Two-Stream RNN/CNN for Action Recognition in 3D Videos

Rui Zhao, Haider Ali, Patrick van der Smagt

Abstract—The recognition of actions from video sequences has many applications in health monitoring, assisted living, surveillance, and smart homes. Despite advances in sensing, in particular related to 3D video, the methodologies to process the data are still subject to research. We demonstrate superior results by a system which combines recurrent neural networks with convolutional neural networks in a voting approach. The gated-recurrent-unit-based neural networks are particularly well-suited to distinguish actions based on long-term information from optical tracking data; the 3D-CNNs focus more on detailed, recent information from video data. The resulting features are merged in an SVM which then classifies the movement. In this architecture, our method improves recognition rates of state-of-the-art methods by 14% on standard data sets.

I. INTRODUCTION

Recognition of human activity in 3D videos has received increasing attention since 2010 [1]–[7]. Compared to 2D videos, 3D videos provide more spatial information and could be more informative. Action recognition with 3D videos is applied in different fields, such as health monitoring for patients, assisted living for disabled people, and robot perception and cognition.

Following this line of research, this paper proposes and applies novel deep-learning methods on what is currently the largest 3D action recognition dataset. Our results are compared with existing best approaches and are shown to be superior. Our proposed deep-learning methods consist mainly of three parts: a novel skeleton-based recurrent neural network structure, using a 3D-convolutional [8] neural network for RGB videos, and sketching a new two-stream fusion method to combine RNN and CNN. All methods are evaluated on the NTU RGB+D Dataset [2]. The dataset was published in 2016 and contains more than 56k action samples in four different modalities: RGB videos, depth map sequences, 3D skeletal data, and infrared videos. The dataset consists of 60 different action classes including daily, health-related, and mutual actions. In this paper, we use both the 3D skeletal data and RGB videos.

Traditional studies on 3D action recognition use different kinds of methods [1], [9]–[52] to compute handcrafted features, while deep-learning approaches [2]–[7], [53], [54] are end-to-end trainable and can be applied directly on raw data. Focussing on the latter, for skeleton-based activity analysis, [2]–[5] used different kinds of recurrent neural networks to acquire state-of-the-art performances on various of 3D action datasets. Du et al. [3] propose an hierarchical RNN, which is fed with manually divided five groups of the human skeleton, such as two hands, two legs, and one torso. Inspired from this, Shahroudy et al. [2] present a novel long short-term memory (LSTM) [55] cell, called part-aware LSTM, which is also fed with separated five parts of skeleton. Evolved from these two ideas, Zhu et al. [5] provide a novel deep RNN structure, which can automatically learn the co-occurrence, similar to grouping data into five human body parts, from skeleton data. Most recently, Liu et al. [4] propose a skeleton tree traversal algorithm and a new gating mechanism to improve robustness against noise and occlusion.

However, our proposed RNN structure makes a different contribution. Our method is inspired by recent normalization technologies [56] and a novel recurrent neuron mechanism [57]. With these advanced deep learning technologies embedded into our RNN structure, it can be trained with 13 times fewer iterations and for each iteration consumes 20% less computational time, compared to a normal RNN model with LSTM cells. Our contribution focuses more on making the network much easier to train, less inclined to overfitting, and deep enough to represent the data. More importantly, our proposed RNN structure outperforms all other skeleton-based methods on the largest 3D action recognition dataset.

To process RGB videos, our method is inspired by different kinds of convolutional neural networks [8], [58]–[60]. We use a 3D-CNN [8] model on the RGB videos of the NTU RGB+D dataset. We compare the results with proposed RNN models, and fuse their output.

To combine the RNN and CNN models, we propose two fusion structures: decision and feature fusion. The first is very simple to use, whereas the second provides a better performance. For decision fusion, we illustrate a voting method based on the confidence of the classifiers. For feature fusion, we propose a novel two-stream RNN/CNN structure, shown in Fig. 4, which combines temporal and spatial features from the RNN and CNN models and boost the performance by a significant margin. Our two-stream RNN/CNN structure outperforms the current state-of-the-art method [4] more than 14% on both cross subject and cross view settings.

