Modeling temporal dynamics and spatial configurations of actions using recurrent neural networks and self-attention gates

Mohammadmahdi Ghahramani
muhammadmahdi0007@gmail.com
https://github.com/MohammadMahdiGhahramani7/Computer_Vision_Course_Project

Abstract

Recently, skeleton-based action recognition gained more popularity due to cost-effective depth sensors coupled with real-time skeleton estimation algorithms. Due to the improved accuracy of depth detection sensors and cameras, it is possible to detect three-dimensional coordinates of body joints at high frame rates. These coordinates are input into the activity detection system. Using deep neural networks to solve machine learning problems has become very common with the advent of powerful hardware today. By utilizing the attention mechanism, a method that has significantly improved natural language processing in the past two years, we designed a structure to prevent unnecessary and junk data from entering both temporally and spatially. By dividing the body into five parts and recombining them together, this organizational structure shifts the focus of the model to the part of the body that performs the majority of the activity. In this research, I develop a model and examine its performance on the NTU RGB+D dataset. The original version of this dataset categorizes different human activities into 60 classes, each containing approximately 9000 data samples. However, this research considers 20 labels of the original dataset, each containing 200 data samples. This model provides an accuracy of 91.6% and 80.5%, respectively, for the train and test sets. Once the model was trained, this research attempts to bring the model into the federated learning space to take advantage of multiple clients whose data are privately considered to train the model.

1. Introduction

Recognition of human actions [1] has become one of the most active areas of computer vision research and there are many important research problems, including event recognition [14], group-based activities recognition [18], human-object interactions [9], and activities in egocentric videos [20, 7]. Several approaches have been proposed for recognizing actions in RGB videos captured by two-dimensional cameras. Nonetheless, it remains a challenging problem for three reasons. The first problem is that high-dimensional and low-quality input data are difficult to well extract. A second problem is that RGB video can be very sensitive to factors like illumination changes, occlusion, and clutter in the background. Finally, identifying actions requires high-level visual clues such as human poses and objects, which are very difficult to obtain directly from RGB videos. Using a few spots describing the motions of skeleton joints, humans can recognize actions [15], and experiments show that a large set of actions can be recognized from skeletons alone [16]. Unlike RGB-based action recognition, skeleton-based action recognition can avoid the laborious task of extracting features from videos and explicitly model the dynamics of action.

In this research, the NTU RGB+D is used in order to evaluate the model performance. There are two versions of this dataset, the first one with 120 labels and the second one with 60 labels. The second dataset has about 54000 data samples, i.e. about 9000 data samples per label. Since analyzing all this data is computationally expensive and this research does not want to restrict the result to hardware limitations, I selected 20 labels randomly, each containing 200 data samples. Using the hold-out approach with the aim of model fine-tuning, I split the data into three subsets, training, validation and test set, respectively containing 70%, 20% and 10% of the whole data. Once the model is fine-tuned, the model is trained on the combination of training and validation sets and the performance is examined on the unseen test set.

In spite of the fact that too many studies use pre-trained models, this study constructs a deep attention-based model from scratch to classify human activities. Observe that pre-trained models primarily use Convolutional neural networks (CNNs) to extract features and classify images, not a sequence of three-dimensional joint coordinates. Using Long-short term memory (LSTM) units, I first split the joints of the body into five major parts: the body, the right hand, the left hand, the right leg, and the left leg, and then I reconstructed the full body using these inputs. By passing these

inputs from the LSTM layers and the self-attention layers, the newly constructed full-body contains the information about the joints most prominently involved in a given activity. Therefore, a linear layer combined with a softmax activation function can be used to classify activity.

The study also provides a visual representation of the results. We need to check how different wrong-labeled data are from their original labels to ensure that the model is working fairly well. It means that the model has not yet trained properly if it labels data whose real label is `brush-hair` as `jumping`. Alternatively, if the model misclassifies this data as `brush-teeth`, this error is meaningful since these two tasks are much more similar than `jumping`. This dataset, however, does not provide us with well-formed data that is easy to visualize. This study provides a gif-based visualization of the results by manipulating the input and applying several algebraic transformations.

