TEAM 59: LEAP MOTION HAND GESTURE CLASSIFICATION

Xuezhu Hong, Yuchen Lu, and Shengzhe Zhang

University of California San Diego, La Jolla, CA 92093-0238

ABSTRACT

Hand gesture recognition forms a foundation for low-level Human-Computer Interaction (HCI). In this paper, Hand Gesture Recognition Database from Kaggle with ten different gestures is used to train a model. We implemented a Convolutional Neural Network (CNN) with data augumentation and used a 3-phase training technique to increase training throughput and reduce overfitting. The final testing accuracy obatined is 99.9%. We also leveraged transfer learning to train a model for American Sign Language (ASL) dataset with 3-phase training technique and achieves a testing accuracy of 99.6%.

***Index Terms*—** Convolutional Neural Network (CNN), Transfer Learning, Image Augmentation, Image Segmentation

1. INTRODUCTION

Hand gestures are a fundamental way for human beings to express ideas. Being able to recognize hand gestures allows human to send information and instructions directly to computers. It is a powerful and robust method of Human-Computer Interaction (HCI). Hand gesture recognition has been a widely discussed topic in the computer vision community. Recent years, most researchers use deep learning to classify hand gestures [1]. In this paper, we leverage convolutional neural network and transfer learning to classify ten different hand gestures using near infra-read images obtained from the leap motion sensor.

2. RELATED WORK

In [2], an unsupervised approach with pair-patch comparison features is used for training hand gesture dataset. Machine learning model is based on random forest and distribution of samples can be quickly described by pair-patch comparison features. To increase detection accuracy and efficiency, skin detection is used before implementing the random forest classifier. We draw the idea of preprocessing and image segmentation used in this paper from [2].

It is also been shown in [3] that Deep Convolutional Network demonstrate outstanding performance in image classification and recognition. An Adapted Deep Convolutional Neural Network (ADCNN) is trained with data augmentation. The presence of network initialization and regularization helps reduce overfitting. Inspired by [3], data augmentation is also implemented in this paper.

3. DATASET AND FEATURE PROCESSING

The dataset we use is the leap motion hand gesture recognition database [4]. The dataset is composed of 10 different hand gestures performed by 5 men and 5 women. The ten classes are showing in Fig. 1. There are 20000 photos with size 640x240 in the dataset. All the gestures were captured in leap motion by infrared cameras, therefore besides static gesture classification, it also provides the possibility of accomplishing real-time gesture detection and classification.

For preprocessing the photos, firstly we convert them to grayscale, and resize them to 224x224, which is a suitable input size for many pretrained models such as vgg. Secondly we do the image augmentations with random variances. Then all the photos are subject to segmentations to extract more meaningful features. We also normalize the images to get a more stable training result. Then we split train and test sets by ratio of 0.8 / 0.2. At training time, we use data generator in Keras to further add shift variances to the training images. The purpose of this step is to introduce more new samples. This will enlarge the training space and further improve the generality of the model.

3.1. Image Augmentation

Many captured hands are not located at the center of the photos originally, and we do not crop the photos by taking the hands as the center since some position variances in the photos are desired. Therefore, we mannully injected some variances to the images. Each image is randomly subject to one of these four variances: rotation, shift, horizontal flip, and zoom

Fig. 1. Top row: C, palm side, index, OK, down. Bottom row: fist, L, palm, fist side, thumb

Fig. 2. Left: original photo; right: rotated photo

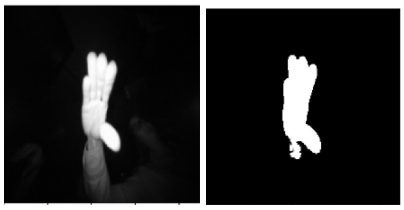

Fig. 3. Left: original photo; right: segmented photo

in or zoom out. Fig. 2 is an exmaple of rotation. By performing data augmentation, we double the size of the dataset, which improves the robustness of the model. The added variances also help reduce overfitting and enhance the generalization ability.

