Recognising Social Touch Gesture by Using Ensemble Classifier of Random Forest and Convolutional Neural Network

Dou Liu
s2352184
Computer Science, EEMCS

Qiuke Li
s2597403
Computer Science, EEMCS

Sathvik Guru Rao
s2431114
Computer Science, EEMCS

Yiran Wei
s2379198
Interaction Technolgy, EEMCS

ABSTRACT

In recent years, social touch gesture recognition has been considered as an important topic in touch morphology research, which can realize efficient and real touch human-computer interaction. This paper investigates on random forest, deep-learning methods, which are 2D CNN and 3D CNN, and an ensemble classifier combines the three models for the social touch gesture recognition on CoST dataset. What's more, we compared two different method to construct input data for CNN. As a result, 3D CNN outperformed random forest and 2D CNN on social touch gesture recognition problem with the accuracy of 60.67% on CoST dataset. And the weighted average ensemble classifier outperformed the other models implemented in this experiment with a recognition accuracy 62.27%.

1 INTRODUCTION

As a vital non-verbal language, touch is an interpersonal social interaction conveying social clues and communicating emotions [5]. In Human-Robot Interaction (HRI), recognition of social touch enables the artificial agents to understand the human's touch input, which is beneficial for designing an understandable and natural human-robot interaction[2]. And there is more attention for recognition of behaviors related to visual and auditory modalities than the recognition of social touch in the HRI field [15]. To spark the interest for recognition of touch behavior, social touch challenge was proposed in 2015 [10], initial four research groups classified gestures using the corpus of social touch (CoST) and the human-animal affective robot touch (HAART) dataset, and among their classifiers, random forest classifier of Ta et al.[14] performed better than the other classifiers [3, 6, 7]. And previously 3D-CNN showed good performance for detection of human actions[9].

In our project, the 3D Convolution Neural Network model is used for the classification of gestures. Continuous frames from the gestures are collected to form a 3D cube which is then used for the 3D CNN model. The research question that is answered in this paper is if the accuracy of the classification increases if the 3D CNN is used when compared to a 2D CNN. By the results of Merel et al[10], random forest has performed well in the classification of the gestures. A multimodal model is built to see if the classification accuracy increases when CNN, 3D CNN and random forest is combined.

We will show the state-of-art of social touch detection and introduce the dataset we worked on. Then we will explain how we train the model and how we combine the multi-classifier. Finally we will show the results and reflect on our method and results.

2 RELATED WORK

2.1 Research Context

A number of studies have used various classification methods to classify the CoST dataset according to the different numbers of features extracted from the original input data.

In the work of Zhou and Du [16], they applied three different deep learning approaches for the social touch gesture recognition on the Human-Animal Affective Robot Touch (HAART) database. The first two approaches are recurrent neural networks using long short-term memory(LSTM) with different features: one is handmade using geometric moments and another is extracted by CNN. However, the performance of LSTM based methods is not satisfactory. Then, the authors used 3D-CNN and this approach obtained 76.1% accuracy on HAART, outperforming the previous best submitted result significantly. In their model, they set 4 convolutional layers and they found that 3D-CNN performs best when the configuration of feature maps for 4 convolution layers is set to 16-32-64-128. The kernel sizes in every convolutional layer were all set to be 3×3×3. But this approach is not applied on the CoST dataset.

Hughes et al. [7] first explored the social touch recognition challenge using deep learning approaches like CNN and RNN. For the CoST and HAART data sets, the convolutional neural network classification ratios used are 42.34% and 56.10%, respectively. The classification rates of convolutional-recurrent neural networks (CNN-RNN) are 52.86% and 61.35%, respectively. For the encoder-regression neural network, the classification rates are 33.52% and 61.35%, respectively.

What's more, Albawi et al. [1] proposed a nearly real time classification system for social touch gesture using deep neural networks on the CoST dataset. Compared to the previous method, this approach has two benefits. First, it does not need to preprocess the data or do the feature extraction, and can be applied as end-to-end. Second, the classification operation begins after the minimum number of frames (frame length 85) are received, which makes the real time classification possible.

2.2 CoST Dataset

To collect the social touch gesture for CoST [1], the gesture was performed on the mannequin arm wrapped with the pressure sensor of the 8x8 grid, sampled at 135Hz in 64 channels ranging from 0 to 1023.