The final step in this work is to bring the model structure and data into a federated learning environment. We use this technique to train an algorithm across multiple decentralized edge devices or servers holding local data samples.

The experiments performed in this study vary from visualization to model manipulation and federated learning model adjustments. By doing them, we obtain a model which is capable of classifying the data in a rational manner, i.e. the predicted label is not much different than the activity performed by the real label. These experiments will be explained in their corresponding sections later.

The main message that this study tries to send is how it could be easier to classify human activities by utilizing body joints' coordinator instead of RGB-colored image frames. LSTM and self-attention layers are able to extract the needed information and linear layers along with softmax unit are capable of classifying this extracted information. The code associated with visualization, model construction and training as well as federated-based model construction is available in my github of https://github.com/MohammadMahdiGhahramani7/Computer_Vision_Course_Project.

2. Related work

In this section, we briefly review action recognition approaches related to ours. The two aspects are as follows.

2.1. Action recognition with deep networks

Deep neural networks have made great progress in the area of action recognition. Different architectures are investigated to exploit local spatio-temporal information using 3D Convolutional neural networks (CNNs) [13, 17]. To capture complementary information between appearance and motion, a two-stream CNN architecture is developed for RGB based action recognition [25].

Recently, Recurrent neural networks (RNN) have been widely used for action recognition. Srivastava et al. [26] use multilayer Long Short Term Memory (LSTM) networks to learn representations of video sequences. Donahue et al. [6] develop an end-to-end trainable Long-term recurrent convolutional networks (LRCN) architecture which can simultaneously learn temporal dynamics and convolutional perceptual representations from RGB videos. Deep convolutional and RNNs have also been proposed and applied for activity recognition [12, 22].

2.2. Features based on skeletons

Previous skeleton based action recognition methods mainly focus on handcrafted features [2]. To get representations of postures, one straightforward feature is the pairwise joint location difference, which can be simply concatenated [21], or casted into 3D cone bins to build a histogram of 3D joints locations [28] for action recognition. Joint orientation is another good feature as it is invariant to the human body size. For example, Sempena et al. [24] apply dynamic time warping based on the feature vector built from joint orientation along time series. Bloom et al. [3] use AdaBoost to combine five types of features, i.e., pairwise joint position difference, joint velocity, velocity magnitude, joint angle velocity and 3D joint angle to recognize gaming actions, for real-time action recognition. There are some work that groups the joints of skeletons to construct planes from joints and then measures the joint-to-plane distance and motion. Yun et al. [30] capture the geometric relationship between the joint and the plane spanned by three joints. Sung et al. [27] compute the joint's rotation matrix with respect to the person's torso, hand position with the respect to the torso and joint rotation motion as features.

3. Dataset and visualization

In this part, we review the data and the way that they are selected from the original dataset. Moreover, we talk about an approach to get this data visualized.

3.1. Data selection

Previously mentioned, due to computational costs, we select a subset of categories. Table 1 gives a detailed description on available labels in the original dataset and also those labels that are selected for this study.

3.2. Data visualization

The dataset provides the 3D coordinates of 25 joints of the human full body. The full body is divided into five main parts, including the body, left hand, right hand, left leg, and right leg, and contains respectively five, six, six, four, and four joints. So each data sample has the information of 25 joints. As the coordinates are not well-defined by the

