Static and Dynamic Hand Gesture Recognition Using CNN Models

Keyi Wang*
19835 19th Ave. NW, Shoreline, Washington, USA.

* Corresponding author. Tel.: 2065023766; email: christinawang22@outlook.com
Manuscript submitted January 30, 2021; accepted April 12, 2021.
doi: 10.17706/ijbbb.2021.11.3.65-73

Abstract: Similar to the touchscreen, hand gesture based human computer interaction (HCI) is a technology that could allow people to perform a variety of tasks faster and more conveniently. This paper proposes a training method of an imaged-based hand gesture image and video clip recognition system using Convolutional Neural Networks (CNN). A dataset containing images of 6 different static hand gestures is used to train a 2D CNN model. ~98% accuracy is achieved. Furthermore, a 3D CNN model is trained on a dataset containing video clips of 4 dynamic hand gestures resulting in ~83% accuracy. This research demonstrates that a Cozmo robot loaded with pre-trained models is able to recognize static and dynamic hand gestures.

Keywords: Deep learning, hand gesture recognition, convolutional neural network, human computer interaction (HCI).

1. Introduction

Hand gesture recognition is an important and inspiring field of research as it has numerous applicarions, such as sign language recognition [1], robot control [2], home automation [3], lie detection [4], virtual environments [5], and personal electronic devices [6]. For example, Hand gesture recognition technology allows for the development of sign language translations systems that can turn sign language into spoken languages, and vice versa. This research develops a hand gesture recognition system based on two types of CNNs [7]-[10] to identify two types of human hand gestures: static and dynamic gestures. A 2D CNN was trained with images of six static hand gestures. A static hand gestures (Fig. 1) is a single image displaying a still and motionless hand pose or position. A 3D CNN was trained with video clips of four dynamic hand gestures. A dynamic hand gesture (Fig. 2) is a hand movement, represented by a video clip, or a sequence of images.

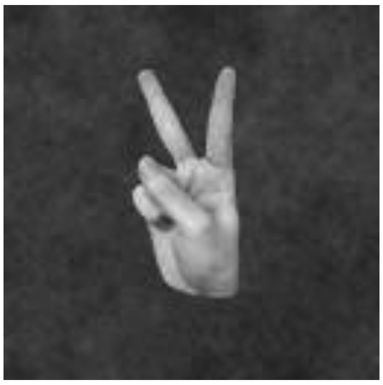


Fig. 1. static hand gesture.

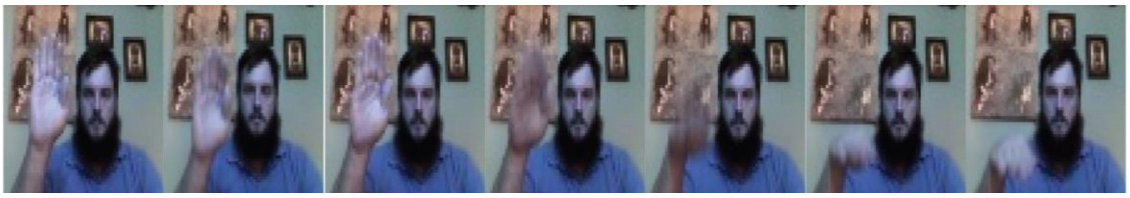


Fig. 2. Dynamic hand gesture.

This research leveraged the Anki Cozmo robot [11] (Fig. 3), which is a small robot that has a well-documented SDK, as a platform to test each CNN's ability to recognize static and dynamic hand gestures using the trained models. A black and white camera is embedded in the movable head of the robot, allowing it to capture images and frames of hand gestures.

Fig. 3. Anki Cozmo robot.

2. Neural Network Model

The hand gesture recognition system created in this research uses a Convolutional Neural Network, which has proven to be effective at image classification and identification. The network consists of three different types of layers: convolutional layers, pooling layers, and fully connected layers. In this study, each layer in the 2D convolutional neural network is 2 dimensional with one channel since the input images used were grayscale images. Once an input is received, the neural network transforms it through hidden layers, which consist of fully connected neurons that have learnable weights and biases. The convolutional layers are used to extract features from input images by sliding filters, or kernels, over the original images to compute output images, or feature maps, of the convolutional layer. The pooling layer then reduces the dimensions of each feature map, decreasing the number of parameters and computations in the network, hence avoiding the occurrence of overfitting. Fully Connected layers make up the last few layers in the network and primarily serve to classify the outputs of pooling layers into classes. In this study, two different Neural Network structures were used to recognize two different kinds of data: images (static gestures) and video clips (dynamic gestures). Details of each structure, layer, operation, and block are presented in this section.