The duration of each gesture is variable without restriction either for performing or repeating each gesture, meaning the amount of frame for each gesture is different. There were 31 subjects asked to perform 14 social gestures including grab, hit, massage, pat, pinch, poke, press, rub, scratch, slap, stroke, squeeze, tap, and, tickle [1]. They are required to perform gestures in three variations, gentle, normal, and rough respectively. Each gesture with particular variants was repeated six times. In total, 252 gestures were performed by each subject, while a few samples were removed because of data loss, eventually the CoST contains 7,805 gestures from the subjects. Touch gesture type, gesture variation, and subject number were labeled in the dataset.

3 METHODOLOGY

Firstly, we train the three models independently, and tune the parameters for Random Forest and tune the hyperparameters for two CNN models. Then we make ensemble models according to the three classifier's performance.

3.1 Random Forest Classifier

Random Forests is an ensemble learning method consisting of a large number of individual decision trees. Each individual tree generates a class prediction, and the class with the most votes becomes Random Forest model's prediction. Random forests have been used for their ability to handle large sets of features and to include irrelevant features.

Based on the previous studies[14], we selected and modified several features for feature extraction. They are divided into two groups, group 1 is global feature including, consisting 14 features and representing overall statistics of the gestures. Length of touch. It is the number of frames representing the duration of each touch.

- Average pressure on 64 channels over all frames.
- Maximum value of pressure on 64 channels over all frames.
- Percentage of no-signal frames. We define the no signal frame as its average pressure on 64 channels below a threshold T (T=50), then obtain the percentage of no-signal frames for one touch.
- Variations around a specific value c. This feature was improved by the zero-crossing rate in speech application. The mean pressure value of 64 channels for each frame is considered as one dimensional temporal signal. Then we calculated the number of crossings around the central value (c) of one touch, and the average slope of each crossing (slope = difference between two values who crosses c/ sum of two values who crosses c). And we took central value c as the 35th, 50th, 65th, 80th and 95th-percentile of the maximal value. Therefore, each touch has 10 sub-features among this feature.

Group 2 is channel based feature extracted from each channel. Basically, it focuses on the spatial relationship between different channels ignoring the temporal relationship of frames. Each touch has 64 features in this group, in total 128 features for channel based feature set. Overall, we get 144 features to train the random forest model.

- Average pressure of each channel over all frames
- Percentage of pressure value over threshold T on each channel. We set T as 90-percentile of all pressure values in one touch, and for each channel we look for the pressure value over T to get the percentage among all frames.

After feature extraction, we ran Random Forest model from Scikit-learn library with standard parameters[4], and applying "GridSearchCV" to tune the parameters including "e_estimators", "max_depth", "max_features", "min_samples_leaf" and "min_samples_split".

3.2 Input structure of CNN

Unlike the Random Forest, CNN model requires to specify the input structure. The sensor size in CoST is 8 times 8 and the gesture maintained for N frames. The frame length in the cost is variable for each sample. In order to implement CNN effectively, the input size must be fixed. There are mainly two processes to get a fixed frame length N as Albawi et al.[1] proposed. One is to assume the input size according to the sample with the maximum frame length of all the sample, and zero padding is used for the sample with short frame length. The other method is to divide the samples according to a fixed length, that is, each sample is divided into subsamples. When the number of frames is not divisible by the given length of the subsample, the remainder of the subsample whose number of frames is less than the given number of frames will be padded by zeros, as shown in Figure 1. We implemented both input structure methods to conduct neural network experiments, and compare the difference of results between them.

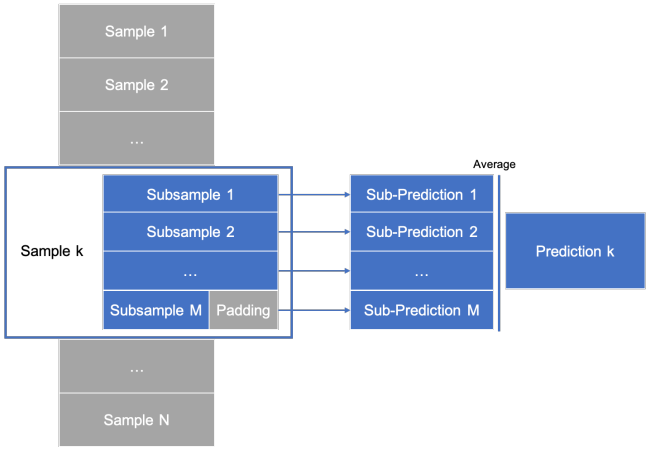

Figure 1: Framework of Subsample

3.3 2D-CNN Classifier

3.3.1 Covolutional Neural Network. Neural Networks is one of most used algorithms. It replicates the operations of a human brain by using interconnected neurons. Artificial neural networks(ANN) consist of many layers which are interconnected through neurons. An input layer takes the input, the hidden layer carries out the operations and the output layer gives the output. This stacking of multiple layers like input layer, hidden layer and output layer is popularly known as deep learning.

