Online Hand Gesture Recognition Using 3D Convolutional Neural Networks

Yinghao Qin
190189237
Tijana Timotijevic
MSc FT Artificial Intelligence with Industrial Experience

***Abstract*—In human computer interaction, real-time detection and classification of dynamic hand gestures is challenging as: 1) the system must run in a real-time video stream and there is no noticeable lag in response after performing a gesture; 2) there is a large difference in how people perform gestures, making recognition more difficult. In this paper, an online hand gesture recognition system is proposed, which is able to localize gestures in real-time video stream and recognize what these gestures are. To improve the robustness of the system, the sliding window approach is used to refine results from multiple windows. All of the models in my project are trained on Jester database, achieving 98+% accuracy for detector and 90+% accuracy for classifier. For the overall performance of the system, the best group can respond within three seconds and reach 37.5% Levenshtein accuracy on the homemade dataset. The project codes used in this work are publicly available.**

***Index Terms*—online, hand gesture recognition, 3D CNN, Levenshtein metric**

I. Introduction

Recently, hand gesture recognition has attracted more and more attention in human machine interaction. It can be applied in numerous fields, such as sign language recognition, in-vehicle gestural interface, video surveillance system, gaming and virtual reality control etc. By and large, there are two kinds of hand gesture recognition technologies. First, wearable sensor devices (Abhishek et al., 2016; Haba et al., 2018; Jung et al., 2015) can precisely track information such as hand position, movement velocity, acceleration etc. From the data, hand gesture can be inferred easily, whereas the approach is unrealistic for most people due to its inconvenience and additional cost. Second, computer-vision based methods require large amount of data to train the system for generalizing the unknown scenarios instead of requiring extra devices. Nowadays, the most popular approach in the field of dynamic hand gesture recognition is using deep convolutional neural networks (CNN) to learn and generalize visual information e.g., (Köpüklü et al., 2019; Molchanov et al., 2016, 2015a; Abavisani et al., 2019; Kopuklu et al., 2018).

Although there have been a lot of deep learning researches about dynamic gesture recognition, most of them just focus on how to learn more representative spatiotemporal features in pre-segmented video clips. Indeed, this has spawned a number of works with high classification accuracy on many well-known databases. However, few researchers put their sights on the real-world application of hand gesture recognition. This paper is an attempt to bridge the gap between them.

Notably, dynamic hand gesture recognition in the real world still faces numerous challenges. Firstly, how to identify when a gesture begins and ends in the video, which is actually a temporal action localization/recognition task. In addition, the system should be robust to uncertain environment such as illumination, background, occlusion. Furthermore, when people perform hand gestures, the temporal scale is required to be considered since people gesture with different duration. Finally, we should take cost and efficiency into account as well, i.e. the system should cost relatively little computational resources and react quickly.

There are basically two core tasks in real-world system for dynamic hand gesture recognition - temporal gesture localization and gesture classification. The former is designed to localize the gesture proposals in the video stream, and the latter aims to gain a high classification accuracy with low consumption of resources.

Overall, the contributions of this project are mainly in three aspects. 1) Inspired by Köpüklü et al. (2019), an online hand gesture recognition system is designed, which can respond to hand gestures within 3 seconds and achieve 37.5% Levenshtein accuracy in self-collected dataset; 2) the framework is systematically evaluated from components to the entire system using the Levenshtein evaluation metric; 3) a couple of 3-dimensional (3D) CNN models trained on Jester database (Materzynska et al., 2019) are obtained.

The rest of the paper is organized as follow. Section II reveals the related work regarding action and gesture recognition. In section III, the online hand gesture recognition system is described in detail. Next, section IV presents the experiment results and evaluations. The paper is then concluded in section V. Lastly, section VI gives some ideas about future work.

II. Related work

Many handcrafted spatiotemporal features for visual data analysis are first developed in the area of action and gesture recognition. These methods typically extract features such as shape, appearance, optical flow to perform gesture classification. In their paper, Shen et al. (2012) state an exemplar-based approach for dynamic hand gesture recognition, which first detects salient regions from motion divergence fields and then extracts local descriptors to index gesture from the pre-trained vocabulary. Tamrakar et al. (2012) systematically evaluate the efficacy of action recognition using 7 low-level static

and dynamic features, all of which are represented by BoW descriptors and an SVM approach is used to classify gesture. Ohn-Bar and Trivedi (2014) apply a descriptor combining RGB and depth features to classify hand gestures, and meanwhile propose an architecture with two independent modules to detect hand gestures and recognize them. Wang et al. (2016a) introduce a video representation by deriving feature points matches from multiple frames using SURF descriptors and dense optical flow, and also compare two feature encoding approach - Fisher vector and Bag-of-words histogram.