2.1. Static Hand Gesture Recognition Network Structure

The structure of the neural network model used to recognize static hand gestures is a 2D convolutional network. The model contains four convolutional layers, four max pooling layers, followed by dropout. A visual representation of the network structure is shown in Fig. 4. The output, kernel, and filter size of each layer are shown in Table 1.

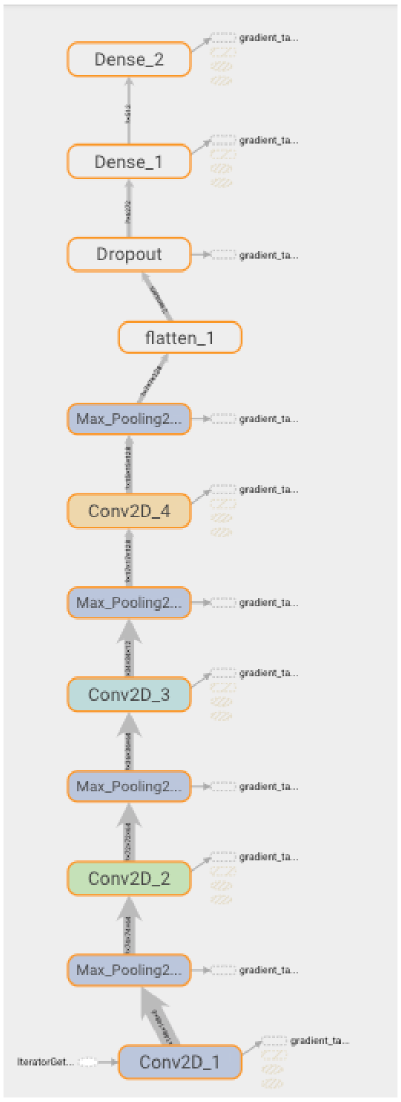


Fig. 4. static image recognition network structure.

Table 1. Network Architecture I

Layer/Operation	Parameters	Output Size
Inputs	-	150, 150, 1
Convolutional Layer	3x3 kernel	148, 148, 64
Max Pooling	2x2 filter	74, 74, 64
Convolutional Layer	3x3 kernel	72, 72, 64
Max Pooling	2x2 filter	36, 36, 64
Convolutional Layer	3x3 kernel	34, 34, 128
Max Pooling	2x2 filter	17, 17, 128
Convolutional Layer	3x3 kernel	15, 15, 128
Max Pooling	2x2 filter	7, 7, 128
dropout	0.5	6272
Fully Connected Layer	ReLU activation	512
Fully Connected Layer 1	Softmax activation	6

2.2. 2D Convolutional Layers

The convolutional layers are responsible for most of the computational tasks in the network as well as extracting features from the input images. This research primarily focuses on grayscale images with an intensity of 0 to 255, represented by two-dimensional matrixes. In a convolutional layer, each input image is batch normalized, then convolved; a bias as well as an activation function is applied to produce the outputs of the convolutional layer.

Features of the input image are extracted with a kernel, a filter that modifies the value of a pixel based on the values of neighboring pixels. The kernel slides across the original image, computing a feature map, or the output image of the convolutional layer.

Each CNN layer uses ReLU (Rectified Linear Unit) as activation function. Dropout with probability of 0.5 is used to avoid the appearance of over fitting. The Adam Optimizer is used with epsilon of 0.1.

2.3. Fully Connected Layers

Fully connected layers are an essential part of Convolutional Neural Networks (CNNs) as they form the last few layers in the network. Each input from one layer is connected to an activation unit of the next layer. For a layer with m inputs and n outputs, there are m weight parameters and 1 bias parameter for each of the n outputs, and therefore, the entire layer has m x n weights and n biases. For ease of calculation, the weights are stored as an m x n matrix (i.e. 2-dimensional array), and the biases are stored as a 1 x n vector. To acquire the outputs, the following calculation is performed: g(Wx+b). W is the weight matrix with dimensions [m,n] where m is the number of neurons in the previous layer and n is the number of neurons in the current layer; X is the input vector with dimension [m,1]; b is the bias vector with the dimensions [m,1]; g is the activation function, which in this study is ReLU.