Table 1: **Twenty randomly-selected labels of the dataset**

Number	Label	Number	Label	Number	Label	Number	Label	Number	Label	Number	Label
1	drink water	11	reading	21	take off a hat cap	31	point to something	41	**sneeze cough**	51	**kicking**
2	eat meal	12	**writing**	22	**cheer up**	32	**taking a selfie**	42	staggering	52	pushing
3	**brush teeth**	13	**tear up paper**	23	hand waving	33	check time	43	falling down	53	pat on back
4	brush hair	14	put on jacket	24	**kicking something**	34	rub two hands	44	headache	54	point finger
5	**drop**	15	take off jacket	25	reach into pocket	35	**nod head bow**	45	chest pain	55	**hugging**
6	pick up	16	put on a shoe	26	hopping	36	shake head	46	**back pain**	56	giving object
7	**throw**	17	**take off a shoe**	27	**jump up**	37	**wipe face**	47	neck pain	57	**touch pocket**
8	sit down	18	put on glasses	28	phone call	38	salute	48	**nausea vomiting**	58	shaking hands
9	stand up	19	**take off glasses**	29	play with phone tablet	39	put palms together	49	fan self	59	**walking towards**
10	clapping	20	put on a hat cap	30	type on a keyboard	40	cross hands in front	50	punch slap	60	walking apart

dataset, we must scale them and perform some algebraic transformations before any visualization can be done.

The first step to prepare data for visualization is to extract coordinates from the dataset. To follow this procedure, let's suppose we select a random data sample from the left-hand part of the full body. Since it has six joints, each having 3D coordinates, we have an array of lengths of 18. This is just one single frame of this data. By breaking down this 18-element array list we will have access to the coordinates. Generally, the shape of the dataset is (X, Y, Z), X indication the whole number of data samples which is set at 4000 in this research, Y showing the number of frames per data sample which is set at 300 and Z illustrating the length of a specifying joint. If the joint belongs to the body the Z=5*3, if it belongs to the left hand or right hand the Z=6*3 and Z=4*3 for both legs. Figure1 shows the 25 joints of the body used in this dataset and the way that we categorize them into five main parts of the full body.

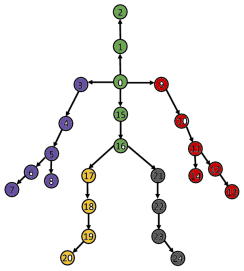
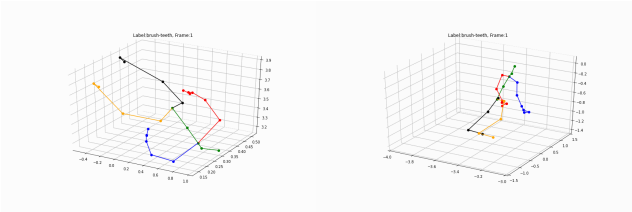


Figure 1: **The five main components of the human body.** The figure is a graphic representation of the dataset. It considers five main parts of the body and provides the 3D coordinates of them in separate array lists.

The second step is to rotate the joints in a way that they can get properly visualized. By defining the rotation matrix on Eq1 and multiplying it to the coordinates of joints we can rotate the full body in our desired direction in the three-dimensional space.

$$\mathcal{R}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}, \tag{1}$$

where θ is the angle to rotate in radians. This study, by trial and error, uses the $\theta=\frac{105\pi}{180}$ radians to rotate the full-body joints. Figure2 shows how the raw data is transformed and scaled in a proper way.



(a) Before (b) After

Figure 2: **3D visualization before and after scaling and applying algebraic transformations.**

4. Method

So far, we have seen that the input data is a time series. This data can be considered as a sentence. A sentence is made up of a number of words and means when the words are in the correct order. Now map this example to our input data. A number of detailed coordinates are stored over a period of time. The order of all this data is important in achieving the result. However, if we are careful, we will find that not all data is always stored this way, and our data collection system may be unreliable. These noises reduce the accuracy of the diagnosis. Also, when it comes to recognizing human activity, there may be movements that focus only on one part of the body, such as shaking hands, which is the most common change in the joints of the arm and hand, and the lower extremities do not change or have very little change.

Therefore, in this approach, I decided to use the attention mechanism and a unique hierarchical structure. The attention mechanism handles noise and distortion data, and the hierarchical structure makes the network focus entirely on the part where the changes are taking place. This structure Hierarchy is in fact responsible for paying attention to the spatial domain.

