are $3N(D+N+1)$, and $3N(D+N+1)+D(D+N+1)$, respectively. We calculate the computational complexity by counting the number of floating-point operations (FLOPs) including all multiplication and addition operations. At a time slot, adding attention to the layer as in 4 and 5 takes $D(D+N+1)$ multiplication operations and $D(D+N)$ addition operations. Then the complexity increases from $N(6D{+}6N{+}5)$ to $N(6D{+}6N{+}5){+}D(2D{+}2N{+}1)$, which is approximately proportional to the number of parameters.

Table 7 shows the effect of the number of parameters under different experimental settings on the NTU dataset. Note that "m-GRU(n)" denotes the baseline scheme which is built by m GRU blocks (layers) with each layer composed of n neurons. "m-EleAtt-GRU(100)" denotes our scheme which includes

Table 6: Performance comparisons about relaxing the constraint to EleAttG on the NTU dataset in terms of accuracy (%).

Protocols	Method	Baseline	EleAttG-1^{st}	EleAttG-2^{nd}	EleAttG-3^{rd}
CS	w/ constraint	75.2	75.0	72.7	72.0
	wo/ constrain	75.2	**78.7**	**77.3**	**76.4**
CV	w/ constraint	81.5	83.7	79.1	78.8
	wo/ constrain	81.5	**84.9**	**83.5**	**82.5**

Table 7: Effect of the number of parameters on the NTU dataset.

Scheme	# parameters	CS	CV
2-GRU(100)	0.14M	75.5	81.4
2-GRU(128)	0.21M	75.8	81.7
3-GRU(100)	0.20M	75.2	81.5
3-GRU(128)	0.31M	76.5	81.3
2-EleAtt-GRU(100)	0.20M	78.6	85.5
3-EleAtt-GRU(100)	0.28M	79.8	87.1

m EleAtt-GRU layers with each layer composed of 100 neurons. We can see that the performance increases only a little when more neurons ("2-GRU(128)") or more layers ("3-GRU(100)") are used in comparison with the baseline "2-GRU(100)". In contrast, our scheme "2-EleAtt-GRU(100)", achieves significant gains of 3.1-4.1%. Similar observation can be found for three-layer case. With the similar number of parameters, adding EleAttG is much more effective than increasing the number of neurons or the number of layers.

6 Conclusions

In this paper, we propose to empower the neurons in recurrent neural networks to have the attentiveness capability by adding the proposed EleAttG. It can explore the varying importance of different elements of the inputs. The EleAttG is simple yet effective. Experiments show that our proposed EleAttG can be used in any RNN structures (*e.g* standard RNN, LSTM and GRU) and any layers of the multi-layer RNN networks. In addition, for both human skeleton-based and RGB-based action recognitions, EleAttG boosts performance significantly. We expect that, as a fundamental unit, the proposed EleAttG will be effective for improving many RNN-based learning tasks.

Acknowledgment

This work is supported by National Key Research and Development Program of China under Grant 2016YFB1001004, Natural Science Foundation of China under Grant 61773311 and Grant 61751308.

References

1. Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., et al.: Theano: A python framework for fast computation of mathematical expressions. arXiv preprint arXiv:1605.02688 **472**, 473 (2016)
2. Cho, K., van Merriënboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: EMNLP. pp. 1724–1734. Association for Computational Linguistics (Oct 2014)
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. arXiv preprint arXiv:1406.1078 (2014)
4. Chollet, F.: Keras. `https://github.com/fchollet/keras` (2015)
5. Chollet, F.: Resnet50 model. `https://github.com/fchollet/deep-learning-models/releases/download/v0.2/resnet50_weights_tf_dim_ordering_tf_kernels.h5`
6. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
7. Diba, A., Sharma, V., Van Gool, L.: Deep temporal linear encoding networks. In: CVPR. pp. 2329–2338 (2017)
8. 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: CVPR. pp. 2625–2634 (2015)
9. Du, Y., Wang, W., Wang, L.: Hierarchical recurrent neural network for skeleton based action recognition. In: Computer Vision and Pattern Recognition (CVPR). pp. 1110–1118 (2015)
10. Evangelidis, G., Singh, G., Horaud, R.: Skeletal quads: Human action recognition using joint quadruples. In: ICPR. pp. 4513–4518. IEEE (2014)
11. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with lstm (1999)
12. Gers, F.A., Schraudolph, N.N., Schmidhuber, J.: Learning precise timing with lstm recurrent networks. JMLR **3**(Aug), 115–143 (2002)
13. Han, F., Reily, B., Hoff, W., Zhang, H.: Space-time representation of people based on 3d skeletal data: A review. CVIU **158**, 85–105 (2017)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
16. Hu, J.F., Zheng, W.S., Lai, J., Zhang, J.: Jointly learning heterogeneous features for rgb-d activity recognition. In: CVPR. pp. 5344–5352. IEEE (2015)
17. Hu, J.F., Zheng, W.S., Ma, L., Wang, G., Lai, J.: Real-time rgb-d activity prediction by soft regression. In: ECCV. pp. 280–296. Springer (2016)
18. Jhuang, H., Gall, J., Zuffi, S., Schmid, C., Black, M.J.: Towards understanding action recognition. In: ICCV. pp. 3192–3199. IEEE (2013)
19. Jozefowicz, R., Zaremba, W., Sutskever, I.: An empirical exploration of recurrent network architectures. In: ICML. pp. 2342–2350 (2015)
20. Ke, Q., Bennamoun, M., An, S., Sohel, F., Boussaid, F.: A new representation of skeleton sequences for 3d action recognition. In: CVPR. pp. 4570–4579. IEEE (2017)