To summarize, our contributions in this paper are:

- A novel RNN structure is proposed, which converges 13 times faster during training and costs 20% less computational power at each forward pass, compared to a normal LSTM;
- Two fusion methods are proposed to combine the proposed RNN structure and a 3D-CNN structure [8]. The decision fusion is easier to use, while the feature fusion has superior performance.
II. METHODOLOGY

In this section, we first introduce the concept of recurrent neural networks and batch normalization, and then describe the proposed LSTM structure: a deep bidirectional gated recurrent neural network with batch normalization and dropout. Afterwards, the applied 3D-CNN model and two-stream RNN/CNN fusion architectures, i.e., decision fusion and feature fusion, are introduced.

A. Recurrent Neural Network

1) Vanilla Recurrent Neural Network: Recurrent neural networks can handle sequence information with varied lengths of time steps. This transforms the input X to a internal hidden state h_t at each time step. The network passes the state h_t along with the next input x_t+1 to the neuron, time step after time step. The neuron learns when to remember and forget information with nonlinear activation functions:

\[ h_t = \sigma(W x_t + b) \]
\[ y_t = \sigma(V h_t) \]

where \( t \in \{1, \ldots, T\} \) represents time steps, and \( \sigma \) represents a nonlinear activation function such as a standard logistic sigmoid function \( \sigma(x) = 1/(1 + e^{-x}) \) or a hyperbolic tangent function \( \tanh(x) \).

Multiple layers of RNN can be stacked to increase the complexity:

\[ h_t^{(l)} = \sigma(W_{l}^{(l)} h_{t-1}^{(l)}) \]
\[ y_t^{(l)} := x_t \]

where \( l \in \{1, \ldots, L\} \) denotes the layer number.

In practice, a vanilla RNN does not remember information over a longer time; a problem which is related to the vanishing gradient problem.

2) Long Short-Term Memory: This problem can be solved by LSTM [55] which stores information in gated cells at the neurons. This allows errors to be backpropagated through hundreds or thousands of time steps:

\[ i = \text{sigm}(W_i x_t + b_i) \]
\[ f = \text{sigm}(W_f h_{t-1} + b_f) \]
\[ o = \text{sigm}(W_o h_{t-1} + b_o) \]
\[ c_t = f \odot c_{t-1} + i \odot \hat{c}_t \]
\[ h_t = o \odot \tanh(c_t) \]

where \( i, f, o \) and \( \hat{c}_t \) denote input gate, forget gate, and output gate, respectively. \( \hat{c}_t \) represents new candidate values, which could be added to the cell state \( c_t \). We use \( \odot \) for element-wise multiplication.

3) Gated Recurrent Unit: An improvement to LSTM called gated recurrent unit (GRU) was proposed in [57]. GRU has a simpler structure and can be computed faster. The three gates from LSTM are combined into two gates, respectively updating gate \( z \) and resetting gate \( r \) in GRU. GRU also combines cell state \( c_t \) and hidden state \( h_t \) into one state \( h_t^i \). The mathematical description is as follows:

\[ \hat{h}_t = \tanh(W x_t + r \odot h_{t-1}) \]
\[ y_t = \tanh(W x_t + \hat{h}_t) \]

where \( \hat{h}_t \) denotes new candidate state values.

4) Bidirectional Recurrent Neural Network: A bidirectional RNN [61] performs a forward pass and a backward pass, which runs input data from \( t = T \) to \( t = 1 \) and from \( t = 1 \) to \( t = T \), respectively.

For classification, the output of an RNN \( y_t \) can be passed to a fully-connected layer with softmax activation functions; this allows us to interpret the output as a probability.