A convolution neural network(CNN) [12] is similar to ANN but is most suitable for complex computations like processing grid like topology data like image data. The neurons in the layers are

organized based on the dimensions of the data like height and width. Another difference from ANN is that the layers are connected to a small region of neurons within the layer. The architecture of CNN has three main layers : convolutional layer, pooling layer and fully connected layer.

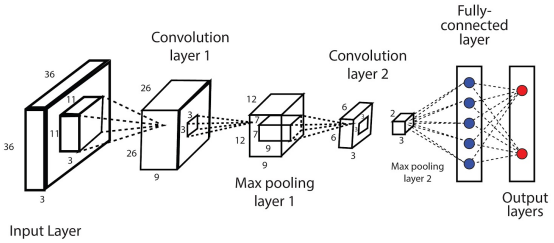

Figure 2: Convolutional Neural Network

3.3.2 Convolution Layer. One of the main layers of the CNN model is the convolutional layer which is a dot product of 2 matrices. The receptive field of input data and the kernel matrix. This layer plays a major role in the networks computations. The height and width of the kernel is smaller than the input size but has the same depth size. The depth is the channels of the input. As other neural networks, CNN also has the forward pass and back propagation. In forward pass, the kernel slides across the height and width of the input and the dot product produces the activation map. The sliding window size is known as strides.

The advantage of using convolutional neural networks is constant filter parameters. As the kernel slides over the input the same weight is applied for the convolution operation which reduces the parameters. Using a small size kernel compared to the size of input, helps to detect the meaningful information which helps in storing less parameters and reducing the memory of the model. While performing the dot product multiplication, information in the edges of the input can be lost or considered less to gain information. To overcome this, zero padding is performed which adds zeros in the edges and helps to have more convolution layers.

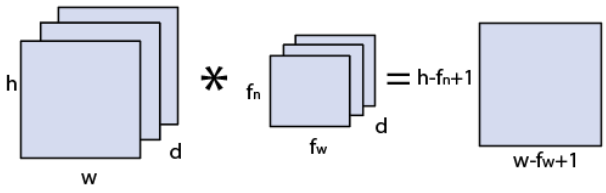

Figure 3: Convolution operation

3.3.3 Pooling Layer. The pooling layer is used to reduce the dimensions of the input which further reduces the parameters.Pooling layer operates on each feature map independently. This layer replaces the output of the previous layer by taking the summary statistics like max pooling or average pooling. The size of the pooling filter is given which is a matrix which slides over the output of the previous layer and for each window it takes the average or the maximum which helps in reducing the parameters. Max pooling is one of the most commonly used pooling methods which selects the maximum value inside that filter. This will pick the most activated pixel and pass it as the output to the next layer which in turn reduces overfitting.

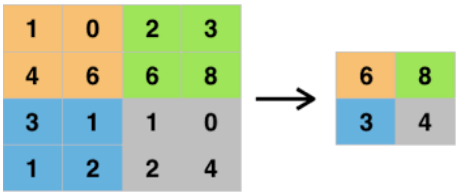

Figure 4: Convolution operation

3.3.4 Fully Connected Layer. This layer is the same as the layers in ANN where all the neurons in the preceding layer are connected to the neurons in succeeding layers. Basic multiplication is carried out to map the representation between input and output. The main purpose of the layer is to take the output from convolution and pooling layers to classify the labels. The output is flattened into a single layer which shows the probability of the label to which it belongs.

3.3.5 Architecture of CNN. Convolution neural network models consist of many layers. The input data is processed with convolution operation in the convolution layer using the kernels. In the model, 3 convolution layers are used. The parameters for this layer is the filter size and strides. A filter of 3 x 3 is used with stride 1. Convolution is a linear operation so a non linear activation function is used after every convolution layer to get non-linearity. ReLU activation function is used in the model which is max(0,k).