After that, due to the breakthrough of CNNs in static images, many people try to apply 2D CNNs in video analysis. The typical method is to first extract features from individual frame using 2D CNNs, then take fusion measures to track motion of objects in time dimension, e.g., (Karpathy et al., 2014; Simonyan and Zisserman, 2014). In addition, Simonyan and Zisserman (2014) also apply CNNs in stacked optical flow maps from consecutive frames to generalize action in video. Long term recurrent convolutional networks are proposed by Donahue et al. (2015), where 2D CNNs are used to derive visual features from each frame and these features are fed into a stack of recurrent sequence models (LSTMs) to generalize actions. Temporal segment network (TSN) is proposed by Wang et al. (2016b), based on the idea of long-range temporal structure modeling, uses 2D CNNs to extract spatial information from color modality and temporal information from optical flow modality in uniformly distributed samples.

Gradually, some researchers begin to explore whether they could use a single CNN framework to learn spatiotemporal features instead of using multiple 2D CNNs. To the best of my knowledge, the idea of 3D CNNs for action recognition is first stated by Ji et al. (2012). Tran et al. (2015) propose a homogeneous convolutional 3D architecture with small $3\times3\times3$ convolution kernels, where the input data has 4 dimensions including the number of channels, the number of frames in a clip, the height and width of the frame. Molchanov et al. (2015b) fuse multi-modalities data to train the 3D CNN model for hand gesture recognition of driver. In the following, 3D ResNets, ResNeXts, DenseNets are subsequently used for spatiotemporal information learning (Hara et al., 2017, 2018).

In addition, there are other developments in hand gesture and action recognition. Firstly, many hand gesture and action databases have appeared in recent years, such as Sports-1M (Karpathy et al., 2014), NVGesture (Molchanov et al., 2016), EgoGesture (Zhang et al., 2018), Jester (Materzynska et al., 2019) etc. Kopuklu et al. (2018) propose to fuse motion information into still images as the spatiotemporal representatives of an action, namely Motion Fused Frames (MFFs). They claim that it is the first time that data-level fusion has been applied to deep-learning based motion and gesture recognition. Molchanov et al. (2016) state a system, simultaneously detecting and classifying hand gesture from online video stream, which is composed of a recurrent 3D CNN and a connectionist temporal classifier, CTC. A real-time hand gesture recognition system is proposed by Köpüklü et al. (2019), which contains a detector and a classifier. In addition, they also recommend the Levenshtein distance as the evaluation metric.

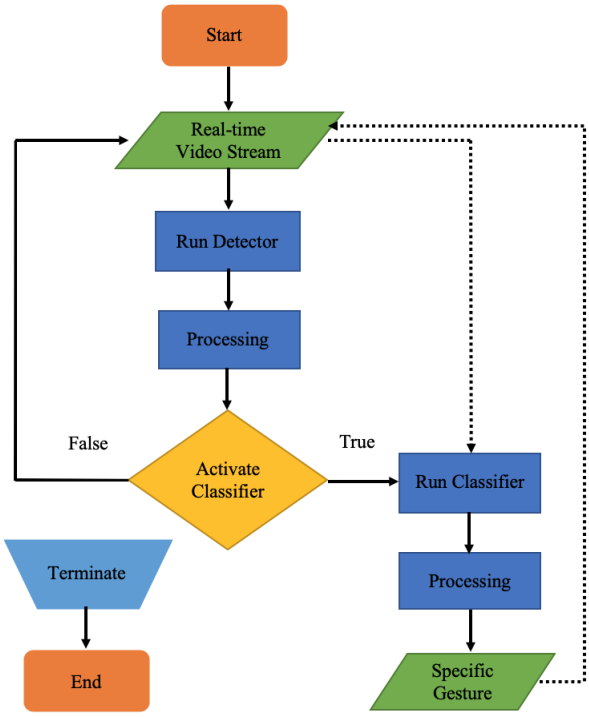

Fig. 1. The flow chart of the online hand gesture recognition system.

III. Approach

In this section, I elaborate on the hierarchical architecture which is used to recognize dynamic hand gesture, and then describe the detection and classification methods. By the way, the Levenshtein evaluation metric is also introduced.

A. Architecture

The subsection first demonstrates the overall architecture and sliding window approach of the system and then analyzes qualities the system should meet. Following that, the evaluation metric on the overall system is introduced.

The workflow of the hand gesture recognition system is designed as shown in Fig. 1. It consists of two phases - detection and classification. In detection stage, real-time video stream captured by RGB sensors is fed into the system, meanwhile the detector runs to distinguish between 'gesture' and 'no gesture' classes from the video stream, and then the detection results are further processed to improve the detection performance. Afterwards, we activate the classifier if 'gesture' exists based on the detection results, otherwise update the video stream for the next round of detection. When the system enters the classification phase, video stream is fed into the classifier to generate probabilities of different gesture classes, which are then further processed to predict predefined gestures. In addition, the system can be manually terminated at any time the system is running.

