Dynamic Hand Gesture Recognition using a Lightweight 3D CNN Model

Shan Xue*, Simon Wu†, Shuao Wu†
*Department of Electrical and Computer Engineering, University of Waterloo, Canada
†Department of System Design Engineering, University of Waterloo, Canada
Emails: s2xue@uwaterloo.ca, {s23wu, s22wu}@uwaterloo.ca

***Abstract*—Hand gesture recognition technology is widely used in various fields, including human-computer interaction, virtual reality, and robotics, due to its versatility. However, existing neural network models for this task often have complex architectures and large numbers of parameters, making them difficult to train in real-time applications. In this study, we designed a light-wight three-dimensional neural network (3D CNN) model, which contains only 0.15 million total parameters. Our model is designed to be trained on a personal computer without GPUs, making it more accessible for researchers and practitioners. We evaluate our model on an experiment dataset: SHREC 2017, which contains 14 different gestures, and it achieve 73.26% accuracy. Overall, our lightweight 3D CNN model demonstrates that good accuracy can be achieved with a much smaller number of parameters, making it a promising solution for dynamic hand gesture recognition in real-world applications.**

***Index Terms*—Dynamic hand gestures recognition, Convolutional neural networks (CNNs), lightweight convolution module.**

I. Introduction

Communication is crucial for humans as it can be considered an act of transmitting information. In our daily lives, people utilize both verbal and nonverbal languages for communication. Nonverbal language can be expressed through hand movements, facial expressions, and body language. Specifically, as seen in sign language, using hand gestures is an essential part of communication. Hand gestures have specific linguistic content and are used to facilitate communication between individuals. Moreover, with the recent development of image processing and visual communications, hand gestures have been increasingly applied to human-computer interaction (HCI) systems, leading to a wide range of applications, including video games, remote surgery, and virtual reality. Therefore, increasing numbers of researchers are dedicating efforts to studying hand gestures recognition (HGR). However, there are several challenges such that HGR is sensitive to variations in size, speed and lighting, performing poorly against complex backgrounds, and accurately detecting the gesturing phase [1].

There are multiple ways to categorize hand gestures. One such way is to classify hand gestures based on observable features or based on the interpretation. Under observable approach, gestures can be classified as either static hand postures or dynamic hand gestures by considering temporal relationships. Static hand postures basically rely on the shape and flex angles of the fingers, while dynamic hand gestures rely on static hand postures and also considers the hand trajectories and orientations. In addition to the observable approach, hand gestures can be categorized based on the interpretation as mentioned before. The most typical classes in this approach have autonomous gestures, illustrators, and regulators. Respectively, autonomous gestures are those that can replace spoken words with gestures, illustrators are gestures used to illustrate spoken words, and regulators are gestures that support the communication between speakers and listeners, such as raising hands to indicate the turn-taking [2], [3].

There are numerous techniques for hand gesture recognition. Those techniques can be classified as (i) Hidden Markov Models (HMM) and other statistical methods, (ii) Artificial Neural Networks (ANNs) and other learning based methods, (iii) Eigenspace based methods, (iv) curve fitting, and (v) dynamic time warping (DTW) algorithms.

In this paper, we mainly focus on HGR in the field of convolutional neural networks (CNNs), with a special emphasis placed on lightweight CNNs. Our goal is to develop a lightweight model that is able to achieve fast training times and maintain good performance without compromising accuracy. To achieve this, we propose a methodology for applying lightweight CNNs to datasets, and experimenting with their performance. The rest of the paper is organized as follows. In the section of related work, we delved into traditional methodologies and CNNs with different architecture. The CNNs are the extended version of standard CNNs. In this section, we reviewed 3D CNN, lightweight CNN, fine-tuned CNN, CNN with long short-term memory (LSTM), and CNN with recurrent neural network (RNN). In the model architecture part, we introduced the model architecture and explained the design inspiration. And the expriment part included data processing before training, model training, fine tuning hyperparameters, and comparison with other publish paper. Finally, we concluded our work in this paper, and setting the future goal of this project.

II. Related Work

The most commonly used techniques in dynamic HGR are HMM, DTW algorithms, and neural network methods. Neural network methods typically employ time-delayed neural networks, CNNs or LSTM networks.

A. Hidden Markov Model

The Hidden Markov Model (HMM) is the most often used statistical method used in HGR. Basically, HMM is a statistical Markov model in which the system being modeled is assumed to be a Markov process with hidden states. The hidden states are unobserved states which means states are not directly visible. Instead, the output dependent on the state is visible. The model incorporates a probability distribution for each state, representing the possible output tokens. Consequently, the output sequence provides some insights into the state sequence [4].