After the 3 convolution layers, the output is given to the maxpool layer which reduces the parameter and followed by a flatten layer which converts the data into 1D array which is then inputted to the fully connected layer. Two fully connected layers are used in the model which also has an activation function, ReLU function after the first fully connected layer. Before the output is given to the last, output fully connected layer, a dropout is used to drop some of the neurons to prevent the model from overfitting. 20% dropout was used in the model. The final fully connected layer has the softmax function to get the probability of the labels as the output. The label which has the highest probability is considered as the label for that input.

The CNN follows back propagation which calculates the error in each layer and uses an error function which helps in reducing. The goal of the error function is to reduce the error by back propagating. Negative Log-Likelihood function is used in the model. Pytorch has a function called cross entropy which is the combination of

Table 1: Parameters of 2D CNN architecture

Layer number	Element	Parameter	Value
1	Convolutional filter	Input channels	100
		size	3×3
		Stride	1
		Padding	1
2	Convolutional filter	Input channels	64
		size	3×3
		Stride	1
		Padding	1
3	Convolutional filter	Input channels	56
		size	3×3
		Stride	1
		Padding	1
	Max pooling	Size	3×3
		Padding	1
4	Fully connected	Input to layer	16×33×4×4
		Output from layer	32
5	Fully connected	Input to layer	32
		Output from layer	16
6	Output	Output units	14

negative log likelihood function and the softmax function. So, this criterion is used in the model instead of using them explicitly.

The main objective of training the model is to get the best accuracy, to enhance this, an optimizer is used which helps in getting a better performance. Adam optimizer[11]] is used in the model which takes the models parameter and the learning rate as the parameter. Adam optimizer is well suited for models which are used on large data which uses the stochastic gradient with momentum. Learning rate tells how the model adapts to the error in backpropagation.

3.3.6 Implementation. The input data was of 8 x 8 x N dimension where N is the frame length. The frame length differs for each sample. To use this data in a model, a constant frame length was chosen. First, the experiment was carried out with the minimum and maximum frame length. 80 frame length was chosen, the training was fast but the accuracy of the model was very low. Maximum frame length of 1750 was used, the training time and the memory was very large but comparatively the accuracy increased. The model was built using pytorch and google colab was used as the interface, as GPU was required for the model training. Number of input channels is equal to the frame length. Number of outputs is equal to the number of labels.

For comparing the accuracy's of the Random Forest, 2DCNN and 3DCNN which is discussed in the next section,the same number of frame length was used to build the model and the sub sample data was used to train and test the model.The architecture of this model is given in Table 1.

The CNN model was built using 3 convolution layers, 1 max pool layer, and 2 fully connected layers with cross entropy criteria. The summary of the model is shown in the below Table 1.

Table 2: Parameters of 3D CNN architecture

Layer number	Element	Parameter	Value
1	Convolutional	Input channels	64×6×6
		size	3×3×3
		Stride	1
		Padding	1
	Max pooling	Size	3×1×1
		Padding	1
2	Fully connected	Input to layer	16×33×8×8
		Output from layer	1024
3	Fully connected	Input to layer	1024
		Output from layer	128
4	Fully connected	Input to layer	128
		Output from layer	14
5	Softmax	Output units	14

3.4 3D-CNN Classifier

Ji et al.[8] proposed to deal with problem of human motion detection in video is proposed, and excellent performance is achieved. In the 2D CNN, the 2D feature map is convolved, and only the feature information in the spatial dimension can be captured. Unlike the 2D CNN, a 3d CNN net can capture the motion information encoded in multiple contiguous frames to find the features from temporal dimensions. For the overall architecture of 3D CNNs, there are 3d convolutional layers, max-pooling layers, fully connected layers, and output layer, which are similar to the Section 3.3. And formally, the value at position (x, y, z) on the jth feature map in the ith layer is given by

$$v_{ij}^{xyz} = tanh\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}\sum_{e=0}^{E_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}\right) \quad (1)$$

where R_i is the size of the 3D kernel, w_{ijm}^{pqe} is the (p, q, r)th value of the kernel connected to the mth feature map in the previous layer.

When dealing with social gesture recognition challenge using 3D CNN, the gesture sequence is input, and we stack the contiguous input frames to get a cube, the height and width are both 8, and these 64 features represent the 64 channels, and the depth of the cube represents the contiguous frames. Then we use a 3D kernel to capture the motion information. We have one 3D convolution layer with kernel size 3×3×3 and a 3D max-pooling layer with size 3×1×1. And three fully connected layers, two dropout layers, and one softmax layer. The current network is shown in the Table 2.