5) Batch Normalization: To train a deep neural network, the internal covariate shift [56] slows down the training process. The internal covariate shift is the distribution of each layer’s input changes during training, because the parameters in the previous layer are changing. To reduce the internal covariate shift, we could whiten the layer activations, but this takes too much computation power. Batch normalization, a part of the neural network structure, approximates this process by standardizing the activations \( x \) using a statistical estimate of the mean \( E(x) \) and standard deviation \( \text{Var}(x) \) for each training mini-batch. It can be shown that

\[ \text{BN}(x; \gamma, \beta) = \gamma \frac{x - \hat{E}(x)}{\sqrt{\text{Var}(x) + \epsilon}} + \beta \]

where \( \gamma \in \mathbb{R}^d \) and \( \beta \in \mathbb{R}^d \) are scale and shift parameters for the activation \( x \in \mathbb{R}^{n \times d} \). With these, identity transformation for each activation could be presented. \( \epsilon \in \mathbb{R} \) is a constant added as a regularization parameter for numerical stability. The division in Eq. (12) is performed element-wise. \( \gamma \) and \( \beta \) are learned during training and fixed during inference.

6) Proposed RNN Structure: For skeleton-based action recognition tasks, the data set consists of the 3D coordinates of a number of body joints. We feed this information, together with action labels, to an RNN. This RNN network has two bidirectional layers, each of which consists of 300 GRU cells. After the recurrent layers follows the batch normalization layer, which standardizes the activations from the RNN layer. Then the normalized activations flow to the next fully-connected layer with 600 rectified linear unit (ReLU) [62] activation functions. During training, in each iteration the network randomly drops out 25% of the neurons between the batch normalization layer and the next fully-connected layer to reduce overfitting. Lastly, a softmax layer
maps the compressed motion information (features) to 60 action classes. Fig. 1 shows the structure of this RNN network.

To highlight the improvements of this final proposed model, we compare our approach to simpler models. These models are a standard RNN; an LSTM-RNN; LSTM plus batch normalization (“LSTM-BN”), LSTM-BN with dropout (“LSTM-BN-DP”), GRU-BN-DP, and a bidirectional GRU-BN-DP which we call “BI-GRU-BN-DP”. All these models have one recurrent layer. The next complexity is adding gated recurrent layers with batch normalization, dropout, one hidden fully-connected layer, and one output softmax layer. For clarity, the temporal recurrent structure of GRU cells is not shown here.

B. Convolutional Neural Network

To process RGB videos, we choose to use the 3D-CNN model from [8], as it shows promising performances on 2D video action recognition tasks. We believe that 3D convolution nets are more suitable for learning features from 2D video action recognition tasks. We believe that 3D convolution nets are more suitable for learning features from videos than 2D convolution nets.

2D convolution generates a series of 2D feature maps from images. Inspired by this, a 3D convolution processes frame clips, where the third dimension is time step, which results in a series of 3D feature volumes, as shown in Fig. 2. This compressed representation contains spatiotemporal information from the video clips. To learn a rich amount of features, multiple layers of convolution and max-pooling operations are stacked into one model.

To be specific, the 3D-CNN model [8], which we choose, has five convolutional groups, each group has one or two convolutional layers and one max-pooling layer, two fully-connected layers, and one softmax output layer. The details of this model are presented in Fig. 3.

We finetune this model with pretrained parameters [8] on Sports-1M Dataset, which has approximately one million YouTube videos. This reduces overfitting and demands less training time on the current dataset.

C. Two-stream RNN/CNN

As having the proposed RNN structure for the skeleton data and the 3D-CNN model for the RGB videos, we want to combine the strengths of RNN and CNN nets. To improve the performance, we propose two fusion models, decision fusion and feature fusion.

1) Decision Fusion: In the case of decision fusion, we use a simple but efficient voting method, inspired by majority voting. As a result of having only two classifiers, we cannot apply majority voting. Instead, the fusion method predicts based on voting confidence.

We first split the dataset into training, validation and testing. The same training set is used to train the RNN and CNN nets. The validation set is then used to find the best parameters, trust weights \( w_r \) and \( w_c \) for both RNN and CNN classifiers. Afterwards, we compare the confidences, which are the highest probabilities of softmax output from both classifiers for each prediction. The more confident one wins:

\[
y(x_i) = \begin{cases} 
  y_r(x_i), & \text{if } w_r \times y_r(x_i) > w_c \times y_c(x_i) \\
  y_c(x_i), & \text{otherwise}
\end{cases}
\]

where \( y(x_i) \) is the fused prediction for sample \( x_i \); \( y_r \) and \( y_c \) denote RNN and CNN prediction, respectively.