TABLE I
THE ARCHITECTURES OF RESNET-10 AND RESNEXT-101

layer	output	ResNet-10	ResNeXt-101
conv1	dx56x56	3×7×7, 16, stride=(1,2,2)	3×7×7, 64, stride=(1,2,2)
		3×3×3 max pool, stride=2	3×3×3 max pool, stride=2
conv2	d/2×28×28	$\begin{bmatrix} 3\times3\times3,\ 16 \\ 3\times3\times3,\ 16 \end{bmatrix} \times 1,\ 16$	$\begin{bmatrix} 1\times1\times1,\ 128 \\ 3\times3\times3,\ 128,\ C=32 \\ 1\times1\times1,\ 128 \end{bmatrix} \times 3,\ 128$
conv3	d/4×14×14	$\begin{bmatrix} 3\times3\times3,\ 32 \\ 3\times3\times3,\ 32 \end{bmatrix} \times 1,\ 32$	$\begin{bmatrix} 1\times1\times1,\ 256 \\ 3\times3\times3,\ 256,\ C=32 \\ 1\times1\times1,\ 512 \end{bmatrix} \times 3,\ 256$
conv4	d/8×7×7	$\begin{bmatrix} 3\times3\times3,\ 64 \\ 3\times3\times3,\ 64 \end{bmatrix} \times 1,\ 64$	$\begin{bmatrix} 1\times1\times1,\ 512 \\ 3\times3\times3,\ 512,\ C=32 \\ 1\times1\times1,\ 1024 \end{bmatrix} \times 3,\ 512$
conv5	⌈d/16⌉×4×4	$\begin{bmatrix} 3\times3\times3,\ 128 \\ 3\times3\times3,\ 128 \end{bmatrix} \times 1,\ 128$	$\begin{bmatrix} 1\times1\times1,\ 1024 \\ 3\times3\times3,\ 1024,\ C=32 \\ 1\times1\times1,\ 2048 \end{bmatrix} \times 3,\ 1024$
	1×1×1	average pool, 128-d fc, softmax	average pool, 2048-d fc, softmax
	# params	0.90×10^6	70.49×10^6

The sliding window method is applied to further process the original results from detector and classifier. Intuitively, using data from multiple sliding windows can increase prediction confidence as these windows cover more useful information. For the input data (real-time video stream), consecutive frames are encapsulated into a clip as a 4-dimensional input value. For the sake of simplicity, we represent video clips with the sign of $c \times d \times h \times w$, where c refers to the number of channels, d is the depth of the clip, and h and w represents the height and width of the frame respectively. As shown in Fig. 2 and Fig. 3, the detector and classifier are fed by video clips with size n and m respectively. In details, a detector clip contains only 8 frames, whereas a 16-frame or 32-frame video clip is used in classification stage. For each round of detection, four successive clips with stride of s are used to judge whether there is a gesture in the area, as shown in Fig. 2. In this experiment, the stride s of detector is set as 1. In the classification stage, multiple windows are used to decide which specific gesture it is. This specific method is explained in the classification subsection.

Besides, the dynamic hand gesture recognition system should meet certain qualities. First, it should be highly efficient, i.e. the system can respond as quickly as possible. When a gesture ends, the system can return recognition result in no more than three seconds, and we define it is highly efficient. Second, the system should have an acceptable accuracy in recognition, namely 30% Levenshtein accuracy. Third, the system should be compatible with different detectors and classifiers, which allows different models to be changed easily.

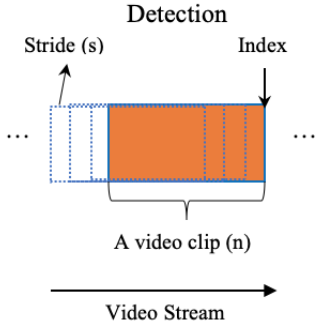

Fig. 2. Process video stream using sliding window in detection stage.

Levenshtein accuracy is proposed by Köpüklü et al. (2019) for online evaluation, which can measure the difference between the prediction results and the ground truth. Levenshtein accuracy of the recognition is calculated by the corresponding Levenshtein distance and the size of the ground truth samples. Levenshtein distance is originally a string metric for measuring the difference between two sequences (Levenshtein, 1966). In details, it is the minimum number of single-character edits (i.e. intersections, deletions or substitutions) required to change one sequence into the other. Specifically, the metric takes into account missing detections, misclassifications and multiple recognitions in the field of gesture recognition. The Levenshtein accuracy calculation formula is presented in (1).

$$LA(prediction) = (1 - \frac{LD(target,\ prediction)}{length(target)}) \times 100\% \tag{1}$$

Where LA denotes the Levenshtein accuracy of the prediction, LD represents the Levenshtein distance between the true target sequence and the prediction sequence, $length(target)$ refers to the length of the true target sequence.