3.5 Ensemble Model

To make ensemble model, firstly we train the models independently to get N classifier, and then make the ensemble by two solutions, as shown in Figure 5. In solution 1, the result is obtained by averaging the responses of the N classifier by majority vote, which is formulated as:

$$y = \frac{1}{N}\sum_{i=1}^{N} y_i. \quad (2)$$

For solution 2, we will assign a set of weights for classifiers based on their own performance, in order to take a weighted average of their estimates, which is formulated as:

$$y = \frac{1}{N}\sum_{i=1}^{N} w_i y_i (\sum_{i=1}^{N} w_i = 1). \quad (3)$$

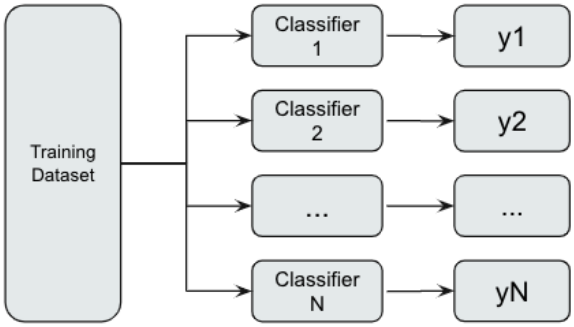

Figure 5: Making ensemble model process

4 RESULTS

4.1 Results of random forest

80% of the data was used for training and then the model was tested with testing data. We ran a Random Forest model from the Scikit-learn library with default parameters and got 60.15% accuracy, then we tested tuned models and got the same overall performance 60.15% , although these two models showed slight differences for specific gestures Figure 10.

Among all parameters, "e_estimators" influenced performance the most. We apply the model of Scikit-learn library with standard parameters [4] to make the ensemble classifier afterwards. Besides, we estimate the effectiveness of the feature set. With group 2 channel based features, the accuracy boosted around 17% Table 4, and we checked the feature importance. Among all the features, channel based feature has higher importance than global feature Figure 13, and the average pressure of each channel over all frames feature matters most in all features.

4.2 Results of 2D-CNN

We trained the model by utilizing 80% of the data with 30 epochs and then the model was tested with testing data. The accuracy of the model with testing data was 42% for the sub sample data with 100 frames as shown in Figure 11. Some of the gestures were predicted with good accuracy but some have a very low accuracy. Grab and pinch have been classified well but rub and scratch have not.

An experiment was done out of curiosity about how the variants: gentle, normal and rough pressure affects the classification of gestures. The same method using maximum frame length and the model for that was used to train the data for different variants. The variants were first grouped and for each group the training and testing was performed. From the results we can see that the normal pressure variant has a better performance compared to others. This can be seen in Figure 13

4.3 Results of 3D-CNN

For the 3D CNN experiments, we used Pytorch[13] as the deep-learning framework. Furthermore, the iterations number is set to 30000 and batch size to 128. The learning function is simple stochastic gradient descent (SGB), and the learning rate is selected as 0.001.

First, we implemented two different methods, one sets the frame length to the maximum, and another splits the samples with a fixed length to construct our input data. Moreover, a grid search is performed to get the optimal frame length by running the experiments for the frame length from 10 to 200 with every ten intervals. The result is shown in 6. Generally speaking, the increasing frame length can improve the performance of the classification accuracy. When the frame length is less than 40, the accuracy is low, under 50% or even worse. And the accuracy reaches the maximum value at 100 frames and has little change when increasing the frame length. On the other hand, when we applied the maximum frame length as the input size, the average accuracy is 60.67%, which is lower than the splitting samples method. In conclusion, we set the input dimension as 100 frames for our CNN.

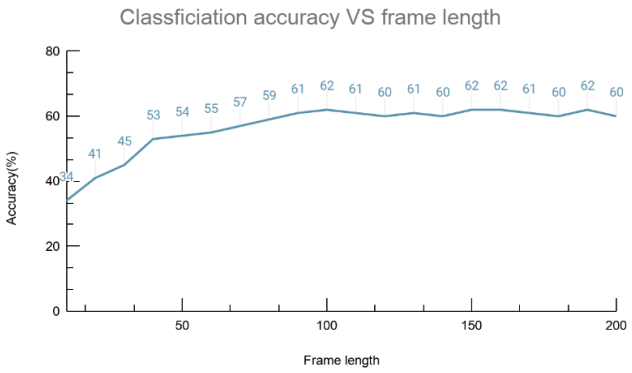

Figure 6: Evaluation of the subsample splitting method. Performance is improved with the increase in the frame length of the data.