Yang et al. [5] proposed a method based on HMM to recognize complex single-hand gestures. The authors utilized a state-based spotting algorithm to segment continuous gestures and extract features from each segment. Then they applied a data aligning algorithm to align the feature vector sequences for training. Then an HMM is trained individually for each gesture, demonstrating the effectiveness and accuracy of the proposed method.

B. Dynamic time warping algorithm

Kajan et al. [6] proposed a comparison of algorithms for dynamic HGR by using the Kinect v2 sensor. They implemented DTW algorithm, HMM, LSTM networks and CNNs into their application. The DTW algorithm achieved the highest classification success on the original database. Its advantage lies in its simple implementation and good classification success with a relatively small database of reference signals. However, the disadvantage of using the DTW algorithm in real-time applications is the increasing evaluation time as the number of output classification classes increases.

C. Lightweight CNN

Yang et al. [7] proposed a lightweight deep neural network which is MDHandNet for hand gesture and sign language recognition by using micro-Doppler images. Micro-Doppler imaging is a technique that uses radar to capture the Doppler signature of a moving object, such as a human hand, and extract unique features related to hand motion. The authors leverage micro-Doppler imaging to capture the movement of the hand and develop a deep learning model that can recognize hand gestures and sign language. There are three parts in the network. The first part is the backbone subnet, the second part is the hierarchical feature processing subnet, and the third part is the output subnet. In the first subnet, it extracts features from various size receptive fields to gather maximum feature information. The second subnet processes the features to meet high-performance requirements for classification tasks. Finally, the last part combines the features and maps them into category probabilities to classify hand gesture or sign language actions.

D. CNN + LSTM

Zhang et al. [8] proposed a Short-Term Sampling Neural Network (STSNNs) which aims to recognize dynamic hand gestures using a sliding window technique combined with neural networks. Hand gesture recognition requires capturing both spatial and temporal information from video inputs. This approach divides consecutive frames into groups and takes a random sample from each group, ensuring a fixed number of samples for each video sample regardless of its duration. Representative samples are created by combining RGB frames and optical flow frames, and a ConvNet is used to capture features from each sample. All ConvNets share the same parameters to reduce training parameters. An LSTM module is used to learn long-range temporal features from the feature sequence, and the SoftMax function is used to calculate predicted class probabilities.

III. Model Architecture

In our lightweight 3D CNN model for hand gesture recognition, the objective is to efficiently process and classify hand gestures with computational efficiency while maintaining accuracy. After learning from existing research, especially inspired by AlexNET [12], our model comprises 3 convolutional layers and 2 fully connected layers, as described below.

The first initial convolutional layer performs convolution operations on the input frames using a set of filters. These filters are designed to detect low-level features, such as edges and textures, from hand gesture images. The depth channel proves advantageous in this layer, as it supplies supplementary information on the distance between the camera and objects. This assists the model in comprehending the spatial relationship between hand joints more effectively.

The second convolutional layer is built upon the first convolutional layer, the second layer refines the detected features by identifying more intricate features like corners and curves. These features aid in differentiating between various hand gestures.

For the third convolutional layer, it extracts even higher-level features that capture the unique characteristics of each hand gesture, facilitating the model's ability to generalize well across different hand shapes, sizes, and orientations.

The fourth level is the first fully connected layer of our model. This layer flattens the high-level features extracted by the convolutional layers and combining them into a single vector. The layer is designed to effectively captures the relationships between the features and prepares the data for the final classification step.

As the final layer of the model, it functions as a classifier. It takes the output from the first fully connected layer and assigns probabilities to each of the 14 gestures. The gesture with the highest probability is deemed the predicted gesture.

To reduce the overfit, 3D max pooling, drop out and batch normalization are used in multiple layers as regularization. In the following experiments, we fine tuned these hyperparameters, as well as some other parameters.

IV. Experiment

A. The Dataset

We selected a skeleton-based dataset called SHERC dataset [13] to evaluate our model. It contains 14 gestures which can

be performed in two ways, either using one finger or the whole hand. There are 28 participants who perform each gesture between 1 and 10 times, resulting in a total of 2800 sequences. All participants are right-handed. Each sequence is labeled based on the type of gesture, the number of fingers used, the performer, and the trial number. Each frame of the sequences contains a depth image and the coordinates of 22 joints. These coordinates are provided in both the 2D depth image space and the 3D world space, forming a full hand skeleton. The data was captured using the Intel RealSense short-range depth camera, with a capture rate of 30 frames per second and a depth image resolution of $640 * 480$. Thus, the shape of each frame is $640*480*4$, where 4 is the RBG plus depth. The depth channel in hand gesture recognition provides valuable spatial information, allowing the model to better understand the 3D structure of hand movements and reducing the influence of skin color and lighting conditions on the model. The length of the sample gestures varies between 20 to 50 frames.