3.2. Image Segmentation

We perform the image segmentation using Otsu's threholding [5]. During the segmentation process, each pixel will be assigned to certain groups or clusterings based on a thresholding rule. It can help separate the object we want to focus on from the background. Otsu's thresholding assumes the image histogram is bimodal, and it will automatically determine an optimal threshold above which the pixel value will be assigned 255, or the pixel value will be assigned 0 if it is below the threshold. Since the infrared photos were taken with infrared cameras so they are almost black and white with the highest intensity at the hand regions, therefore Otsu's method is reasonable to be used to do segmentation. Fig. 3 shows an example of segmented image.

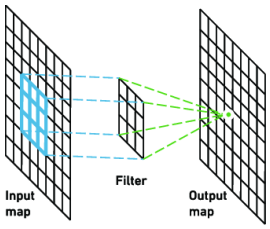

Fig. 4. convolutonal layer

4. METHODS

4.1. Convolutional Neural Network

Convolutional Neural Network(CNN) is a certain type of neural network which is frequently used to deal with computer vision tasks such as image classification. It is composed of several convolutional layers which is able to extract features from images. Convolution operation on images will have a convolutional kernel sliding horizontally and vertically through the whole image with predefined size and step length. The process is referred to as sliding dot product or cross-correlation [6]. The resulting image will be a matrix containing significance for each pixel. Fig. 4 shows the schematic of a simple convolutional layer.

In most cases, convolutional layer is not the only type of trainable layer in CNN. Several Dense layers, which are fully-connected, are located near the output layer. These fully-connected layers map different features obtained from convolutional layers to the hidden nodes, which could be regarded as perceptrons or "weak learners", and eventually connect them to different labels.

The existence of non-linearity is one of the main reasons why neural network works so well in regression and classification tasks. The output of each linear layer is passed through a non-linear activation function (usually ReLU ,Tanh or Sigmoid) to provide the required non-linearity. Additionally, some other non-linear layers, e.g. Dropout and Max-Pooling, are frequently implemented to reduce over-fitting.

4.2. Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is currently the standard method in training neural networks [7]. It aims at minimizing the loss for a whole batch in each gradient step. It has several hyper-parameters: the learning rate determines the step size towards the computed direction; weight decay regulates the l-2 norm of the whole trainable parameter matrix so that they would not explode; momentum determines how much the previous direction should take into consideration in current step, while the optional annihilation parameter makes the gradient step even smaller when the neural network has almost converged.

In our experiments, we will only implement Adam optimizer [8] for all models and training processes. Adam has a more complicated regime of implementing previous momentum, which is regulated by β_1, and β_2, which are the first and second order momentum decay rate respectively. We will use all default hyper-parameters given by Keras, which are $lr = 1e-3, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-7$.

4.3. Transfer Learning

Transfer Learning is a popular network training technique. It allows us to train the network starting from the saved weights

from a model pre-trained for a different but related task [9]. It is theoretically determined how much these would help with a faster convergence, while we could understand such process as leveraging the existing structure in the network. We could describe some natural semantics to make it quicker adapt to the new task.

To maintain the previously learned "structures", we may want to freeze some of the trainable layers. That means we make them not trainable to preserve the information from previous tasks [10]. This is also known as "fine tuning" as it will only change part of parameters in the network. Also, freezing the layers near the input end is helpful in preventing the vanishing of gradient due to the deepness in an extremely deep network without recurrent structure, such as Xception.

While transfer learning may be beneficial to training, it is not suitable for all cases. If the natural semantics of the pre-trained dataset and current dataset are so different, transfer learning may not work very well. Also, if too many trainable layers are frozen, the remaining trainable parameters may be insufficient for new classification tasks.

5. EXPERIMENTS

In this section, we (1) use the dataset described in Section 1 for finding the capable CNN model and training strategy, and (2) combine with American Sign Language (ASL) dataset to implement and test transfer learning. The latter contains 87,000 colored hand gesture images for 29 classes. For transfer learning, we reshape the input size to be 100×100.

5.1. Baseline Model

For the first experiment, we started with the baseline of a simple LeNet [11], which is a 5-layer simple network invited by Yann LeCun in 1998. The first 2 weighted layers are convolutional layers, while the rest 3 are fully-connected dense layers. The activation function the model implemented is ReLU. Max-Pooling middle layers are set after each convolutional layer.