2.4. Dynamic Gesture Recognition Network Structure

Unlike 2D convolutional layers, 3D Convolutional layers consists of three dimensions: width, height, and depth. The model contains three blocks of each which contains a 3D convolutional layer followed by batch normalization, max pooling and dropout (Table 2). Two dense layers are used for classification. Batch normalization and dropout with probability of 0.25 are used. Adam is used for optimizer and categorical cross entropy is used for loss function. Layers of the network are shown in Table 2. A visual representation of the network structure is shown in Fig. 5.

Table 2. Network Architecture II

Layer/Operation	Parameters	Output Size
Inputs	-	30,64,64,1
Convolutional Layer	5x5x5 kernel	3999, 30, 64, 64, 32
Batch Normalization		3999, 30, 64, 64, 32
Max Pooling	3x3x3 filter	3999, 10, 22, 22, 32
Dropout	0.25	3999, 10, 22, 22, 32
Convolutional Layer	5x5x5 kernel	3999, 10, 22, 22, 64
Batch Normalization		3999, 10, 22, 22, 64
Max Pooling	3x3x3 filter	3999, 4, 8, 8, 64
Dropout	0.25	3999, 4, 8, 8, 64
Convolutional Layer	5x5x5 kernel	3999, 4, 8, 8, 128
Batch Normalization		3999, 4, 8, 8, 128
Max Pooling	3x3x3 filter	3999, 2, 3, 3, 64
Dropout	0.25	3999, 2, 3, 3, 64
Flatten		3999,2304
Fully Connected Layer	ReLU activation	3999,128
Fully Connected Layer 1	Softmax activation	4

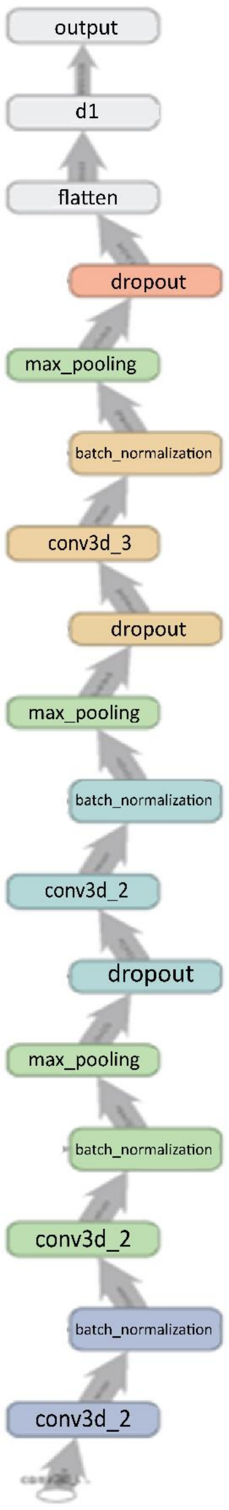


Fig. 5. Video clip recognition network structure.