4.1. Attention mechanism, temporal space gate

The use of self-attention gates has played an important role in the development of activity detection. But so far,

most of the gates designed are made of neural networks themselves which have problems such as gradient vanishing or gradient explosion. Therefore, in this approach, we use an attention mechanism that uses a conversion system instead of neural networks. This method consists of two parts: encoder and decoder. An overview of this conversion can be seen in Figure3.

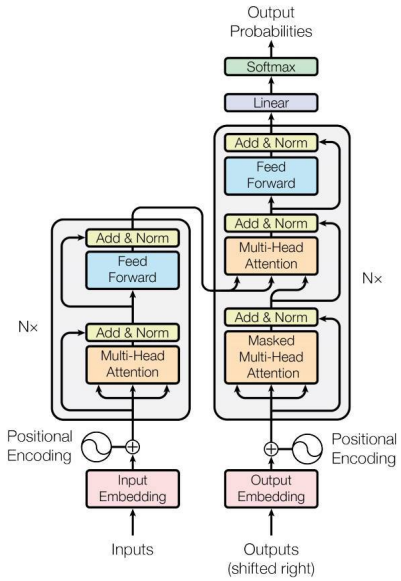


Figure 3: **Transformer structure.** This kind of structure is widely used in Natural language processing (NLP) in machine translation task.

An attention self component with `n` inputs will include `n` outputs. Simply put, this component allows inputs to interact with each other and decide which input to pay more attention to. In fact, the output is the attention score of each input. The temporal space gate in this method consists of a self-attention gate and an LSTM layer that we are already familiar with. In fact, the output of the attention self gate is considered as an input to LSTM layer.

One of the disadvantages associated with RNNs is gradient vanishing when backpropagating through time [10, 23]. It causes long-term dependencies are not detected. To improve the performance of RNNs to also detect long term dependencies, LSTM units are introduced [11]. However, LSTM is not the only way to address memory problem. Gated recurrent units (GRUs), the simpler version of LSTM units [5], address the same problem either. The idea behind either LSTM or GRU is to consider a memory cell that determines how much information should be memorized and transformed to the next units. LSTM with three gates named forget gate, update gate, and output gate, and GRU with one gate called update gate, handle this memory transformation.

Compared to RNNs, LSTM takes advantage of a memory cell $c^{<t>}$ to transform information. $c^{<t>}$ is calculated by weighting to the previous memory cell $c^{<t-1>}$ and a candidate cell $\tilde{c}^{<t>}$ which is related to the current LSTM unit. The update and forget gates handle the weighting task. They specify how much memory from previous units must be transformed and how much information from the current unit must be considered. Having $c^{<t>}$ calculated, the output gate is used to calculate the current hidden state $a^{<t>}$. Finally, $a^{<t>}$ is used to make a prediction. As LSTM units are consecutive, $a^{<t>}$ and $c^{<t>}$ are transformed to the next LSTM unit, and this process continues until the last unit. The following equations describe the order in which each of the above elements is calculated:

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c), \tag{2}$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u), \tag{3}$$

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f), \tag{4}$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o), \tag{5}$$

where Ws and bs are weight matrices which are updated through backpropagation step and biases respectively. Γ_u, Γ_f and Γ_o are update gate, forget gate and output gate respectively. σ is also the sigmoid activation function. Using these gates along with candidate cell $\tilde{c}^{<t>}$, Eq 6 and Eq 7 calculate $c^{<t>}$ and $a^{<t>}$ to be passed through the rest of LSTM units as following:

$$c^{<t>} = \Gamma_u \circ \tilde{c}^{<t>} + \Gamma_f \circ c^{<t-1>}, \tag{6}$$

$$a^{<t>} = \Gamma_o \circ \tanh(c^{<t>}). \tag{7}$$

In each unit, the corresponding value of the forget gate determines how much information we are better to forget. If it is close to 1, the previous memory cell contains valuable information from the past and should be transformed to the current memory cell. If it is close to 0, it means that the major part of the current memory cell is calculated by the candidate cell. The Γ_u, Γ_f and Γ_o are not constant for all LSTM units. They are vectors containing the sigmoid output for each LSTM unit. It is worth mentioning that gates, hidden states, memory cells, and candidate cells are all vectors, so Eq 6 and Eq 7 use element-wise products between these vectors. As the last step, if we wish to predict the current LSTM unit, Eq 8,

$$\hat{y}^{<t>} = g(W_y a^{<t>} + b_y), \tag{8}$$

is employed to return the output where `g` is an activation function. Fig 4 illustrates a schematic overview of a classic LSTM unit. There are other kinds of LSTM units such as `peephole connections` which apart from $a^{<t-1>}$ and $x^{<t>}$ considers $c^{<t-1>}$ to calculate gates values in Eq 3, Eq 4 and Eq 5 [8].

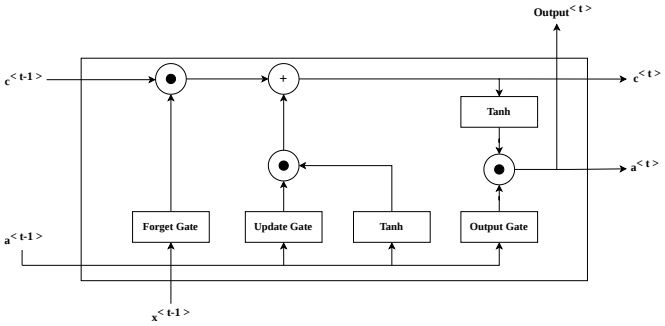


Figure 4: **LSTM unit architecture.**

4.2. Hierarchical structure

The skeleton data has a lot of flexibility. This allows them to be separated. In this study, the human skeleton is divided into five parts: the right hand, left hand, body, right foot, and left foot. The purpose of diversification is to increase the network's focus on parts in a specialized manner. As a result of this division, we have 5 sets of data as inputs to the network. Our goal is to finally restore this split skeleton to its original form. Combining the body part with each of the other 4 parts, we first create the right and left torso of the upper and lower torso. Then, by combining the right and left torsos, the upper and lower skeletons are made, and finally, by combining the upper and lower torsos we get the final skeleton. Figure5 illustrates how this mechanism happens.

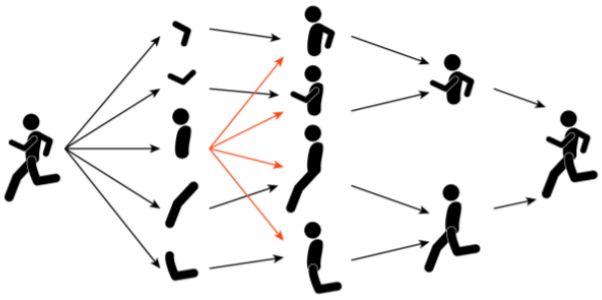


Figure 5: **LSTM unit architecture.**

4.3. Training the model

By combining two previous structures, I constructed the deep neural network which takes advantage of self-attention gates. By passing the result through a fully-connected layers and using softmax activation function the task of action classification is done. Figure6 shows the model architecture and Figure7 indicates how this model is capable of analyzing various movements in 3D space.

Regarding model training, this model uses hold-out approach to train and fine tune the model. By considering 70% of the data as training set and 20% as validation set the mentioned processes take place. Once it gets done, 250 epochs is considered to train the model on 90% of data. Figure8 shows the provided accuracy and loss by training the model on the combination of training and validation set.