Based on this concept, we develop a way to fuse the predictions from RNN and CNN. We evaluate the performance with the validation dataset and search for the best trust weights for decision fusion. Having only two parameters \( w_r \) and \( w_c \), only little tuning is needed.

2) Feature Fusion: Another way to combine these two neural networks is feature fusion.

We first train the RNN and CNN models on the training dataset. As in training, neural nets can learn discriminant information from raw data. Thus, we use the trained RNN model to extract temporal features from 3D skeleton data.
This dataset has two standard evaluation criteria [2]. The first one is a cross-subject test, in which half of the subjects are used for training and the other half are used for testing. The second one is a cross-view test, in which two viewpoints are used for training and one is excluded for evaluation.

B. Implementation details

In our experiments, the implementation consists of RNN, CNN, and Fusion. For all these models we use the same training, validation and testing splits. The validation set is composed of 10% of the subjects in the training set in [2]. The remaining subjects in the training set [2] make up the training set.

1) RNN Implementation: In the RNN experiments, we have two human skeletons as input, each skeleton has 25 3D coordinates. Since the longest time step is 300, we pad all the action sequences to a length of 300. The dimension of each action sample is 300 (time steps) × 150 (coordinates).

We use TensorFlow [63] with TFlearn [64] and run the experiments on either one NVIDIA GTX 1080 GPU or one NVIDIA GTX TITAN X GPU. We train the network using RMSprop [65] optimizer and set learning rate as 0.001, decay as 0.9, and momentum as 0. We train the network from scratch using mini-batches of 1000 sequences for one-layer models and use mini-batches of 650 sequences for two-layer models. For all RNN nets, we use 300 neurons for each single-directional layer, double the amount of neurons for bidirectional layers, and we use a 75% keep probability for dropout. For batch normalization, we initialize $\gamma$ as 1.0, $\beta$ as 0.0, and set $\epsilon$ as $1 \times 10^{-5}$. The estimated means and variances are fixed during inference.

As a comparison, the mentioned parameters are the same for all proposed RNN models, only the structure changes.

2) CNN Implementation: We use the 3D-CNN model [8] in Caffe [66] and train it on RGB frames from the NTU RGB+D dataset, with pretrained parameters [8] from the Sport1M dataset. From RGB videos, we extract the frames, crop and resize them from $1920 \times 1080$ pixels to $320 \times 240$ pixels [8]. Videos are split into non-overlapped 16-frame clips.

We refer to the input of CNN model as a size of $c \times t \times h \times w$, where $c$ is the number of channels, $t$ is the number of time steps, $h$ and $w$ are the height and width of the frame, respectively. The network takes video clips as input and predicts the 60 action labels which belong to the 60 different actions. It further resizes the input frames to $128 \times 171$ pixel resolution. The input dimensions are $3 \times 16 \times 128 \times 171$ pixel. During training we use jittering on the input clips by random cropping them into $3 \times 16 \times 112 \times 112$ pixel. We fine-tune the network with stochastic gradient descent optimizer using mini-batches of 44 clips, with initial learning rate of 0.0001. The learning rate is then reduced by half, when no training progress was observed [65]. The training stopped after around 20 epochs.

For video-based prediction, the model averages the predictions over all 16-frame clips split from the same video and provides the final prediction for the input video. A similar

![Diagram](image-url)
idea is applied for extracting features from fc-6 layer, which averages the 4096-dimensional feature vectors over all clips in the same video, resulting in one 4096-dimensional vector for each video.

3) Fusion Implementation: We fuse the best RNN structure, 2 layer BI-GRU-BN-DP-H, with the 3D-CNN model, first using decision fusion, then using feature fusion.

For decision fusion, we first extract the softmax output, then search for the fusion parameters, trust weight $w_r$ and $w_c$ for RNN and CNN from the validation split. The parameters are $w_r = 1.00$ and $w_c = 2.88$ for the cross subject setup, and $w_r = 1.00$ and $w_c = 3.02$ for the cross view setup.