B. Sample Frames

Due to limitations in computational resources, it is necessary to reduce the number of frames for each sequence. To maintain the coherence of the sequence, a modified stratified sampling method was employed to sample 15 frames for each sequence. The method is defined by the following algorithm:

$$n = 15 * a + b \tag{1}$$

$$b = 2 * c + d \tag{2}$$

In (1), n represents the total number of frames in a sequence, a represents the initial sub-group size, and b represents the remainder that is used to adjust the sub-group size. The quotient c is obtained from (2), and the size of the first c sub-groups is increased by 2. For the remainder d, if it is equal to 1, the size of the last sub-group is increased by 1 while the sizes of the other sub-groups remain unchanged. If d is equal to 0, the sizes of all remaining sub-groups are left unchanged. By using this modified stratified sampling method, each sequence is stored as a 4-dimensional array with shape $640 * 480 * 15 * 4$.

C. Bounding Box

Due to the presence of both gesture information and participant silhouettes in each image frame, we have decided to use the skeleton information of the hand to set the bounding box and only retain the gesture information within the box. The dataset has included the full hand skeleton information, which are 22 pairs of data that represent the location of the 22 joints. Thus, the boundaries of the bounding box were decided by the leftmost, rightmost, topmost, and bottom-most points.

D. Downsampling

Since we applied the bounding box to frames, the first two dimensions of each frame equal to the size of its bounding box and the size of the bounding box was different, we used the cubic spline interpolation to resample each frame. Cubic spline interpolation is a good option for image downsampling because it can preserve the overall shape and smoothness of the image. After downsampling the image in the original ratio, we distort each frame into a square fit. Since each frame contains only a hand gesture, distortion doesn't have much impact on the contained information of each frame. By using cubic spline interpolation for downsampling and distortion for reshaping, the computational complexity was reduced while maximizing the preservation of information for subsequent gesture recognition tasks using the deep neural network.

E. Fine Tune

After we finished the image processing, we randomly split the dataset for training and testing, where 70% of the data was allocated to the training set and 30% to the testing set. In the experiment in this paper, we use the PReLU activation function [16] and Adam optimizer [17] at first, as PReLU has been shown to alleviate the "dying ReLU" problem by allowing negative values and thus improving the generalization ability of deep neural networks [21]. To avoid overfitting, we decided to fine tune the epoch number and use dropout as the regularization technique in the model. When training a neural network, it is often difficult to determine the optimal number of epochs that too few epochs may let the model underfitting, and too many epochs may let the model overfitting. Here, we use a "patience" parameter during training. It refers to the number of epochs that the model can continue training without improvement on the validation before the training is stopped. Because of the light-weight characteristic of the model, the "patience" was set as 10 to avoid the training process being stopped too early to the model. At the beginning, we randomly drop out 50% of the first fully connected layer's output at the same time. The accuracy achieved 70.02% when the model was applied on the testing dataset. The plottings of loss and accuracy on training are smooth, but are zigzagged on validation, which means this model is not robust for the dataset. Then, we implement different dropout designs on the mode, these designs and testing accuracies are listed in Table I. Considering both robustness and testing accuracy, we finally utilized the dropout design (0.2, 0.3, 0.4, 0.5) in the model. Then, we consider using different optimizers to increase the testing accuracy. In the experiment, five optimizers were selected which are Adam [17], AdamW [18], Adadelta [19], Adamax [18], and Adagrad [20]. Figure 1 shows the loss on each epoch for each optimizer in both training and validation. Figure 2 shows the accuracy on each epoch for each optimizer in both training and validation. Figure 3 shows the testing accuracy for using each optimizer. The best optimizer is Adamax, its testing accuracy achieves 73.26% based on 53 training epochs.