One thing should be noted here: using the maximum frame length method to construct the input data makes the training process spend more time than another input structure because the maximum frame length provides CNN with a considerable input size and is computationally expensive. As a result, considering the efficiency, the maximum frame length is not used here.

We've gotten a 62% accuracy on 3D CNN as shown in Figure 12, which is better than the performance of most of the previous models on CoST data. But it should be noted that using maximum frame length provides CNN with a huge input size and is computationally expensive.

4.4 Results of ensemble classifiers

For the ensemble classifiers, two types of ensemble structure are implemented in our experiment. First one is simple average ensemble, which sums up the prediction of all three classifiers above and choose the highest probability as the gesture type predicted. And the second is a weighted average ensemble classifier, which give different weight to each model. By using grid search, the weights are set to $(0.38, 0.12, 0.5)$ here.

Table 3: Performance of four model

Classifier	Accuracy(%)
Random Forest	57.85
2D-CNN	42.41
3D-CNN	60.67
Ensemble (Simple Avg)	56.31
Ensemble (Weighted Avg)	**62.27**

As shown in Table 3, the simple average ensemble classifier has accuracy of 56.31%, which is slightly lower than Random Forest model. Because there is a certain difference in the accuracy of the three models that make up ensemble, especially the 2D CNN has a lower accuracy than the other two models. Thus for the simple average model, it is reasonable to have lower accuracy. And the weighted average ensemble classifier's accuracy is better than 3D CNN, which reaches 62.27. Thus, with a proper weights found by grid search, the weighted average ensemble classifier can achieve better performance than the previous model we have.

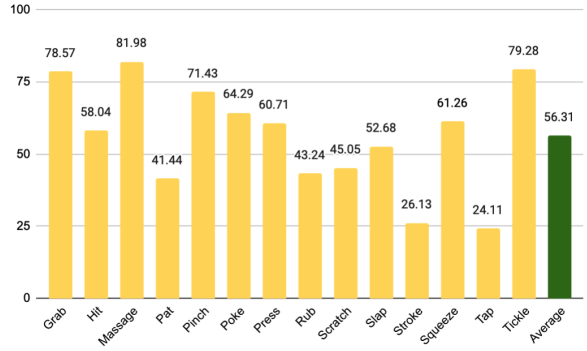

Figure 7: Results for Simple Average Ensemble Model

5 DISCUSSIONS

For the feature of random forest, we did not take the sequence feature into consideration, for example applying Fast Fourier transform to get the sequence of average pressure to get the frequency value as the feature, it helped improve the accuracy for recognition in previous study [14].

Since the data was collected by 31 subjects and each person performed with 3 variants. A general notion is that some gestures can be misclassified based on the pressure, as the action of those gestures is similar but only depends on the pressure rate. So comparing the three variants accuracy of these classifiers can be conducted in the further study.

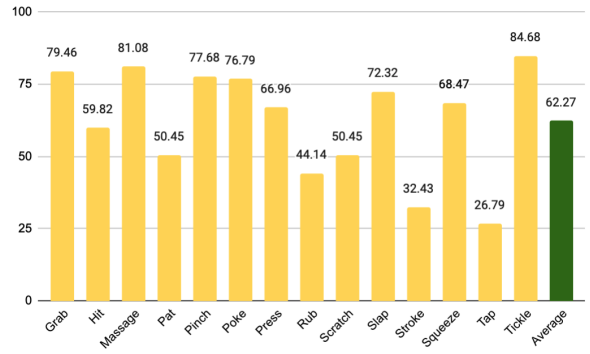

Figure 8: Results for Weighted Average Ensemble Model

For 3D CNN, the performance when using maximum frame length padding is slightly worse than subsampling. That is why we choose the subsampling method to construct the input data. Another important reason is that for 3D CNN, the parameters are already considerable; if we use the maximum length padding for all the sample, then the input size would be even bigger, which will make the training computation expensive.