For feature fusion, we extract the RNN features (600 dimensions) from the fully-connected layer, and extract CNN features (4096 dimensions) from the fc-6 layer [8]. We then concatenate them into one feature array (4,696 dimensions) and apply L2 normalization. In the end, we have normalized RNN/CNN features from training, validation, and testing splits. We use training and validation splits to find the optimal value of $C$ for linear SVM [67] model. For both cross-subject and cross-view setups, we find that $C = 8.0$ gives the best validation accuracy.

Among all the models in this paper, feature fusion model shows the best testing results. We refer to this model as a two-stream RNN/CNN structure as shown in Fig. 4.

C. Experimental Results and Analysis

The evaluation results are shown in Tab. I. The first 16 rows are skeleton-based methods. The 3D-CNN model (17th row) uses RGB videos as input. The decision fusion (18th row) and feature fusion (19th row) models use the best RNN structure, which is the 2 Layer BI-GRU-BN-DP-H (16th row), and the 3D-CNN (17th row) model.

Tab. I shows that our RNN structure, the 1 Layer LSTM-BN, already outperforms the baseline method part-aware LSTM reported in [2] because batch normalization improves the LSTM model. Adding a dropout procedure reduces overfitting and further improves the results (rows 11, 12). From rows 12 and 13 we can see that the performances of LSTM and GRU cells are similar [68]. GRU is better in the cross-subject test and LSTM is better in the cross-view test. On the other hand, GRU is faster than LSTM both in computational speed and converge rate. As presented in Fig. 5 left, for 1k training steps, the same model performs 5.42% more accurately and takes 20% less computational time when using GRU cells than when using LSTM cells.

The addition of the extra fully-connected layer brings another significant improvement. This increases the complexity of the neural network, which helps the model capture more inherent features from the 3D skeleton data [69]. The recurrent layers before the fully-connected layer can be seen as a temporal feature extractor, which compact input information (dimension $300 \times 150$) into 600 dimensions. The latter part of the RNN structure can be considered as a classifier learning to map these 600-dimensional features to 60 different action categories. Altogether, our novel RNN model, 2 Layer BI-GRU-BN-DP-H, outperforms all the other skeleton-based models including ST-LSTM (Tree traversal) + Trust Gate [4].

Then, we use the RGB video data to train the 3D-CNN model. We use the voting method based on confidence to fuse the 2 Layer BI-GRU-BN-DP-H and 3D-CNN model. In the next step, we utilize a linear SVM [8] to fuse the fc-6 features from the CNN and the fc features from the RNN. This further improves results by over 13% in comparison to our best RNN, and by more than 14% compared to literature models [2], [4]. This boosting is due to the features from RNN and CNN model being highly complementary. The RNN model uses 50 3D coordinates for two human bodies over 300 time steps, and learns to find the long-term motion pattern. Whereas the CNN model has 2D RGB frames, which additionally has spatiotemporal information about objects, such as cups, pens, and books. However, the CNN model can