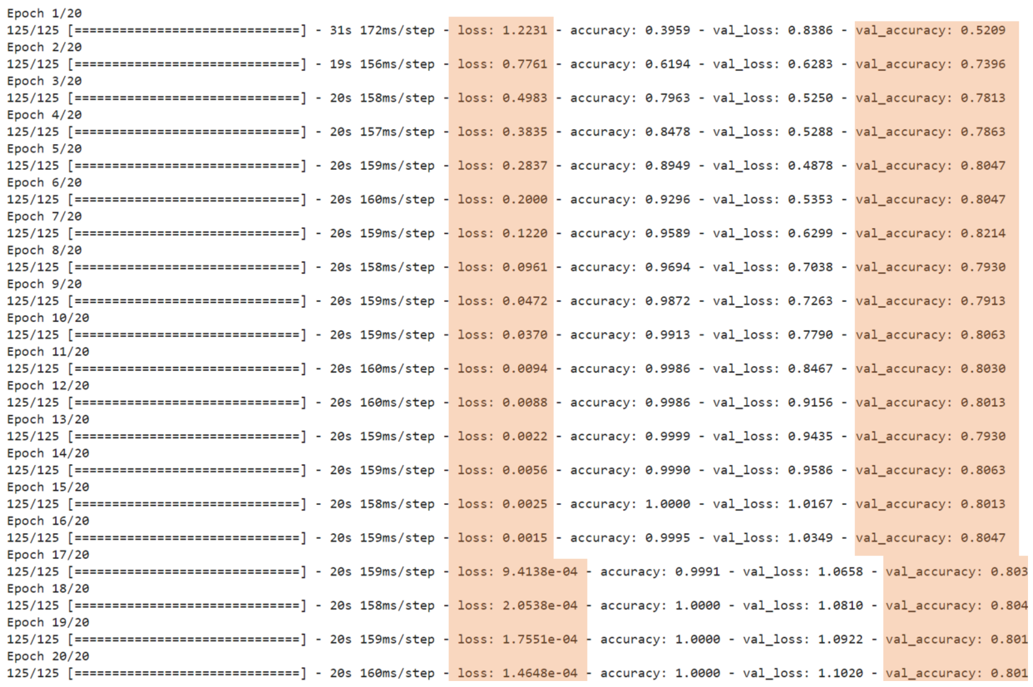
3. Dataset and Training

For static hand gesture recognition, the dataset contains 8337 training images and 1800 testing images belonging to 6 hand gesture classes for finger count 0 to 5. The label for each image is extracted from the image file name and is used as directory name as a standard image data structure used by Tensorflow [12] ImageDataGenerator. All images are rescaled, resized and augmented.

The dataset was split into training and test sets with scikit-learn package. Image data were loaded, resized to 150 x 150, augmented and the pixel values were normalized with tensorflow image package. The processed image data and labels were fitted to the model for training with 10 epochs. The number of epochs is determined when the training and validation errors stop decreasing. Since the validation error does not increase with the increased number of epochs, the number of epochs used does not cause overfitting.

For dynamic hand gesture recognition, 4000 training video clips belonging to 4 hand gesture classes of swiping up, swiping down, swiping right and swiping left, and 600 validation video clips are extracted from 20BN-jester dataset [13]. Each clip is either truncated to or extended to equal frames (30 frames). Each frame is rescaled, resized, and converted to gray image.

The frames are preprocessed with scikit-learn and OpenCV packages. The processed image data and labels are fitted to the model for training with 20 epochs using Tensorflow framework. The number of epochs is determined when the training and validation errors saturate (Fig. 6). The models with high accuracies are saved. Cozmo robot [11] SDK is used to control the robot programmatically. Pre-trained models are loaded. Static images or video clips with defined hand gestures taken by the robot are recognized by the models trained on the static images and video clips, respectively.



```
Epoch 1/20
125/125 [==============================] - 31s 172ms/step - loss: 1.2231 - accuracy: 0.3959 - val_loss: 0.8386 - val_accuracy: 0.5209
Epoch 2/20
125/125 [==============================] - 19s 156ms/step - loss: 0.7761 - accuracy: 0.6194 - val_loss: 0.6283 - val_accuracy: 0.7396
Epoch 3/20
125/125 [==============================] - 20s 158ms/step - loss: 0.4983 - accuracy: 0.7963 - val_loss: 0.5250 - val_accuracy: 0.7813
Epoch 4/20
125/125 [==============================] - 20s 157ms/step - loss: 0.3835 - accuracy: 0.8478 - val_loss: 0.5288 - val_accuracy: 0.7863
Epoch 5/20
125/125 [==============================] - 20s 159ms/step - loss: 0.2837 - accuracy: 0.8949 - val_loss: 0.4878 - val_accuracy: 0.8047
Epoch 6/20
125/125 [==============================] - 20s 160ms/step - loss: 0.2000 - accuracy: 0.9296 - val_loss: 0.5353 - val_accuracy: 0.8047
Epoch 7/20
125/125 [==============================] - 20s 159ms/step - loss: 0.1220 - accuracy: 0.9589 - val_loss: 0.6299 - val_accuracy: 0.8214
Epoch 8/20
125/125 [==============================] - 20s 158ms/step - loss: 0.0961 - accuracy: 0.9694 - val_loss: 0.7038 - val_accuracy: 0.7930
Epoch 9/20
125/125 [==============================] - 20s 159ms/step - loss: 0.0472 - accuracy: 0.9872 - val_loss: 0.7263 - val_accuracy: 0.7913
Epoch 10/20
125/125 [==============================] - 20s 159ms/step - loss: 0.0370 - accuracy: 0.9913 - val_loss: 0.7790 - val_accuracy: 0.8063
Epoch 11/20
125/125 [==============================] - 20s 160ms/step - loss: 0.0094 - accuracy: 0.9986 - val_loss: 0.8467 - val_accuracy: 0.8030
Epoch 12/20
125/125 [==============================] - 20s 160ms/step - loss: 0.0088 - accuracy: 0.9986 - val_loss: 0.9156 - val_accuracy: 0.8013
Epoch 13/20
125/125 [==============================] - 20s 159ms/step - loss: 0.0022 - accuracy: 0.9999 - val_loss: 0.9435 - val_accuracy: 0.7930
Epoch 14/20
125/125 [==============================] - 20s 159ms/step - loss: 0.0056 - accuracy: 0.9990 - val_loss: 0.9586 - val_accuracy: 0.8063
Epoch 15/20
125/125 [==============================] - 20s 158ms/step - loss: 0.0025 - accuracy: 1.0000 - val_loss: 1.0167 - val_accuracy: 0.8013
Epoch 16/20
125/125 [==============================] - 20s 159ms/step - loss: 0.0015 - accuracy: 0.9995 - val_loss: 1.0349 - val_accuracy: 0.8047
Epoch 17/20
125/125 [==============================] - 20s 159ms/step - loss: 9.4138e-04 - accuracy: 0.9991 - val_loss: 1.0658 - val_accuracy: 0.803
Epoch 18/20
125/125 [==============================] - 20s 158ms/step - loss: 2.0538e-04 - accuracy: 1.0000 - val_loss: 1.0810 - val_accuracy: 0.804
Epoch 19/20
125/125 [==============================] - 20s 159ms/step - loss: 1.7551e-04 - accuracy: 1.0000 - val_loss: 1.0922 - val_accuracy: 0.801
Epoch 20/20
125/125 [==============================] - 20s 160ms/step - loss: 1.4648e-04 - accuracy: 1.0000 - val_loss: 1.1020 - val_accuracy: 0.801
```