21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
22. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: Hmdb: a large video database for human motion recognition. In: ICCV. pp. 2556–2563. IEEE (2011)
23. Li, J., Wei, Y., Liang, X., Dong, J., Xu, T., Feng, J., Yan, S.: Attentive contexts for object detection. TMM **19**(5), 944–954 (2017)
24. Li, W., Wen, L., Chang, M.C., Lim, S.N., Lyu, S.: Adaptive rnn tree for large-scale human action recognition. In: In Computer Vision and Pattern Recognition (CVPR). pp. 1444–1452 (2017)
25. Lipton, Z.C., Berkowitz, J., Elkan, C.: A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019 (2015)
26. Liu, J., Shahroudy, A., Xu, D., Wang, G.: Spatio-temporal lstm with trust gates for 3d human action recognition. In: ECCV. pp. 816–833. Springer (2016)
27. Liu, J., Wang, G., Hu, P., Duan, L.Y., Kot, A.C.: Global context-aware attention lstm networks for 3d action recognition. In: CVPR (2017)
28. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
29. Olah, C.: Lstm. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/` (2015)
30. Shahroudy, A., Liu, J., Ng, T.T., Wang, G.: Ntu rgb+d: A large scale dataset for 3d human activity analysis. In: CVPR (June 2016)
31. Sharma, S., Kiros, R., Salakhutdinov, R.: Action recognition using visual attention. arXiv preprint arXiv:1511.04119 (2015)
32. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: NIPS. pp. 568–576 (2014)
33. Song, S., Lan, C., Xing, J., Zeng, W., Liu, J.: An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In: AAAI. vol. 1, p. 7 (2017)
34. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. JMLR **15**(1), 1929–1958 (2014)
35. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS. pp. 3104–3112 (2014)
36. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NIPS. pp. 6000–6010 (2017)
37. Veeriah, V., Zhuang, N., Qi, G.J.: Differential recurrent neural networks for action recognition. In: ICCV. pp. 4041–4049. IEEE (2015)
38. Vemulapalli, R., Arrate, F., Chellappa, R.: Human action recognition by representing 3d skeletons as points in a lie group. In: CVPR. pp. 588–595 (2014)
39. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: A neural image caption generator. In: CVPR. pp. 3156–3164. IEEE (2015)
40. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: ICCV. pp. 3551–3558 (2013)
41. Wang, J., Liu, Z., Wu, Y.: Learning actionlet ensemble for 3d human action recognition. In: Human Action Recognition with Depth Cameras, pp. 11–40. Springer (2014)
42. Wang, J., Liu, Z., Wu, Y., Yuan, J.: Mining actionlet ensemble for action recognition with depth cameras. In: CVPR. pp. 1290–1297. IEEE (2012)
43. Wang, J., Nie, X., Xia, Y., Wu, Y., Zhu, S.C.: Cross-view action modeling, learning and recognition. In: CVPR. pp. 2649–2656 (2014)

44. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: ECCV. pp. 20–36 (2016)
45. Wang, P., Yuan, C., Hu, W., Li, B., Zhang, Y.: Graph based skeleton motion representation and similarity measurement for action recognition. In: ECCV. pp. 370–385. Springer (2016)
46. Wang, Y., Wang, S., Tang, J., O'Hare, N., Chang, Y., Li, B.: Hierarchical attention network for action recognition in videos. arXiv preprint arXiv:1607.06416 (2016)
47. Xia, L., Chen, C.C., Aggarwal, J.: View invariant human action recognition using histograms of 3d joints. In: Computer Vision and Pattern Recognition Workshop (CVPRW). pp. 20–27. IEEE (2012)
48. Xiong, Y.: TSN model. `https://github.com/yjxiong/temporal-segment-networks` (2016)
49. 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: ICML. pp. 2048–2057 (2015)
50. Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: CVPR. pp. 4694–4702 (2015)
51. Zhang, P., Lan, C., Xing, J., Zeng, W., Xue, J., Zheng, N.: View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In: ICCV (2017)
52. Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L., Xie, X., et al.: Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. In: AAAI. vol. 2, p. 8 (2016)