B. Detection

The goal of detection phase is to distinguish gesture and no gesture from the real-time video stream, which is a preparation for the next classification stage. Since the performance of the detector is closely related to the performance of the whole system, thus the detector is required to achieve a high accuracy and be as lightweight as possible. Two steps are taken to attain the goal. Firstly, we need to train a detector, which is actually a binary classification model. Next, after getting this raw detection results, we need to do some post-processing to improve the stability of the results.

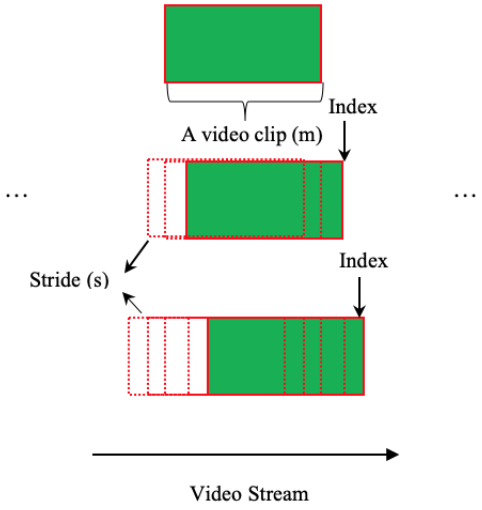

Fig. 3. Process video stream using sliding window in classification stage.

A 10-layer ResNet is chosen as the detector owing to its low complexity and simplicity. In details, the 10-layer ResNet model only has less than 1 million parameters, as shown in Table I. Indeed, Hara et al. (2017) have tried to apply 3D residual networks for action recognition and got exciting results. The basic block of ResNet is shown in the left part of Fig. 4, where the input clip flows into the stacked layers and a shortcut connection, and their outputs finally are combined to generate the new output. The detailed architecture of ResNet-10 in our case is demonstrated in Table I. A $3 \times 8 \times 112 \times 112$ clip inputs the system to generate a 128-dimensional vector. Notably, there is a ceiling brackets in 'conv5' layer for the output of 'd/16', ensuring it is integer value.

After the detector gives multiple detection results using sliding window method, two different methods are employed to make the final detection decisions - the first one utilizes the raw probabilities of the detector predictions and the other one just uses the specific prediction results. Due to the sliding window approach, there is a queue containing k predictions (probabilities or specific prediction results). The size of the queue is selected as 4, as shown in Fig. 2. For the first method, we calculate the average of the values in the queue, and then select the maximum value as the prediction results. For the second one, we just judge whether all of the values are 'gesture' class. If so, the system activates the classifier, otherwise continue the next round of detection.

In addition, in offline experiment, some measures are also taken to refine some proposal areas with gesture. After using the sliding window approach to get multiple original proposals, they will be merged and amended. In details, two consecutive areas are merged into a larger region when the intersection between the two proposal areas is less than the threshold (which is set as 4 in the experiment), and meanwhile some amendments are made to the new proposals. Finally, these proposal areas are used for classification task.

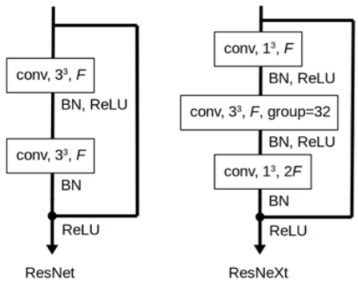

Fig. 4. The basic blocks for ResNet-10 and ResNeXt-101. $conv$, 3^3 or 1^3 represents convolution operation using $3 \times 3 \times 3$ or $1 \times 1 \times 1$ kernel. F, BN, ReLU and group denotes the number of channels, batch normalization, rectified linear unit and cardinality respectively (Köpüklü et al., 2019, p. 4).

TABLE II
THE ARCHITECTURES OF C3D

layer	output	C3D
conv1a	d×112×112	3×3×3, 64, stride=1
conv2a	d×56×56	1×2×2 max pool, stride=(1,2,2)
		3×3×3, 128, stride=1
conv3a,b	d/2×28×28	2×2×2 max pool, stride=2
		[3×3×3, 256, stride=1]×2
conv4a,b	d/4×14×14	2×2×2 max pool, stride=2
		[3×3×3, 512, stride=1]×2
conv5a,b	d/8×7×7	2×2×2 max pool, stride=2
		[3×3×3, 512, stride=1]×2
	d/16×4×4	2×2×2 max pool, stride=2
		max pool, 4096-d fc6&7, dropout=0.5, softmax
# params		78.11×10^6 (16-frame), 111.67×10^6 (32-frame)

C. Classification

Similar to the detection phase, two steps are taken to get the final classification decision – classifier run and post-processing. For the former, different classifiers are trained on Jester dataset with 16-frame clip and 32-frame clip. For the latter, we take advantage of the raw classification probabilities to improve the performance.