Fig. 6. Training and validation errors of video clip dataset.

Batch size which defines the number of samples trained together through the neutral network have been varied and various optimizer such as AdamOptimizer, MomentumOptimizer and Adadelta have been used to optimize accuracy and performance.

4. Results and Discussion

The 2D CNN achieved an accuracy of ~98% (Fig. 7). The 3D CNN model achieved ~83% accuracy (Fig. 8). The trained models were tested on a Cozmo robot. A static image or 30 frames of the hand gestures were captured using Cozmo robot SDK and sent to the models for predictions. The robot recognized the hand gestures and moved according to the hand gesture as commanded.

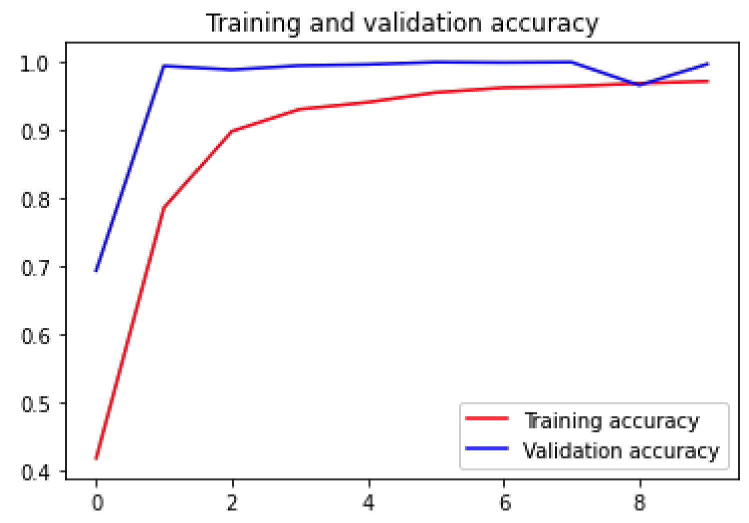


Fig. 7. Training and validation accuracies of static image dataset.