F. Comparison

Our model has a major advantage over the models presented in the three published papers [9]–[11] due to its light-weight nature and the ability to achieve high accuracy with limited computational resources. The PointLSTM-middle model in [9] has a relatively low number of parameters, 1.2M, and a low

TABLE I
DROPOUT DESIGNS AND TEST ACCURACY

Dropout Design	Test Accuracy (%)
0.5 on first fully connected layer	70.02
(0.2, 0.4, 0.4, 0.2) on first four layers	66.19
(0.2, 0.2, 0.2, 0.2) on on first four layers	72.90
(0.1, 0.2, 0.3, 0.4) on first four layers	69.06
(0.2, 0.3, 0.4, 0.5) on first four layers	71.22

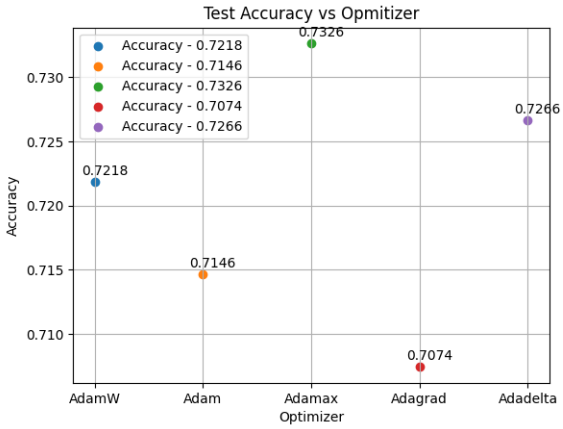

Fig. 3. Testing Accuracy for Each Optimizer

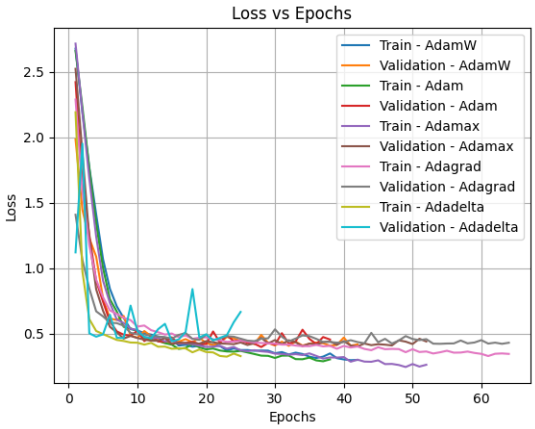

Fig. 1. Training and Validation Loss for Each Optimizer

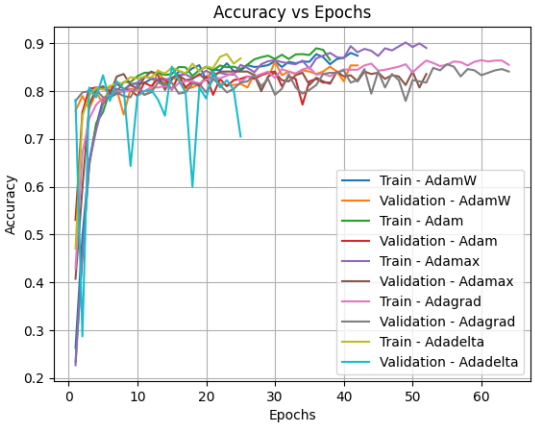

Fig. 2. Training and Validation Accuracy for Each Optimizer

computational complexity of 16.1G floating-point operations (FLOPs), but still achieves high accuracy of around 95%. However, it used a single Tesla P100 GPU to train this model, which may not be available to many researchers. In contrast, the 3D CNN_VIVA_4 model in [10] achieves a similar accuracy of around 80% but uses a much higher number of parameters, 73.1M, and 3.9M convolutional parameters, which require significant computational resources to train the model, and its accuracy is not much higher than us. In [11], the DD-Net model has a light-weight version with only 0.15M parameters, which has similar number of parameters as our model, but its accuracy is 91.8%, which is much higher than our model's accuracy. Additionally, it can achieve a super-fast speed of 3,500 FPS on one GPU or 2,000 FPS on one CPU, making it suitable for real-time applications.

V. FUTURE WORK

In this study, we have developed a lightweight 3D Convolutional Neural Network for hand gesture recognition. Although the model shows some promising results, the model's performance on the validation dataset shows some zigzagging patterns, indicating that the model might not be robust enough for the dataset. Further experimentation with the architecture or hyperparameters could potentially improve the model's robustness. In future work, we will focus on the aspects of advanced model architecture, regularization technique and hyperparameter optimization.

We want to investigate the incorporation of more complex and advanced architectures, such as introducing Residual Networks (ResNets) [14] and DenseNets [15] architecture into our model, to enhance the model's capability to capture complex spatial and temporal patterns.

To further reduce overfitting and improve generalization, we plan to explore additional regularization techniques, such as L1 and L2 regularization, in conjunction with dropout. We also willing to conduct a more comprehensive hyperparameter search, focusing on learning rate, batch size, filter sizes,

number of layers, and number of filters, to optimize the model's performance.

