Human Action Recognition with CNN-LSTM

Aiganym Ospan
Computer Science Department
Nazarbayev University
Nur-Sultan, Kazakhstan
aiganym.ospan@nu.edu.kz

Nuren Zhaksylyk
Computer Science Department
Nazarbayev University
Nur-Sultan, Kazakhstan
nuren.zhaksylyk@nu.edu.kz

Ruslan Imanberdiyev
Computer Science Department
Nazarbayev University
Nur-Sultan, Kazakhstan
ruslan.imanberdiyev@nu.edu.kz

Temirlan Bazarzhan
Computer Science Depratment
Nazarbayev University
Nur-Sultan, Kazakhstan
temirlan.bazarzhan@nu.edu.kz

Yerdaulet Kappar
Mathematics Department
Nazarbayev University
Nur-Sultan, Kazakhstan
yerdaulet.kappar@nu.edu.kz

Yerkebulan Ratkhan
Computer Science Department
Nazarbayev University
Nur-Sultan, Kazakhstan
yerkebulan.ratkhan@nu.edu.kz

***Abstract*—Human action recognition (HAR) is a growing research area in the field of computer vision. In such systems, identifying objects, observing the environment, and predicting the specific patterns of the motions in the video sequences have been recent developments in this research field. This paper introduces a human action recognition method using convolutional neural networks (CNN) and long short-term memory (LSTM) to predict human behaviors. The combination of these two neural networks is used for feature selection and preserving the previous information which will be used further in labeling human activities. This paper implements Long-term Recurrent Convolutional Network (LRCN) for human action recognition. The model was fitted with the KTH dataset. Our implementation of LRCN achieves total accuracy of 78.3%. Results can be further increased with fine-tuning through cross validation and using different frame extraction methodologies. You can find the code of our model here.**

I. Introduction

Since physical abuse has been an inseparable part of child upbringing in most countries for many centuries, it still has been practiced in the modern world. One of the widespread problems that touch upon the violence in the education sector is the kindergarten teacher-perpetrated violence. This is very common issue in Kazakhstan that should be addressed as soon as possible[9]. In an absence of violence detection cameras, physical abuse toward the kids is detected only by detecting bruises on the body of kids.

Monitoring the cameras could not solve this problem, since the cases such as deleting some video scenes with physical abuse should not be ignored. Thus, autonomic detection of abnormal activities in the education sector is one of the solutions that will be touched upon in this paper. In this work, we aim to show that with the help of Human Action Recognition (HAR) it is possible to detect human actions from the continuous video sequences. This technology then may be used in combination with cameras in kinder-garden. The system will automatically send the scenes of physical abuse directly to governmental structures which is needed to disable abusers to delete these scenes.

Human action recognition is a significant problem in computer vision research. This is mostly because recognizing the pattern of human actions in the natural environment is a challenging task, considering the spatio-temporal features, cluttered and uneven background, etc [6]. However, with deep learning models, recognizing certain activities from raw sensor data has become available. A deep learning model can extract relevant poster representation features and predict further actions in an efficient way.

The proposed approach is LRNC, long-term recurrent convolutional Networks. LRCN approach combines Convolution and LSTM layers in a single model [1]. The Convolutional layers are used for spatial feature extraction from the frames, and the extracted spatial features are fed to the LSTM layers at each time-steps for temporal sequence modeling. This way the network learns spatio-temporal features directly in an end-to-end training, resulting in a robust model.

In this study, a standard KTH dataset was used to predict the proposed method. The dataset includes six actions (walking, jogging, running, boxing, hand-waving, and hand-clapping). First, different existing action recognition methodologies will be discussed. Then proposed CNN+LSTM model will be discussed in detail in the following sections. Precision, recall and f1 scores along with accuracy for each 6 classes will be evaluated.

II. Literature Review

In this section, we give some background information about proposed deep learning algorithms and already existing model in human action recognition (HAR).

Action recognition is a challenging task because extracting spatial and temporal information from video sequences is difficult. There are several different deep learning models that are used in HAR. The first approach is autoencoders. Autoencoders have two parts: encoders and decoders. The purpose of encoder is to map the inputs into representation of data, while the purpose of decoder is to reconstruct the input from the representation of data [4]. One of the types of autoencoders is transformers. Transformers are less popular in HAR, since they are mostly used in natural language processing (NLP) problems. Usually, Transformers exploits CNN or RNN on top of the model [5] for HAR purposes.