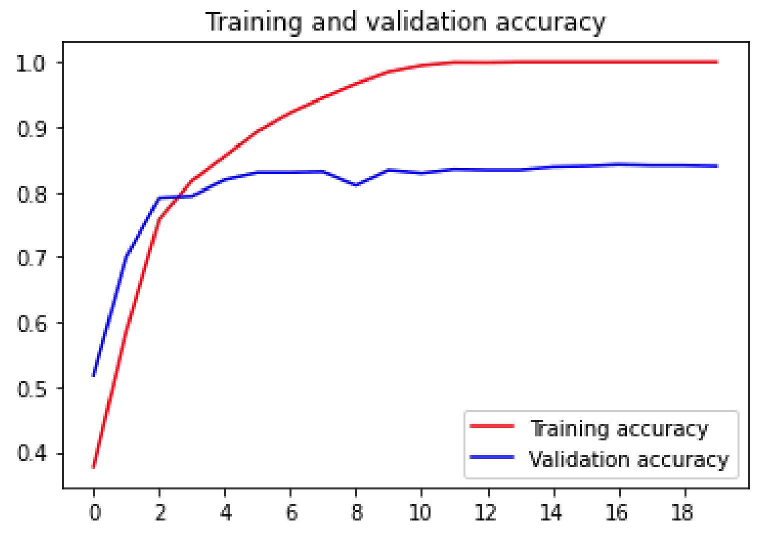


Fig. 8. Training and validation accuracies of video clip dataset.

4.1. Application

The model developed in this research is an effective method for hand gesture recognition using a CNN neural network model and has a wide range of human-computer interaction (HCI) applications, for example:

- A sign language translator that recognizes and interprets hand gestures into spoken languages, and vice versa.
- Hand gesture controlled machines that perform different tasks based on different hand gestures.
- More sanitary and convenient public utilities that could prevent the spread of viruses and other forms of contagious diseases.

4.2. Limitations

The model accuracy to recognize dynamic hand gestures is relatively low. More data will be used to train the model and model parameter tuning will be performed to improve the accuracy. Various algorithms to recognize dynamic hand gestures will be also explored.

5. Conclusion

This research proposes a method for hand gesture recognition and demonstrates that deep learning neural network can be used to recognize static and dynamic hand gestures. As an important application, pre-trained models loaded onto Cozmo robot to recognize hand gesture images and video clips provide a foundation for further developments of human-computer interaction.

Conflict of Interest

The authors declare no conflict of interest

References

[1] Vogler, C., & Metaxas, D. (1998). ASL recognition based on a coupling between hmms and 3d motion analysis. *Proceedings of Int. Conf. Computer Vision* (pp. 363-369). Bombay, India: IEEE.

[2] Lin, H. I., Cheng, C. H., & Chen, W. K. (2013). Learning a pick-and-place robot task from human demonstration. *Proceedings of Int. Conf. Automat. Control* (pp. 312-317). Nantou, Taiwan: IEEE.

[3] Rajesh, R. J., Nagarjunan, D., Arunachalam, R. M., & Aarthi, R. (2012). Distance transform based hand gestures recognition for powerpoint presentation navigation. *Adv. Comput., 3*, 41-48.

[4] Bond, C. F., Omar, A., Mahmoud, A., & Bonser, R. N. (1990). Lie detection across cultures. *J. Nonverbal Behav., 14(3)*, 189-204.

[5] Desai, S., & Desai, A. (2017). Human computer interaction through hand gestures for home automation using microsoft kinect. *Proceedings of the International Conference on Communication and Networks* (pp. 19-29). Singapore: Springer.

[6] Kaur, H., & Rani, J. (2016). A review: Study of various techniques of hand gesture recognition. *Proceedings of the 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems* (pp. 1-5). Delhi, India: IEEE

[7] Lecun, Y., & Cortes, C. (1998). The MNIST database of handwritten digits. Retrieved from http://yann. lecun. com/exdb/mnist/

[8] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1097-1105

[9] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradientbased learning applied to document recognition. *Proceedings of the IEEE* (pp. 2278-2324).

[10] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. *Parallel Distributed Processing, 1*, 318-362.

[11] Anki cozmo robot. Retrieved from http://developer.anki.com

[12] Martín, A. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. Retrieved from https://arxiv.org/abs/1603.04467

[13] Materzynska, J., Berger, G., Bax, I., & Memisevic, R. (2019). The jester dataset: A large-scale video dataset of human gestures. *Proceedings of the IEEE International Conference on Computer Vision Workshops*. Seoul, Korea: IEEE.

Keyi Wang is a high school junior at King's High School located in Shoreline, Washington. Her research interests are artificial intelligence, deep learning, and human computer interaction (HCI). Her research has won second and third place awards in Regeneron International Science and Engineering (ISEF) affiliated fairs. She is a part of FIRST robotics team and was a semi-finalist at the 2018 First Robotics World Championship. Her research papers were accepted by the International Conference on Computer Science and information Technology (ICCSIT) and the International Conference on Bioscience, Biochemistry and Bioinformatics (ICBBB).