Besides, according to the average results of all type of gestures, 3D-CNN performs relatively better than the other classifier, but "stroke" and "tap" gestures show lower accuracy than other classifiers. Random Forest also has good performance but with lower accuracy for "scratch" gesture.

For the two ensemble model, they combine both strength and weakness from individual classifier but got relatively good performance. While "stroke" and "tap" gestures have lower accuracy in comparison than other type of gestures as the 2D-CNN and 3D-CNN have quite low accuracy for these two gestures. They should be improved in the further study. Weighted ensemble model give a relatively even accuracy for the other type of gestures.

6 CONCLUSIONS

Among all the three classifiers we implemented, it is clear to see 3D CNN outperformed the other two classifiers on the CoST dataset. And by evaluating two different input structure, we know that for 3D CNN, using the subsamples as input can gain higher accuracy and efficiency on social gesture recognition compared to padding all samples to the maximum frame length. Such a result also demonstrates that 3D CNN is good at dealing with video-liked 3D data and capture the motion information from contiguous frames so as to find the features from temporal dimensions.

For the ensemble part, average accuracy of weighted average ensemble model perform better than average ensemble model, and also better than the other three classifiers, but overall, random forest perform evenly for 14 type of gestures.

REFERENCES

[1] Saad Albawi, Oguz Bayat, Saad Al-Azawi, and Osman N. Uçan. 2018. Social Touch Gesture Recognition Using Convolutional Neural Network. *Computational Intelligence and Neuroscience* 2018 (2018).

[2] Brenna D. Argall and Aude G. Billard. 2010. A survey of Tactile Human–Robot Interactions. *Robotics and Autonomous Systems* 58, 10 (2010), 1159 – 1176. https://doi.org/10.1016/j.robot.2010.07.002

[3] Tugce Balli Altuglu and Kerem Altun. 2015. Recognizing Touch Gestures for Social Human-Robot Interaction. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction* (Seattle, Washington, USA) *(ICMI '15)*. Association for Computing Machinery, New York, NY, USA, 407–413. https://doi.org/10.1145/2818346.2830600

[4] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

[5] Alberto Gallace and Charles Spence. 2010. The science of interpersonal touch: An overview. *Neuroscience Biobehavioral Reviews* 34, 2 (2010), 246 – 259. https://doi.org/10.1016/j.neubiorev.2008.10.004 Touch, Temperature, Pain/Itch and Pleasure.

[6] Yona Falinie A. Gaus, Temitayo Olugbade, Asim Jan, Rui Qin, Jingxin Liu, Fan Zhang, Hongying Meng, and Nadia Bianchi-Berthouze. 2015. Social Touch Gesture Recognition Using Random Forest and Boosting on Distinct Feature Sets. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction* (Seattle, Washington, USA) *(ICMI '15)*. Association for Computing Machinery, New York, NY, USA, 399–406. https://doi.org/10.1145/2818346.2830599

[7] Dana Hughes, Nicholas Farrow, Halley Profita, and Nikolaus Correll. 2015. Detecting and Identifying Tactile Gestures Using Deep Autoencoders, Geometric Moments and Gesture Level Features. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction* (Seattle, Washington, USA) *(ICMI '15)*. Association for Computing Machinery, New York, NY, USA, 415–422. https://doi.org/10.1145/2818346.2830601

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

[9] S. Ji, W. Xu, M. Yang, and K. Yu. 2013. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 1 (2013), 221–231. https://doi.org/10.1109/TPAMI.2012.59

[10] Merel Madeleine Jung, Xi Laura Cang, Mannes Poel, and Karon E. MacLean. 2015. Touch Challenge '15: Recognizing Social Touch Gestures. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, ICMI 2015*. Association for Computing Machinery (ACM), United States, 387 –390. https://doi.org/10.1145/2818346.2829993 eemcs-eprint-26507 ; null ; Conference date: 09-11-2015 Through 13-11-2015.

[11] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[12] Keiron O'Shea and Ryan Nash. 2015. An Introduction to Convolutional Neural Networks. *ArXiv e-prints* (11 2015).

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703* (2019).