Figure 5 represents the structure of LeNet-5 for our given input size. Even though neither the structure is deep nor any filter is wide, the total volume of parameters is still high due to the dense layer taking the flattened vector as input. The high volume of parameters makes the network slower to train with a higher tendency to become over-fitting.

5.2. Optimization of Model

Based on LeNet, we developed our new network. This is a similar structure, but with one more convolutional layer and one less dense layer. Such alteration makes the input width after flatten layer much narrower, thus the total amount of parameters is reduced from 6 million to 700k. Figure 6 illustrates the structure of our new network.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 6)	60
max_pooling2d (MaxPooling2D)	(None, 111, 111, 6)	0
conv2d_1 (Conv2D)	(None, 109, 109, 16)	880
max_pooling2d_1 (MaxPooling2	(None, 54, 54, 16)	0
flatten (Flatten)	(None, 46656)	0
dense (Dense)	(None, 128)	5972096
dense_1 (Dense)	(None, 84)	10836
dense_2 (Dense)	(None, 10)	850

Total params: 5,984,722
Trainable params: 5,984,722
Non-trainable params: 0

Fig. 5. The structure of LeNet for 224*224 input

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 6)	60
max_pooling2d (MaxPooling2D)	(None, 111, 111, 6)	0
conv2d_1 (Conv2D)	(None, 109, 109, 16)	880
max_pooling2d_1 (MaxPooling2	(None, 54, 54, 16)	0
conv2d_2 (Conv2D)	(None, 52, 52, 16)	2320
max_pooling2d_2 (MaxPooling2	(None, 26, 26, 16)	0
flatten (Flatten)	(None, 10816)	0
dense (Dense)	(None, 64)	692288
dense_1 (Dense)	(None, 10)	650

Total params: 696,198
Trainable params: 696,198
Non-trainable params: 0

Fig. 6. The structure of our model for 224*224 input

By using our default Adam optimizer to train these two networks in TensorFlow-based Keras platform, the two models give similar test accuracy. The curves for LeNet-5 is shown in the left in Figure 7. These models are obviously overfitting to the training data, since the train accuracy reaches almost 1 while the test accuracy stuck at around 0.98 and the test loss gradually increases with more epochs [12].

While our new model does not significantly reduce overfitting, it actually reduces the volume of parameters and maintains similar accuracy. Therefore, in the rest of experiments, we will only implement our own model. It is well-mentioning that we also considered and attempted to implement some more complicated popular CNN models, such as VGG-16, ResNet, or Xception. However, we assumed these models could not work well on our task due to the extent of complication of the natural semantics of hand gesture language versus the super sophistic structures of these networks. Our few attempts justify these assumptions because these models cannot even converge for our dataset with basic training setting.

5.3. Optimization of Training Process

We implement data augmentation technique we described in 3.1 and try to fix overfitting [13]. The image generator can

create new image data by making random manipulations to the original data, meaning that each input we taking to train the network is now equivalent to a sample form a much larger sample space than the original train set. Such randomness we created makes the model much harder to fit all training data. The training curves for our model trained with image augmentation technique is shown in the right in Figure 7. Notice that the training accuracy/loss and testing accuracy/loss are almost identical after 20 training epochs, meaning that the model is not only capable for the training set, but also capable for all the possible real-world data instead. The average test accuracy we reached by this process in about 99%, which is 1 percent better than the baseline model.

However, we still want to get a even higher accuracy and there is a main drawback of using image augmentation during the whole training process: this requires the image generator module to generate random data at the runtime of training and thus seriously slower the training throughput.

To overcome these issues, we propose a 3-phase training process: in stage 1, we only use original training data to train the network to achieve the upper-bound test accuracy in normal case; in stage 2, we implement the random image generator to generate random data for training the network until we reach the similar test accuracy in stage 1. Finally, in stage 3, we use original data again. However, to reduce the parameter we want to change, inspired by transfer learning, we freeze the weights in all convolutional layers. We keep training the network until it reaches the highest accuracy.

Using our model with the newly designed training process, the accuracy versus epoch is shown in Figure 8. The huge accuracy drop occurred when we began implementing data augmentation. The final test accuracy is 0.999, which is the best among our experiments.