For classifier, C3D and ResNeXt-101 models are selected as our classifier models. Tran et al. (2015) propose C3D model, which is one of the initial 3D CNNs and performs well in previous work. The ResNeXt-101 model is first proposed by Xie et al. (2017), by increasing cardinality to gain higher accuracy instead of going deeper or wider. In other words, the ResNeXt structure can improve the accuracy without increasing the complexity of parameters, and meanwhile reduce the number of super parameters. Hara et al. (2018) state that 3D ResNeXt to learn spatiotemporal features for action recognition. The architectures of the two model are demonstrated in Table I and II.

In order to find the balance between the classification accuracy and the reaction time, three different kinds of sliding windows are designed. As shown in Fig. 3, 1, 3, 5 windows of video clips are used to make the final classification decision.

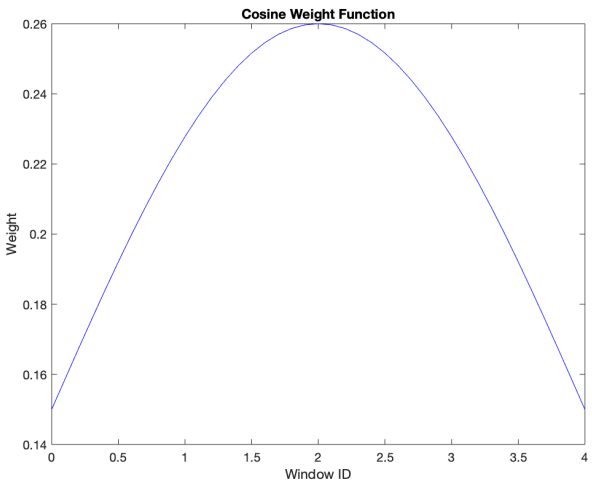

Fig. 5. The cosine weight function according to (2).

The stride of these windows is set as 1. Notably, the windows in the middle should gain more confidence, i.e. the weights corresponding to the middle window should be larger. This is because, according to the theory (Gavrila, 1999; Hinton et al., 2012), there are three temporally overlapping phases in dynamic hand gestures: preparation, nucleus, and retraction, of which the nucleus is the most discriminative. The sliding windows we use should try the best to cover the nucleus part of hand gestures, which probably result in a good performance. Thus, for the 3 windows method (the middle part in Fig. 3) the weights are set as $0.3, 0.4, 0.3$ respectively. And for the 5 windows method (the bottom part in Fig. 3), the weight is formulated as a cosine function as shown in (2).

$$w_i = 0.11 \times cos\left(\frac{\pi}{4} \cdot x - \frac{\pi}{2}\right) + 0.15 \tag{2}$$

Where w_i is the weight of the i^{th} window and x is the window identifier. And the weight function plot is shown in Fig. 5.

Also, the idea of early detection and late detection is used in the project as well. The early detection is when the system responds to a gesture before it is actually over, i.e. zero or negative lag detection. The late detection is a compensation for assuring each gesture should be detected if the early detection does not be activated, which ensures a very high recall rate. Generally, the early detection is more confident than the late detection.

The pseudocode of classification strategy is described in Algorithm 1. For each window, the raw probabilities produced by the classifier are multiplied by the corresponding weight, and then the results are accumulated. From the generating list, the two highest values are selected. When the difference between the two values is greater than the threshold τ_{early}, then the early detection is activated; if the early detection is not triggered and the maximum value is greater than the threshold τ_{late}, the late detection is triggered. The τ_{early} and τ_{late} are set as 0.6, 0.2 respectively in the experiment.

Algorithm 1 Classification

Input: Incoming video frames.
Output: The specific hand gesture class.