[14] Viet-Cuong Ta, Wafa Johal, Maxime Portaz, Eric Castelli, and Dominique Vaufreydaz. 2015. The Grenoble System for the Social Touch Challenge at ICMI 2015. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction* (Seattle, Washington, USA) *(ICMI '15)*. Association for Computing Machinery, New York, NY, USA, 391–398. https://doi.org/10.1145/2818346.2830598

[15] Z. Zeng, M. Pantic, G. I. Roisman, and T. S. Huang. 2009. A Survey of Affect Recognition Methods: Audio, Visual, and Spontaneous Expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 1 (2009), 39–58. https://doi.org/10.1109/TPAMI.2008.52

[16] Nan Zhou and Jun Du. 2016. Recognition of Social Touch Gestures Using 3D Convolutional Neural Networks, Vol. 662. 164–173. https://doi.org/10.1007/978-981-10-3002-4_14

A APPENDIX

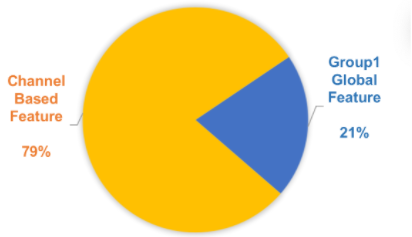

Figure 9: Importance of two groups of features

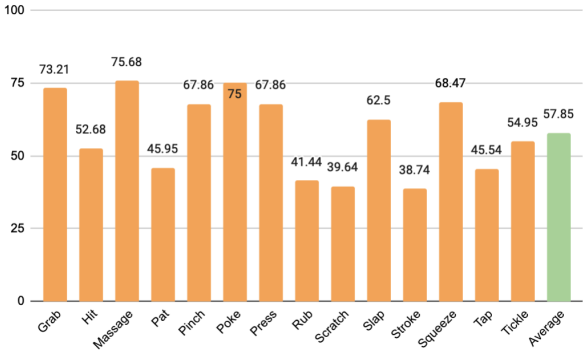

Figure 10: Results for Random Forest

Table 4: Performance of four mode

Classifier	Accuracy(%)
Random Forest	57.85
2D-CNN	42.41
3D-CN	60.67
Ensemble(Simple Avg)	56.31
Ensemble(Weighted Avg)	**62.27**

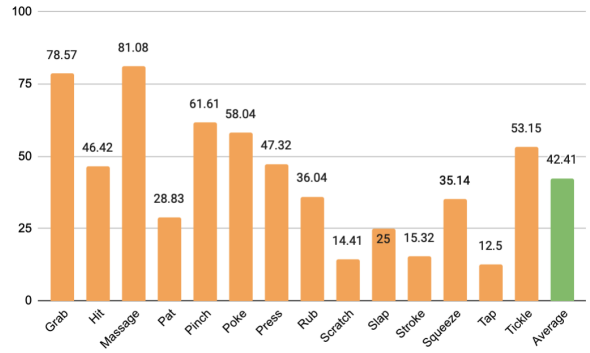

Figure 11: Results for 2D CNN

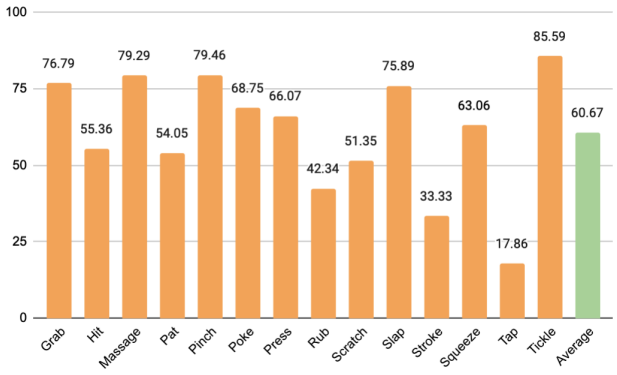

Figure 12: Result of 3D-CNN classifier

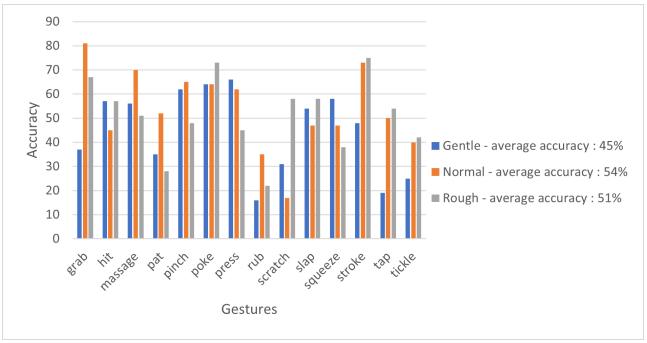

Figure 13: Accuracy for three variants for 2D-CNN