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Method</th>
<th>cross subject</th>
<th>cross view</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Skeleton Quads [2], [9]</td>
<td>38.62%</td>
<td>41.36%</td>
</tr>
<tr>
<td>02</td>
<td>Lie Group [2], [10]</td>
<td>50.08%</td>
<td>52.76%</td>
</tr>
<tr>
<td>03</td>
<td>FTP Dynamic Skeletons [2], [11]</td>
<td>60.23%</td>
<td>65.22%</td>
</tr>
<tr>
<td>04</td>
<td>HBRNN-L [2], [3]</td>
<td>59.07%</td>
<td>63.97%</td>
</tr>
<tr>
<td>05</td>
<td>Deep RNN [2]</td>
<td>56.29%</td>
<td>64.09%</td>
</tr>
<tr>
<td>06</td>
<td>Deep LSTM [2]</td>
<td>60.69%</td>
<td>67.29%</td>
</tr>
<tr>
<td>07</td>
<td>Part-aware LSTM [2]</td>
<td>62.93%</td>
<td>70.27%</td>
</tr>
<tr>
<td>08</td>
<td>ST-LSTM (Tree) + Trust Gate [4]</td>
<td>69.2%</td>
<td>77.7%</td>
</tr>
<tr>
<td>09</td>
<td>1 Layer RNN</td>
<td>18.74%</td>
<td>20.27%</td>
</tr>
<tr>
<td>10</td>
<td>1 Layer LSTM</td>
<td>60.99%</td>
<td>64.68%</td>
</tr>
<tr>
<td>11</td>
<td>1 Layer LSTM-BN</td>
<td>64.07%</td>
<td>71.86%</td>
</tr>
<tr>
<td>12</td>
<td>1 Layer LSTM-BN-DP</td>
<td>64.69%</td>
<td>73.48%</td>
</tr>
<tr>
<td>13</td>
<td>1 Layer GRU-BN-DP</td>
<td>65.21%</td>
<td>70.36%</td>
</tr>
<tr>
<td>14</td>
<td>1 Layer BI-GRU-BN-DP</td>
<td>64.78%</td>
<td>73.12%</td>
</tr>
<tr>
<td>15</td>
<td>2 Layer BI-GRU-BN-DP</td>
<td>66.21%</td>
<td>72.46%</td>
</tr>
<tr>
<td>16</td>
<td>2 Layer BI-GRU-BN-DP-H</td>
<td>70.70%</td>
<td>80.23%</td>
</tr>
<tr>
<td>17</td>
<td>3D-CNN [8]</td>
<td>79.75%</td>
<td>83.95%</td>
</tr>
<tr>
<td>18</td>
<td>Decision Fusion</td>
<td>82.05%</td>
<td>86.68%</td>
</tr>
<tr>
<td>19</td>
<td>Feature Fusion</td>
<td>83.74%</td>
<td>93.65%</td>
</tr>
</tbody>
</table>
only memorize information for 16 time steps long—longer memorization is prohibited by GPU memory limitations. These facts make the features from RNN and CNN model highly complementary, as the testing results show in row 16, 17, and 19 in Tab. 1.

D. Discussion

To better analyze and improve the performance of the model, we take a closer look at actions that are highly confusing to the two-stream RNN/CNN structure. As presented in Fig. 6 and 7, such action pairs include reading vs. writing, putting on a shoe vs. taking off a shoe, and rubbing two hands vs. clapping. These actions are shown in a video at https://www.youtube.com/watch?v=G0PXKCEgIoA. Fig. 8 shows some classified action samples.

There could be several reasons for this observation. First, these actions are sometimes inherent confusing. Secondly, there are flaws in the data. Kinect depth information, from which the NTU skeleton data is created, is quite noisy [70], [71]. Correspondingly, the 3D skeleton data used in our RNNs are also quite noisy [4]. RGB videos data are more accurate and stable, but single frames carry no 3D information. Thirdly, the 3D-CNN model [8] is trained with small video clips, which are 16 time steps long. The CNN model is adapted to find only short-term temporal features in these clips. As GPU memory and computing power increase, the model could also be adapted to find long-term temporal features in each whole video. Lastly, although, the RNN model can memorize the whole action sequences and give final predictions, it has no information about the appearances and movements of surrounding objects, which could be discriminative information for the classification task.

IV. CONCLUSION AND FUTURE WORK

In this paper, we propose a novel RNN structure for 3D skeletons that achieves state-of-the-art performance on the largest 3D action recognition dataset. The proposed RNN model can also be trained 13 times faster and saves 20% computational power on each training step. Additionally, the RGB videos from the same dataset are used to finetune a 3D-CNN model. In the end, an efficient fusion structure, two-stream RNN/CNN, is introduced to fuse the capabilities of both RNN and CNN models. The results of this method are 13% higher than using the proposed RNN alone, and 14% higher than the best published result in the literature. In the future, we want to consider using the other sensor modalities such as depth maps and IR sequences and see what is the best architecture to fuse all these modalities.

REFERENCES

Fig. 8: Some correctly classified action samples (first four frames with green predictions) and some mis-classified action samples (last two frames with red predictions). These samples are randomly picked from the feature fusion model cross view testing results.