Initialize $weigthprobs$ with [0, ... ,0]
for each window w_i **do**
  $probs_i \leftarrow classifier(clip_i)$
  $\alpha_i \leftarrow weight_i \times probs_i$
  $weigthprobs = weigthprobs + \alpha_i$
end for
$(max_1, max_2) = max[weigthprobs]_2$
if $(max_1 - max_2) \geq \tau_{early}$ **then**
  $state \leftarrow$ "$early\ detection$"
  return gesture with max_1
else if $max_1 \geq \tau_{late}$ **then**
  $state \leftarrow$ "$late\ detection$"
  return gesture with max_1
end if

IV. Experiment

All the models are trained from sratch on Quadro RTX 6000 processor, with 100 training epochs. The code is programmed in Python and Pytorch. The detailed setting for these models is described as follows. Cross entropy loss function and stochastic gradient descent optimizer are taken for all the models. For C3D model, the initial gradient learning rate is set as 10^{-3}, the momentum is 0.9 and the weight decay is 5×10^{-4}. In addition, the learning rate will divide by 10 per 10 epochs. For ResNeXt-101 model, I set the initial learning rate is 10^{-2}, the momentum is 0.9 and the weight decay is 1×10^{-3}. Besides, the learning rate will divide by 10 per 15 epochs.

A. Dataset

The Jester database (Materzynska et al., 2019) of hand gestures, has an about 150,000 examples. They are proposed to split into train, validation, test set using the ratio 8:1:1. In the dataset, there are totally 27 classes, each of which has around 4,000 video clips, except for the class 'Doing other things' with nearly 9,500 clips. Particularly, the 'Doing other things' class is the contrast category with the collection of various actions, such as stretching, yawing and playing with hair. For each video clip, its average duration is 3 seconds i.e. roughly 36 frames.

B. Detector

In order to train the detector, a subset is collected from the Jester database and split into 'Gesture' and 'No gesture' two classes. In details, all of the data in the 'No gesture' class in Jester dataset is first collcted as the new 'No gesture', and then randomly pick the same number of data from all of the other classes as the 'Gesture' class. In the end, there are 8,556 train data and 1,066 test data in the subset, in the ratio of about

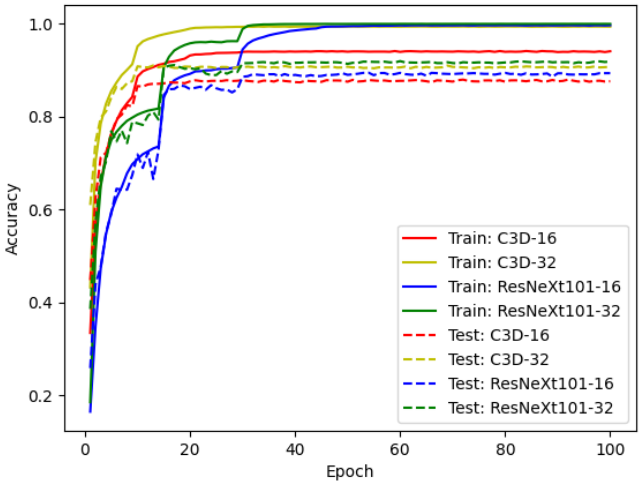

Fig. 6. The accuracy comparison of classifiers during the training.

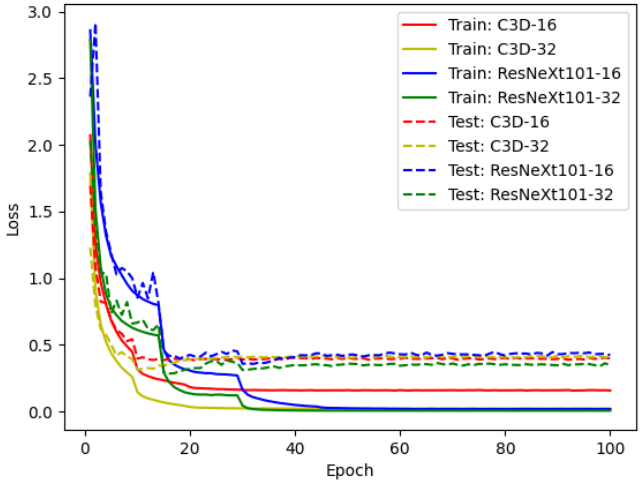

Fig. 7. The loss comparison of classifiers during the training.

8:1, which is exactly proposed by the official Jester dataset paper. Finally, the detector reaches a highest test accuracy of 98.22% on the subset.

C. Classifier

According to Fig. 6, C3D-16 performs the worst due to the lowest test accuracy after these models reach stable. Also, it is notable that the train accuracy difference between C3D-16 and C3D-32 is big, but that between ResNeXt101-16 and ResNeXt101-32 is almost zero after 30 epochs. It seems that training with a larger size of clip does not improve the performance of ResNeXt101, but the test accuracy difference is obvious between ResNeXt101-16 and ResNeXt101-32. Finally, the performances of all the models stabilize from around 40^{th} training epoch to the training end i.e. there is no overfitting, demonstrating that the penalty factor of the optimizer I set is reasonable.