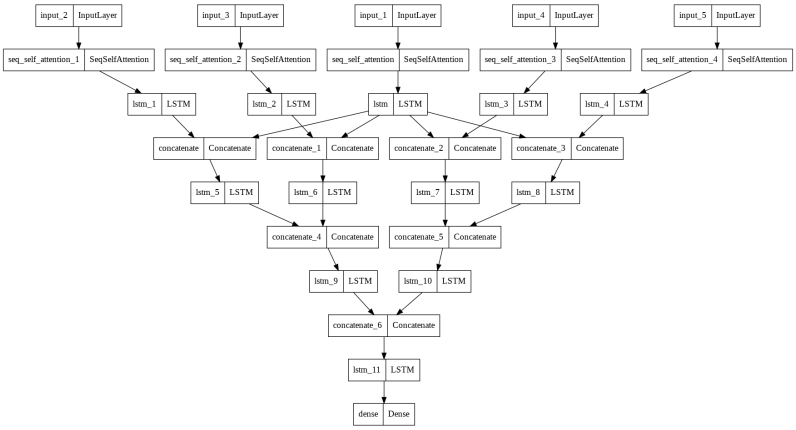


Figure 6: **Model architecture.**

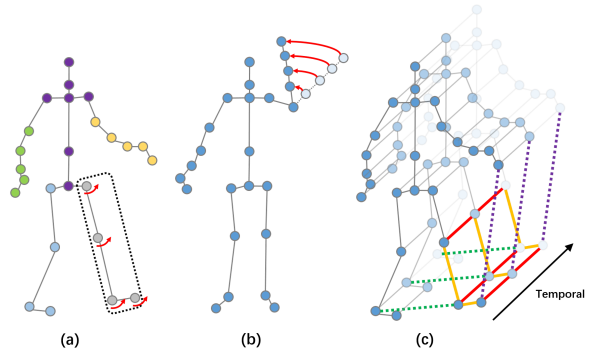
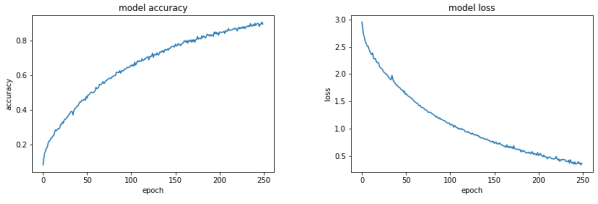


Figure 7: **Considering various movements and analyze them.**



(a) Accuray (b) Loss

Figure 8: **Quality measurement of the training process.**

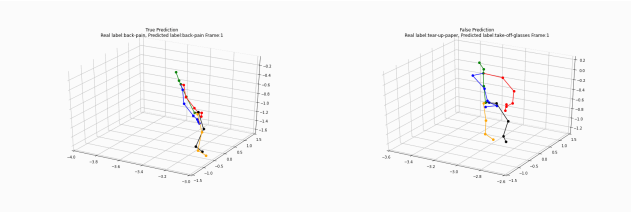
5. Federated learning

As the final step, this study reviews how to provide a federated space to learn the model using decentralized clients.

Federated learning enables multiple actors to build a common, robust machine learning model without sharing data, thus allowing to address critical issues such as data privacy, data security, data access rights and access to heterogeneous data[19, 4, 29]. Since this kind of learning does not manipulate the main structure of the model, the performance of the model is not expected to change. In other words, after bringing the model into this environment we do not expect the accuracy to increase or decrease.

6. Results and improvements

The performance of the trained model is evaluated using 400 data samples. These sample are distributed fairly in different 20 categories. The accuracy provided by this model is 80.5% on unseen data. By consistently reviewing the result and compare the similarity between wrong-labeled data samples to their original labels and then adjusting the model structure, the model is capable to provide this number of accuracy. Since the error produced by the model is meaningful, we claim that the model has been trained correctly. Figure9 provides two data samples, the first one is predicted truly by the model and the second one is wrongly classified by the model.



(a) True (b) False

Figure 9: **Model results.** This figure shows that the wrongly-labeled data are not too different from the concept of their true labels.

Regarding doable improvements on the result of this research, we can consider two potential jobs. The first is to using much more number of data to train the model. The most prominent edge associated with deep learning models compared to traditional machine learning approaches is that if the number of data increases, the performance of deep learning models grow up accordingly. So the first possible improvement is to consider more data sample per label. The second job can be to consider a deeper model and do fine tuning systematically and not by trial and error.