5.4. Transfer Learning

For the second experiment, we attempt to use transfer learning to train a model for ASL dataset based on the model of Leap-Motion dataset with 99% test accuracy. Since ASL has much more volume and classes of data, we are now compressing all data in both datasets to size 100×100 and in grayscale. We firstly train and save a model with the same structure in 5.2 but with compressed input size to reach 99% test accuracy, and then drop the last layer of the network, shifting it to a 29-output dense layer instead.

From here, we have 3 different re-training settings: a base strategy with no special changes; freezing all convolutional layers; and the 3-phase training technique we described in 5.3. The training curves and final accuracy are shown in Figure 9 and Table 1 respectively. From the curves, we find that not freezing the top few layers gives a better converge rate, while from the table we find that the final test accuracy of freezing these layers is slightly higher. However, these results may given by the statistical fluctuation since they are very subtle.

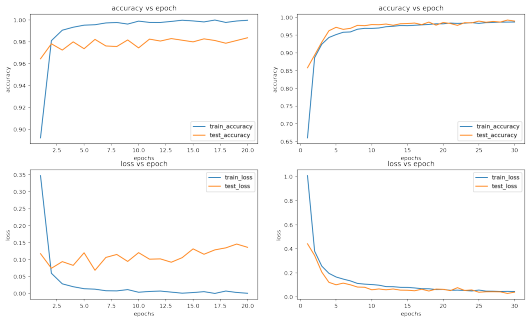

Fig. 7. Left: training curves for LeNet-5 without image augmentation; right: training curves for our model with image augmentation

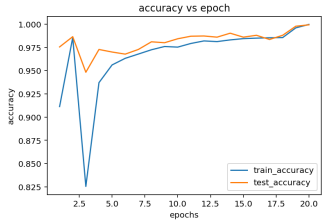

Fig. 8. Accuracy vs epoch of 3-phase training

	base	freeze	3 phase
train	97.61	99.72	99.98
test	97.62	98.61	99.60

Table 1. Accuracy of transfer learning for ASL dataset with different training settings

6. CONCLUSION

We discuss the pre-processing of the dataset we implemented. Based on LeNet-5, we proposed our model and training strategy, which are particularly capable for such a gesture classification task. We managed to use transfer learning to train a network for another relevant dataset.

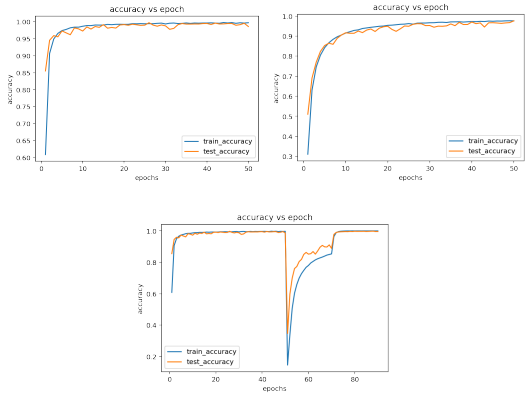

Fig. 9. Left top: TL without layer freezing; right top: TL with layer freezing; bottom: TL with 3-phase retraining.

7. REFERENCES

[1] J. Yashas; G. Shivakumar. Hand gesture recognition : A survey. *International Conference on Applied Machine Learning (ICAML)*, 2019.

[2] X. Zhao; Z. Song; Y. Zhao; F. Zheng. Real-time hand gesture detection and recognition by random forest. In *Communications and Information Processing*. Springer International Publishing, 2012.

[3] A. A. Alani; G. Cosma; A. Taherkhani; T. M. McGinnity;. Hand gesture recognition using an adapted convolutional neural network with data augmentation. *2018 4th International Conference on Information Management (ICIM)*, 2018.

[4] Tomás Mantecón, Carlos R. del Blanco, Fernando Jaureguizar, and Narciso García. Hand gesture recognition using infrared imagery provided by leap motion controller. In Jacques Blanc-Talon, Cosimo Distante, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, pages 47–57, Cham, 2016. Springer International Publishing.

[5] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[6] Wikipedia contributors. Convolutional neural network.

[7] L. Bottou. Online algorithms and stochastic approximations. *Online Learning and Neural Networks*, 1998.

[8] J. Ba D. P. Kingma. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs.LG]*, 2014.

[9] J. West; D. Ventura; S. Warnick. Spring research presentation: A theoretical foundation for inductive transfer. 2007.