TABLE III
THE HIGHEST TEST ACCURACY OF MODELS

Model	Input (clip)	The Highest Accuracy
C3D	16-frame	88.00%
	32-frame	91.03%
ResNeXt101	16-frame	89.47%
	32-frame	91.97%

TABLE IV
TEST ACCURACY COMPARISON ON JESTER DATABASE

Method	Accuracy
20BN's Jester System (Materzynska et al., 2019)	82.34%
C3D	91.03%
ResNeXt101	91.97%
TRN (Zhou et al., 2018)	94.78%
SSNet RGB resnet (Liu et al., 2018)	95.79%
Motion Fused Frames (Kopuklu et al., 2018)	96.28%
Deformable ResNeXt101(Shi et al., 2019)	96.60%

Moving onto Fig. 7, C3D generalize faster than ResNeXt101, since C3D reach stable only using nearly 20 epochs whereas ResNeXt101 spending about 30 epochs. This is probably because C3D has far more parameters than ResNeXt101, for example C3D-32 has over 0.5 times more parameters than ResNeXt101-32. Moreover, there are fluctuations in ResNeXt101 loss curve between 10^{th} and 20^{th} epoch, which is probably because for the initial learning rate (10^{-2}) of ResNeXt101 is higher than that of C3D (10^{-3}) and the decay step of the learning rate (15) is larger than that of C3D (10).

According to Table III, ResNeXt101 architecture with 32-frame input gains the highest test accuracy (91.97%) among these models. Although the accuracies of C3D-32 and ResNeXt101-32 are extremely close, ResNeXt101-32 has only 70.49 M parameters which is roughly a third less than the parameter numbers of C3D-32.

Table IV compares my classifier with other classification methods on Jester database. The accuracies of both my models are around 9% higher than that of the baseline model - 20BN's Jester System. Although my method does not perform better than others in terms of accuracy on Jester database, that is not the concern of my project. The focus of my project is on the real-world application. By the way, it is worth mentioning that some of their methods are too complex and unrealistic to be applied in real world, e.g. (Kopuklu et al., 2018).

Overall, ResNeXt101 with a 32-frame input clip can achieve the best performance among these groups, having the highest test accuracy and the smallest parameters.

D. Entire evaluation

For using the Levenshtein metric to evaluate the entire system, a small dataset is collected. Limited to conditions, the dataset only contains 20 videos, each of which has 4 gestures performed in 20 seconds. In other words, there are totally

TABLE V
THE AVERAGE DURATION FOR EACH STAGE

Detection	Classification	
	1 window	1.16 s
0.91 s	3 windows	3.26 s
	5 windows	5.33 s

TABLE VI
THE LEVENSHTEIN ACCURACY COMPARISON.

	C3D-16	C3D-32	ResNeXt101-16	ResNeXt101-32
1 window	27.50%	25%	37.50%	25%
3 windows	28.75%	18.75%	35%	25%
5 windows	23.75%	18.75%	33.75%	17.50%

80 gestures in 400 seconds video time used for evaluation. Besides, the entire evaluation experiment is made in Quad-Cord Intel Core i7 processor.

Table V reveals the average duration per round of detection and classification. It is clear that each round of detection costs no more than 1 second, whereas each round of classification lasts at least 1 second. Besides, the classification lasts longer with more sliding windows. These two phases are main components of the total response time of the system. Thus, when a gesture is performed, the system will cost at least 2 seconds to give the recognition results. However, it actually takes about 3 seconds to recognize that when using the 1-window classification method, because data capture, redundancy processing and so on take time as well.

From Table VI, astonishingly, both C3D and ResNeXt101 with 16-frame input generally performs better than those with 32-frame input in terms of Levenshtein accuracy. Although the classifiers with 32-frame input have higher classification accuracy, they make less contribution to the overall performance of the system. This is probably because larger size of input covers more distractions though it has a bigger probability of covering more useful information. In addition, using more classification windows seems to harm the overall performance of the system, which is probably because more windows not only enhance the exact gesture information but also strengthen the distractions. To sum up, using larger size of input data and more windows does not benefit the overall performance of the system, which is inconsistent with our intuitiveness.

Overall, the online system, consisting of a ResNet10 detector with 8 frame input and a ResNeXt101 classifier with 16 frames input, is proposed, which has a reaction time of less than three seconds and achieves the highest Levenshtein accuracy (37.5%) in my experiment.

V. CONCLUSION

In the paper, an online hand gesture recognition framework is presented, which can detect and classify gesture from real-time video stream. The proposed system only has a lag of no more than 3 seconds between performing a gesture and its classification, and can achieve a Levenshtein accuracy of 37.5% in self-collected dataset. Besides, a systematical evaluation is made from evaluating detector, classifier to evaluating the entire system. Notably, the Levenshtein metric is used, which can evaluate the misclassification, multiple detection and missing detection of the prediction sequence. In addition, several 3D CNN models well trained on Jester database are gained.