The second approach is convolutional neural networks (CNN). CNN layers can be combined to make deep neural network and to increase the accuracy of the model. Usually, CNN is used on images or time-series data [4]. However, CNN can be not accurate in sequential data [4]. CNN considers spatial information, but not the temporal information. This can affect the efficiency of the model [7].

The third approach is recurrent neural networks (RNN). RNNs are specialized in processing of sequential data (speech or video)[4]. However, RNN have the gradient vanishing and exploding problems that can hamper convergence. These problems are caused due to the memory space produce by recurrent connections [4]. One of the solutions to these problems is Long Short-Term Memory (LSTM) RNN [4]. LSTM computes the hidden layer by adding previous hidden state rather than multiplying as in RNN [10].

In our model, we have decided to use the advantages of both CNN and LSTM models. Since LSTM can process videos and extract temporal features and CNN can extract spatial information from specific video frames, our model uses the combination of CNN and LSTM. Therefore, we used the model proposed by [1]. Their model obtained an accuracy of 87.6% on the UCF-101 dataset. Our model will use the KTH dataset.

III. Methodology

All the training and testing was done on the Google Collaboratory. The optimal batch size was chosen to be 4 that did not overload RAM of the CPU.

A. Preprocessing

Firstly, the video frames are loaded from the dataset. Then, video frames are resized to the fixed height and width of 60×60 and the data is normalized to range [0-1] by dividing each pixel to the 255 in order to reduce computations and training time. Each video was divided into 40 frames and fed to the model after dividing dataset into training and testing data.

B. CNN model

Convolutional neural networks (CNN) are deep learning model that processes image data. A convolutional layer can contain several convolutions, pooling, and fully connected layers and diminish the size of the input in each layer [2]. In the convolutional layer, the kernel matrix is multiplied by the input matrix for every filter to identify edges in input [8]. For example, if the input, filter, and kernel dimension is $N \times N$, M, $P \times P$, then the output will be a matrix with dimension $(M \times (N – P + 1) \times (N - P + 1))$ (not considering padding and stride). In our model, the first convolution layer has used 16 filters and a kernel with dimensions 3 x 3. For the second, third, and fourth convolution layers the dimensions of filter and kernel are (32, 3, 3), (64, 3, 3), and (64, 3,3), respectively. Then, the output of the convolution layer goes to a pooling layer as an input. The purpose of the pooling layer is to obtain the maximum for the cluster of the same dimension as the pool matrix from input to reduce the dimension of input. In other words, if the dimension of input and pool is $N \times N$ and $M \times M$, then the pooling layer returns a matrix with dimension $\frac{N}{M} \times \frac{N}{M}$. The pool matrix dimension for the first, second, third, and fourth pooling layer of our model is (4, 4), (3, 3), (2, 2), and (2, 2). The last layer, which is a fully connected layer, flattens the output matrix of the pooling layer. For example, if the input dimension is $N \times N$, then it converts it to a matrix with dimension $(2N, 1)$. As an activation function our model exploits ReLU in the convolution layers and SoftMax in the last dense layer. The stride is 1 and there is no padding. In general, CNN can be visualised as in Figure1.

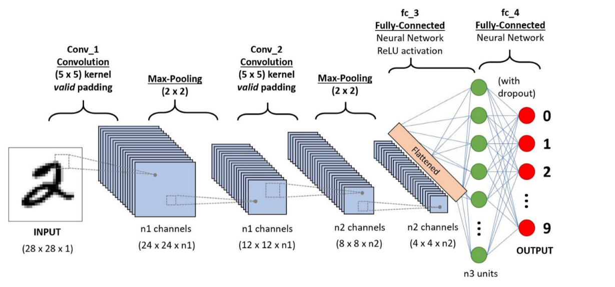


Fig. 1. CNN model

C. CNN-LSTM model

The CNN-LSTM model combines CNN model and Recurrent Neural Network (RNN) as it demonstrated in Figure 2 [1]. CNN-LSTM model passes the input x_t and previous hidden state h_{t-1} to an output z_t and current hidden state h_t. In general, the formula be as following:

$$h_t = f_W(x_t, h_{t-1}) \tag{1}$$

The general structure of our CNN-LSTM model is shown in Figure 3. Our model has used a sequence length of 40 frames. It just slices all video frames into a group of 40 sequential frames. The number of units in a dense layer equals to the number of classes in dataset, that is 6. The dropout value is 25%.

Our model's hyper-parameters: categorical cross-entropy as loss function, 'Adam' optimizer, accuracy metric, 80 epochs. Max pooling as pooling layer and ReLU activation function in convolutional layers.