VI. Conclusion

In [12], we analyzed the skeleton information in sequences and built a light-weight CNN model for dynamic hand gesture recognition. Our model has only 0.15M parameters, making it highly efficient and suitable for training with limited computational resources. We trained the model with different optimizers and achieved a high accuracy of 73.2%, which is impressive considering the limited resources used for training. Although our model shows promising results, but also exhibits some performance issues on the validation dataset. Future work will focus on incorporating more complex architectures, regularization techniques, and conducting a more comprehensive hyperparameter search to optimize the model's performance. This will potentially enhance the model's capability to capture complex spatial and temporal patterns, reduce overfitting, and improve generalization. Overall, we are confident that our research has laid a strong foundation for developing efficient and accurate models for dynamic hand gesture recognition, and we look forward to making further progress in this area.

References

[1] 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, vol. 35, no. 1, pp. 221–231, 2013.

[2] L.L. Barker, L.A. Malandro, A.B. Deborah, Nonverbal Communication, 2nd ed., Addison-Wesley, MA, 1989.

[3] A. Kendon, Gesture and speech: how they interact, in: John M. Wiemann, Randall P. Harrison (Eds.), Nonverbal Interaction, Sage Publications, Beverly Hills, 1983.

[4] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of The IEEE 77 (2), 1989,pp.257–285.

[5] Z. Yang, Y. Li, W. Chen, and Y. Zheng, "Dynamic hand gesture recognition using hidden markov models," in 2012 7th International Conference on Computer Science & Education (ICCSE), pp. 360–365, IEEE, 2012.

[6] S. Kajan, J. Goga and O. Zsíros, "Comparison of Algorithms for Dynamic Hand Gesture Recognition," 2020 Cybernetics & Informatics (K&I), Velke Karlovice, Czech Republic, 2020, pp. 1-5, doi: 10.1109/KI48306.2020.9039850.

[7] Y. Yang, J. Li, B. Li, and Y. Zhang, "Mdhandnet: a lightweight deep neural network for hand gesture/sign language recognition based on micro-doppler images," vol. 25, 2022.

[8] W. Zhang, J. Wang, and F. Lan, "Dynamic hand gesture recognition based on short-term sampling neural networks," IEEE/CAA Journal of Automatica Sinica, vol. 8, no. 1, pp. 110–120, 2021.

[9] Y. Min, Y. Zhang, X. Chai and X. Chen, "An Efficient PointLSTM for Point Clouds Based Gesture Recognition," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 5760-5769, doi: 10.1109/CVPR42600.2020.00580. [Accessed: 11-Apr-2023].

[10] D. Wang, G. Zhao, G. Li, L. Deng, and Y. Wu, "Compressing 3D CNNs based on tensor train decomposition," arXiv.org, 11-Aug-2020. [Online]. Available: https://arxiv.org/abs/1912.03647. [Accessed: 11-Apr-2023].

[11] F. Yang, S. Sakti, Y. Wu, and S. Nakamura, "Make skeleton-based action recognition model smaller, faster and better," arXiv.org, 18-Mar-2020. [Online]. Available: https://arxiv.org/abs/1907.09658. [Accessed: 11-Apr-2023].

[12] Krizhevsky A, Sutskever I, Hinton G. ImageNet classification with deep convolutional neural networks. Communications of the ACM. 2017;60(6):84-90. doi:10.1145/3065386

[13] Q. d. Smedt et al. SHREC'17 Track: 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset. 3DOR - 10th Eurographics Workshop on 3D Object Retrieval, Apr 2017, Lyon, France. pp.1-6, doi: 10.2312/3dor.20171049

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2016:770-778. doi:10.1109/CVPR.2016.90

[15] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely Connected Convolutional Networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2017:2261-2269. doi:10.1109/CVPR.2017.243

[16] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet Classification. In: 2015 Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 1026-1034

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv.org, 30-Jan-2017. [Online]. Available: https://arxiv.org/abs/1412.6980. [Accessed: 12-Apr-2023].

[18] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," arXiv.org, 04-Jan-2019. [Online]. Available: https://arxiv.org/abs/1711.05101. [Accessed: 12-Apr-2023].

[19] M. D. Zeiler, "Adadelta: An adaptive learning rate method," arXiv.org, 22-Dec-2012. [Online]. Available: https://arxiv.org/abs/1212.5701. [Accessed: 12-Apr-2023].

[20] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." [Online]. Available: https://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf. [Accessed: 12-Apr-2023].

[21] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," arXiv.org, 27-Nov-2015. [Online]. Available: https://arxiv.org/abs/1505.00853. [Accessed: 12-Apr-2023].