All in all, this project is a good attempt to bridge the gap between theory and practical application. Unlike a lot of previous work which just focus on improving the accuracy of the classifier, the project takes a holistic consideration on the whole system in terms of reaction time, recognition accuracy and computational cost.

VI. FUTURE WORK

The system can still be improved in many aspects. For example, the system is not sensitive to the temporal scale of the gesture, the three-seconds response time of the system cannot satisfy people's expectations, the recognition accuracy of the system is not high enough etc. Despite these flaws, it is possible for us to improve the system by applying more prominent models and finding more refined algorithm.

About the evaluation, it is imperative to find a unified evaluation criteria for online gesture recognition system. Although Levenshtein evaluation metric is used in my experiment, there is no an evaluation metric that is agreed by most researchers so far. In addition, the evaluation in this paper can be further in-depth, such as using larger data sets, adding more comparisons, etc.

Real-world deployment of the system is also a valuable direction. The system is possible to be deployed on vehicles, personal computers and even mobiles. Imagine using gestures to play virtual reality games with no extra devices and to control the computer without a mouse or keyboard, which would be so amazing.

ACKNOWLEDGMENT

I am gratefully acknowledge the support from my supervisor - Tijana Timotijevic.

REFERENCES

M. Abavisani, H. R. V. Joze, and V. M. Patel. Improving the performance of unimodal dynamic hand-gesture recognition with multimodal training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1165–1174, 2019.

K. S. Abhishek, L. C. F. Qubeley, and D. Ho. Glove-based hand gesture recognition sign language translator using capacitive touch sensor. In *2016 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC)*, pages 334–337. IEEE, 2016.

J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. 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.

D. M. Gavrila. The visual analysis of human movement: A survey. *Computer vision and image understanding*, 73(1): 82–98, 1999.

C. G. Haba, L. Breniuc, R. C. Ciobanu, and I. Tudosa. Development of a wireless glove based on rfid sensor. In *2018 International Conference on Applied and Theoretical Electricity (ICATE)*, pages 1–6. IEEE, 2018.

K. Hara, H. Kataoka, and Y. Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 3154–3160, 2017.

K. Hara, H. Kataoka, and Y. Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

P.-G. Jung, G. Lim, S. Kim, and K. Kong. A wearable gesture recognition device for detecting muscular activities based on air-pressure sensors. *IEEE Transactions on Industrial Informatics*, 11(2):485–494, 2015.

A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. 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.

O. Kopuklu, N. Kose, and G. Rigoll. Motion fused frames: Data level fusion strategy for hand gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2103–2111, 2018.

O. Köpüklü, A. Gunduz, N. Kose, and G. Rigoll. Real-time hand gesture detection and classification using convolutional neural networks. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, pages 1–8. IEEE, 2019.

V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

J. Liu, A. Shahroudy, G. Wang, L.-Y. Duan, and A. C. Kot. Ssnet: scale selection network for online 3d action prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8349–8358, 2018.

J. Materzynska, G. Berger, I. Bax, and R. Memisevic. The jester dataset: A large-scale video dataset of human gestures. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

P. Molchanov, S. Gupta, K. Kim, and J. Kautz. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–7, 2015a.

P. Molchanov, S. Gupta, K. Kim, and K. Pulli. Multi-sensor system for driver's hand-gesture recognition. In *2015 11th IEEE international conference and workshops on automatic face and gesture recognition (FG)*, volume 1, pages 1–8. IEEE, 2015b.

P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4207–4215, 2016.

E. Ohn-Bar and M. M. Trivedi. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. *IEEE transactions on intelligent transportation systems*, 15(6):2368–2377, 2014.

X. Shen, G. Hua, L. Williams, and Y. Wu. Dynamic hand gesture recognition: An exemplar-based approach from motion divergence fields. *Image and Vision Computing*, 30(3): 227–235, 2012.

L. Shi, Y. Zhang, J. Hu, J. Cheng, and H. Lu. Gesture recognition using spatiotemporal deformable convolutional representation. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1900–1904. IEEE, 2019.

K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

A. Tamrakar, S. Ali, Q. Yu, J. Liu, O. Javed, A. Divakaran, H. Cheng, and H. Sawhney. Evaluation of low-level features and their combinations for complex event detection in open source videos. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3681–3688. IEEE, 2012.

D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A robust and efficient video representation for action recognition. *International journal of computer vision*, 119(3):219–238, 2016a.

L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016b.

S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

Y. Zhang, C. Cao, J. Cheng, and H. Lu. Egogesture: a new dataset and benchmark for egocentric hand gesture recognition. *IEEE Transactions on Multimedia*, 20(5):1038–

1050, 2018.

B. Zhou, A. Andonian, A. Oliva, and A. Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 803–818, 2018.