In the way of search for best model for our data set, number of convolutional and pooling layers, loss function, activation function and filters of convolutional layers were changed. The best model, in terms of total accuracy on test data is represented in Figure3.

IV. Dataset

For training and testing, we have used the KTH dataset. This dataset consists of 6 classes of human actions such as running,

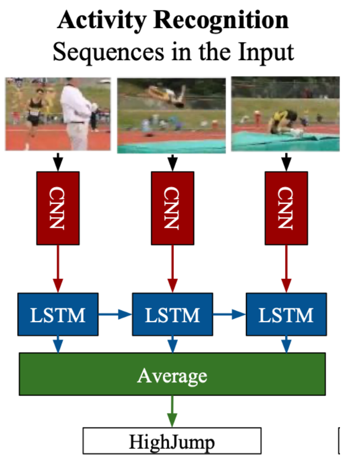


Fig. 2. CNN-LSTM model

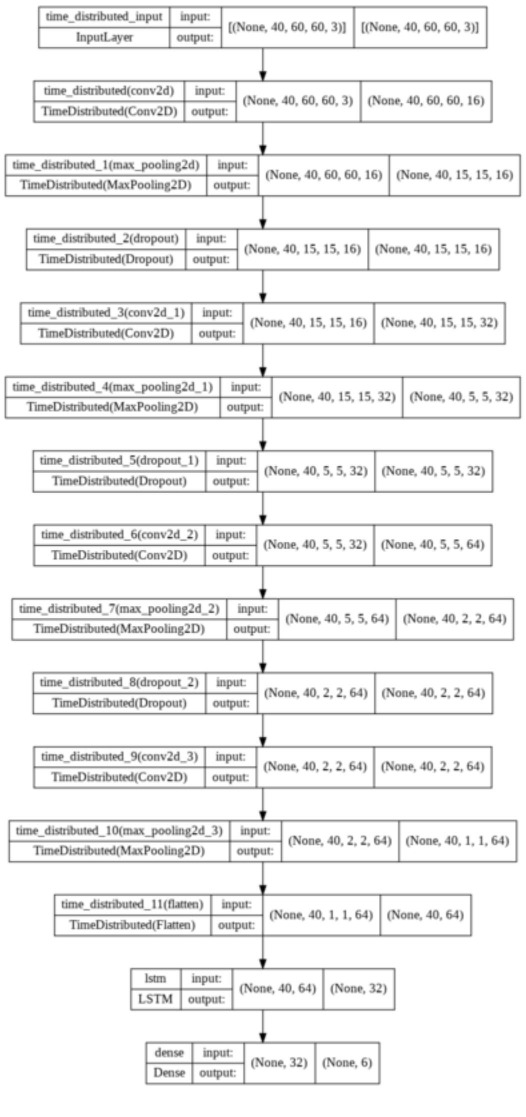

time_distributed_input | InputLayer | input: [(None, 40, 60, 60, 3)] | output: [(None, 40, 60, 60, 3)]
time_distributed(conv2d) | TimeDistributed(Conv2D) | input: (None, 40, 60, 60, 3) | output: (None, 40, 60, 60, 16)
time_distributed_1(max_pooling2d) | TimeDistributed(MaxPooling2D) | input: (None, 40, 60, 60, 16) | output: (None, 40, 15, 15, 16)
time_distributed_2(dropout) | TimeDistributed(Dropout) | input: (None, 40, 15, 15, 16) | output: (None, 40, 15, 15, 16)
time_distributed_3(conv2d_1) | TimeDistributed(Conv2D) | input: (None, 40, 15, 15, 16) | output: (None, 40, 15, 15, 32)
time_distributed_4(max_pooling2d_1) | TimeDistributed(MaxPooling2D) | input: (None, 40, 15, 15, 32) | output: (None, 40, 5, 5, 32)
time_distributed_5(dropout_1) | TimeDistributed(Dropout) | input: (None, 40, 5, 5, 32) | output: (None, 40, 5, 5, 32)
time_distributed_6(conv2d_2) | TimeDistributed(Conv2D) | input: (None, 40, 5, 5, 32) | output: (None, 40, 5, 5, 64)
time_distributed_7(max_pooling2d_2) | TimeDistributed(MaxPooling2D) | input: (None, 40, 5, 5, 64) | output: (None, 40, 2, 2, 64)
time_distributed_8(dropout_2) | TimeDistributed(Dropout) | input: (None, 40, 2, 2, 64) | output: (None, 40, 2, 2, 64)
time_distributed_9(conv2d_3) | TimeDistributed(Conv2D) | input: (None, 40, 2, 2, 64) | output: (None, 40, 2, 2, 64)
time_distributed_10(max_pooling2d_3) | TimeDistributed(MaxPooling2D) | input: (None, 40, 2, 2, 64) | output: (None, 40, 1, 1, 64)
time_distributed_11(flatten) | TimeDistributed(Flatten) | input: (None, 40, 1, 1, 64) | output: (None, 40, 64)
lstm | LSTM | input: (None, 40, 64) | output: (None, 32)
dense | Dense | input: (None, 32) | output: (None, 6)