[10] V. Roman. Cnn transfer learning fine tuning.

[11] Y. Lecun; L. Bottou; Y. Bengio; P. Haffner. Gradient-based learning applied to document recognition. *IEEE*, 1998.

[12] A. Skrondal B.S. Everitt. *Cambridge Dictionary of Statistics*. Cambridge University Press, 2010.

[13] J. Wang L. Perez. The effectiveness of data augmentation in image classification using deep learning. *arXiv:1712.04621 [cs.CV]*, 2017.

Individual Contributions

Yuchen Lu: Yuchen finished the Abstract, Introduction and Related work section. He also relied on questions from other reviewers. Coding-wise, he implemented a simple CNN as a baseline for our model at the early stage.

Shengzhe Zhang: Shengzhe was in charge of image loading, extraction and integration. He did the image preprocessing and feature extraction including augmentation and segmentation. He also did the corresponding parts in presentation and paper.

Xuezhu Hong: Experimentations of different models and training strategies. Finish these contents in presentation and report.

Replies to critical reviews

Critical review from team 64:

1. Potentially add a live video demonstration, where you can detect hand gestures on a real time camera feed. The presentation, particularly in the background slide, made it slightly implied that they had detected hand motions from a live camera feed.

   Our response: It is a good idea to do real-time hand gesture detection. We decided that the scope of our project only focuses on non-real-time because we want to obtain a confident result with high accuracy.

2. It's unclear why they chose this particular model for their training? What are the pros and cons of this method over other general deep learning methods?

   Our response: We started off with LeNet as a baseline and modified it and implemented our own CNN. We also used a data generator to avoid overfitting. Compared to other general methods, our model ensures a high accuracy without overfitting the model. The downside is that training time is quite long mainly due to the data generator along the training process. Please refer to section 5 in our paper for more information.

Critical review from team 66:

1. We think that group 59 should add more background information to introduce the importance of the hand gesture detections.

   Our response: Agreed. We emphasized its importance in HCI in our paper.

2. We think that it needs more clarification about why they chose this particular model, as well as the pros and cons of such a model.

   Our response: Same question as question 2 from team 64. Please see above.

Critical review from team 36:

1. The presentation was supposed to be of 12 minutes. The rest 10 minutes should be code run through. It would have really helped if you reduced the talk to 12 minutes and spent the rest of time talking about the code.

   Our response: Agreed.

2. The only code we could see was the prediction part. Including data preprocessing/model training in the presentation would have made things clearer.

   Our response: Agreed.

3. In the presentation, you mentioned that the dataset was doubled. Was there class imbalance in the dataset? If yes, how did you fix it?

   Our response: The dataset itself is quite balanced in terms of image per category. However, some of them are off-centered and we don't want to simply crop the hand without leaving some position variance in the photos. Therefore, we randomly lift, flip, zoom in/out on each image and double our dataset.

4. While explaining LeNet5, you say that average pooling is used for non-linearities. Don't we use activation functions for that. Why is the activation function not used?

   Our response: I think we made a mistake there. Average pool is not for non-linearities. It helps extraction "sharpest" feature in our image. Activation function (ReLU) is used.

5. In the improvement for LeNet5, it would be better if you provided plots for accuracy and convergence to prove your claim that the accuracy is the same but convergence rate is higher.

   Our response: Agreed. That would be more convincing. In the paper, we numerically justified.

6. For the revised training model, why did you select these particular values of the epochs for the different layers? Did you try different sets of epochs as well?

   Our response: After some experiments, we found that the training accuracy/loss and testing accuracy/loss are almost identical after 20 training epochs.

7. You mentioned that you compared your results with deeper neural networks like VGG16. What was the result that you got in comparison with these models? A plot showing accuracy would help compare.

   Our response: Agreed with adding a plot. We did some attempts with a basic training setting and found VGG cannot converge for our dataset.

8. It would really help if all the final results were shown graphically, comparing accuracies of all the implemented models with each other.

   Our response: Agreed. It is a good idea for presentations to use more graphics than numbers. In our paper, however, due to page limit, we only reported selected graphs to emphasize our optimization process using data augmentation and 3-phase training technique. Anyways, good point indeed.