7. Conclusion

According to the outputs obtained in the previous part, we showed to what extent a hierarchical structure can shift the focus of the network to the location of activities. As mentioned, most of the activities we do are performed only with a specific part of our body and other parts of the body are at rest. This phenomenon was used and a structure was presented that by dividing the skeleton of the body, we can enter each part into the neural network separately. However, we did not suffice on this and by designing a unique attention mechanism, we weighted each part of the body at the very beginning of the model. Finally, we were able to implement an activity detection system with acceptable accuracy.

But the point that is not addressed in this study is the lack of a mechanism to announce the starting point of an activity and its end. This system is programmed in such a way that the coordinate frames of the skeletal joints are entered during an activity, then it is divided into specified components and enters the neural network to be labelled by the system. However, unlike the laboratory environment where the beginning and end of activities are controlled, in online applications, the starting and ending point of each activity is not known. Hence, this topic can be considered for the future research.

References

[1] Jake K Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):1–43, 2011.

[2] Jake K Aggarwal and Lu Xia. Human activity recognition from 3d data: A review. *Pattern Recognition Letters*, 48:70–80, 2014.

[3] Victoria Bloom, Dimitrios Makris, and Vasileios Argyriou. G3d: A gaming action dataset and real time action recognition evaluation framework. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 7–12. IEEE, 2012.

[4] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[6] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[7] Alireza Fathi, Yin Li, and James M Rehg. Learning to recognize daily actions using gaze. In *European Conference on Computer Vision*, pages 314–327. Springer, 2012.

[8] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[9] Abhinav Gupta, Aniruddha Kembhavi, and Larry S Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(10):1775–1789, 2009.

[10] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Yan Huang, Wei Wang, and Liang Wang. Bidirectional recurrent convolutional networks for multi-frame super-resolution. *Advances in neural information processing systems*, 28:235–243, 2015.

[13] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[14] Yu-Gang Jiang, Subhabrata Bhattacharya, Shih-Fu Chang, and Mubarak Shah. High-level event recognition in unconstrained videos. *International journal of multimedia information retrieval*, 2(2):73–101, 2013.

[15] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & psychophysics*, 14(2):201–211, 1973.

[16] Gunnar Johansson. Visual motion perception. *Scientific American*, 232(6):76–89, 1975.

[17] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[18] Tian Lan. *Beyond actions: Discriminative models for contextual group activities*. PhD thesis, Applied Science: School of Computing Science, 2010.

[19] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, page 106854, 2020.

[20] Yin Li, Zhefan Ye, and James M Rehg. Delving into egocentric actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 287–295, 2015.

[21] Syed Zain Masood, Christopher Ellis, Adarsh Nagaraja, Marshall F Tappen, Joseph J LaViola, and Rahul Sukthankar. Measuring and reducing observational latency when recognizing actions. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 422–429. IEEE, 2011.

[22] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[23] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[24] Samsu Sempena, Nur Ulfa Maulidevi, and Peb Ruswono Aryan. Human action recognition using dynamic time warping. In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, pages 1–5. IEEE, 2011.

[25] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *arXiv preprint arXiv:1406.2199*, 2014.

[26] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852. PMLR, 2015.

[27] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Human activity detection from rgbd images. In *Workshops at the twenty-fifth AAAI conference on artificial intelligence*, 2011.

[28] Lu Xia, Chia-Chih Chen, and Jake K Aggarwal. View invariant human action recognition using histograms of 3d joints. In *2012 IEEE computer society conference on computer vision and pattern recognition workshops*, pages 20–27. IEEE, 2012.

[29] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207, 2019.

[30] Kiwon Yun, Jean Honorio, Debaleena Chattopadhyay, Tamara L Berg, and Dimitris Samaras. Two-person interaction detection using body-pose features and multiple instance learning. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 28–35. IEEE, 2012.