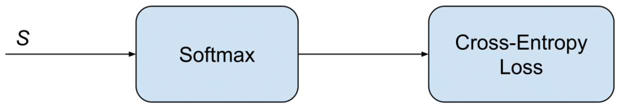
Fig. 3. Structure of our model

walking, jogging, handclapping, handwaving, and boxing (see Figure 4). These actions are carried out by 25 different people in different situations. The number of videos is 605 of size 160×120. The dataset is divided into a training set (80% of data) and a test set (20% of data). Then training dataset was further divided into 80% training and 20% validation datasets. Thus, the model was trained on 64% of the total data and then was validated on 16% of the total data.

Fig. 4. Dataset

V. Evaluation Metric

Categorical cross entropy loss function with an optimizer adaptive moment estimation (Adam) based on accuracy metric was used (see in Figure 5). The optimal hyper-parameters such as frame size (60 × 60), sequence length (40), number of epochs (80) and batch size (4) were found by hand. We have tried to implement GridSearchCV for fine tuning, but there was not enough RAM to implement cross validation. So model was trained several times with different above mentioned parameters and the best was chosen. The models were evaluated using precision, recall and f1-score metrics for each class along with total accuracy of the model. Formulas are given in Figure 6



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

Fig. 5. Categorical Cross Entropy

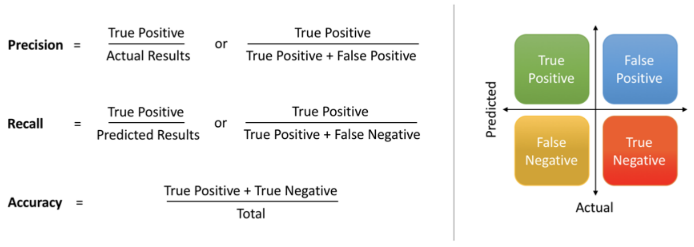
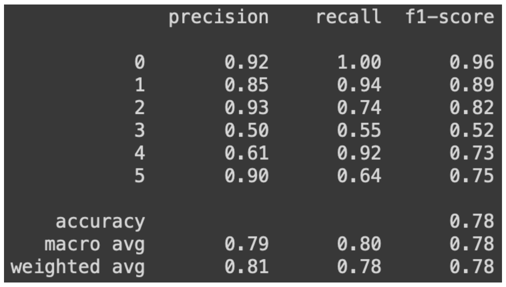

Precision = True Positive / Actual Results or True Positive / True Positive + False Positive
Recall = True Positive / Predicted Results or True Positive / True Positive + False Negative
Accuracy = True Positive + True Negative / Total
True Positive
False Positive
False Negative
True Negative
Predicted
Actual

Fig. 6. Precision, Recall and Accuracy formula

VI. Results and Analysis

Our CNN-LSTM model is tested on KTH dataset obtained an accuracy of 78%. To prevent the overfitting, our model exploited a dropout that randomly set the input values to 0 at random frequency. Also, early stopping algorithm with patience of 15 was used to stop on early epochs when the model starts to overfit on training data. Total accuracy on unseen dataset was 78%, however our model is not performing well on all classes. Jogging class showed lowest precision, recall and f1-score (see in Figure 7). The reason for such dis-balance in performance of different classes is unknown for us. Data samples have the same size for all classes, so under-sampling is not reason. So it might be due to frame extraction algorithm that we have used. We have used the simplest method of frame extracting just by dividing the video into 40 frames which means that the accuracy of our model is directly proportional to the number of chosen sequences. It is inefficient because this also increases computational costs of the model. So it is recommended to use different methods such as Optimal key frame selection method, 3D augmentation method or Motion analysis method [3] for future improvements. Also it is recommended to repeat this experiment on more powerful CPUs that has bigger RAM as it gives an opportunity to fine-tune the model using cross validation.

Source code:http://github.com/ZhNuren/action_recognition



```
              precision    recall  f1-score

           0       0.92      1.00      0.96
           1       0.85      0.94      0.89
           2       0.93      0.74      0.82
           3       0.50      0.55      0.52
           4       0.61      0.92      0.73
           5       0.90      0.64      0.75

    accuracy                           0.78
   macro avg       0.79      0.80      0.78
weighted avg       0.81      0.78      0.78
```

Fig. 7. Evaluation metrics for each class

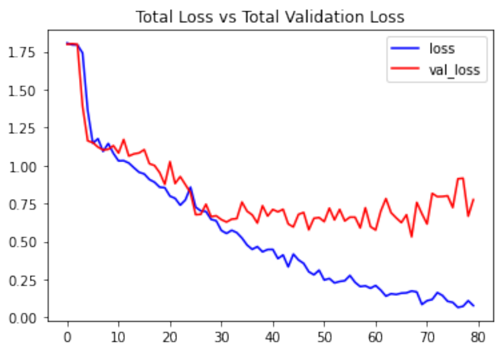


Fig. 8. Total loss vs total validation loss

Change of the total loss and total validation loss by each epoch can be seen in Figure 8.

Change of the total accuracy and total validation accuracy by each epoch can be seen in Figure 9.

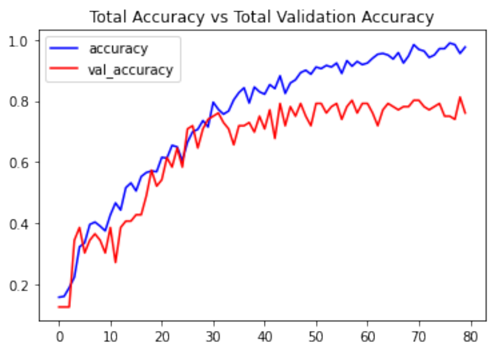


Fig. 9. Total Accuracy vs Total Validation Accuracy

References

[1] Jeff Donahue et al. "Long-term Recurrent Convolutional Networks for Visual Recognition and Description". In: *CoRR* abs/1411.4389 (2014). arXiv: 1411.4389. URL: http://arxiv.org/abs/1411.4389.

[2] Giovanni Ercolano, Daniel Riccio, and Silvia Rossi. "Two deep approaches for ADL recognition: A multi-scale LSTM and a CNN-LSTM with a 3D matrix skeleton representation". In: *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. 2017, pp. 877–882. DOI: 10.1109/ROMAN.2017.8172406.

[3] Ujwalla Gawande, Kamal Hajari, and Yogesh Golhar. "Deep learning approach to key frame detection in human action videos". In: *Recent Trends in Computational Intelligence* 1 (2020), pp. 1–17.

[4] Fuqiang Gu et al. "A Survey on Deep Learning for Human Activity Recognition". In: *ACM Comput. Surv.* 54.8 (Oct. 2021). ISSN: 0360-0300. DOI: 10.1145/3472290. URL: https://doi.org/10.1145/3472290.

[5] Vittorio Mazzia et al. "Action Transformer: A Self-Attention Model for Short-Time Human Action Recognition". In: *CoRR* abs/2107.00606 (2021). arXiv: 2107.00606. URL: https://arxiv.org/abs/2107.00606.

[6] Xi Ouyang et al. "A 3D-CNN and LSTM Based Multi-Task Learning Architecture for Action Recognition". In: *IEEE Access* 7 (2019), pp. 40757–40770. DOI: 10.1109/ACCESS.2019.2906654.

[7] Meet Pandya, Abhishek Pillai, and Himanshu Rupani. "Segregating and Recognizing Human Actions from Video Footages Using LRCN Technique". In: *Advanced Machine Learning Technologies and Applications*. Ed. by Aboul Ella Hassanien, Roheet Bhatnagar, and Ashraf Darwish. Singapore: Springer Singapore, 2021, pp. 3–13. ISBN: 978-981-15-3383-9.

[8] Fatemeh Serpush and Mahdi Rezaei. "Complex human action recognition using a hierarchical feature reduction and deep learning-based method". In: *SN Computer Science* 2.2 (2021), pp. 1–15.

[9] TENGRINEWS. *Nasilie v detskih sadah: ekspert vyiskazalsya o prichinah problemy*. URL: https://

tengrinews.kz/kazakhstan_news/nasilie-detskih-sadah-ekspert-vyiskazalsya-prichinah-434308/. (published: 14 April 2021).

[10] Gary Thung and Helen Jiang Stanford. “A Torch Library for Action Recognition and Detection Using CNNs and LSTMs”. In: 2016.