Hand Gesture Recognition Using Deep Learning Algorithms

Yufei Zhang
Rice University
6100 Main St, Houston, TX 77005
yz218@rice.edu

Github link: https://github.com/yufeiStella/COMP576_Final_report

Abstract

Human-computer interaction is a popular science and technology that effectively improves people's quality of life and facilitates daily travel and life. In addition to voice recognition, gesture recognition is also an important application. In this project, I trained CNN models with different architectures and hyperparameters on a static image hand gesture dataset. Finally, I got very high training and testing accuracy (99%) and good performances on different CNN models. In the future, I will also want to deal with more realistic hand gesture recognition especially on analyzing temporal, sequential data, such video.

1. Background

Human-computer interaction technology is a popular and widely used technology in the field of computer science today. One of the mature and widespread applications is speech recognition, which we use almost every day, such as voice texting, voice search, and in-car voice assistant. [1] This greatly helps free people's hands and improves safety. But thinking about it further, in addition to language, human communication also has the blessing of various gestures and expressions. Gestures are more universal than languages. For example, when you go to a country with different languages， but can use gestures to complete basic communication. When you are sitting in a car through the window to signal that you are giving way to pedestrians. Another example is, in a noisy environment, you use gestures to tell your friends, next step you will go left or right. Here I want to focus on hand gesture recognition, which has also begun to penetrate our daily life. For example, instead of touching a screen, we can interact with a machine by simply waving our hands to switch car music. [7]

Hand gesture recognition requires analyze image data, in other words, the image classification. While most image are unstructured, we need a way to properly analyze the images so we can use them. [6] Although we found that sometimes it is not as precise and fast as we thought. In this project, I'm going to try to find an efficient and accurate way to recognize gestures, at least at the model level.

2. Dataset

For this task, I use one static hand gesture image dataset from Kaggle, the Hand Gesture Recognition Database. [3] The database is composed by 10 different hand-gestures that were performed by 10 different subjects (5 men and 5 women). The database is structured in different folders as:

- /00 (subject with identifier 00)
  - /01_palm (images for palm gesture of subject 00)
    - /01*palm/frame*197957*r.png,…,fram e*198136_l.png, … (images that corresponds to different samples obtained for the palm gesture performed by the subject with identifier 00)
  - /02_l (images for l gesture of subject 00)
  - /10_down
- /01
- /02
- /09 (last subject with identifier 09)

Every root folder (00, 01, …) contains the infrared images of one subject. The folder name is the identifier of each different subject. [3]

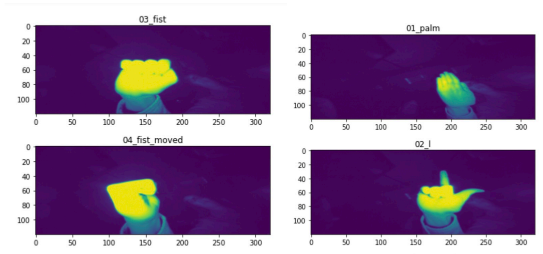


Figure 2-1 dataset

3. Methods and Experiments

As we know, CNNs are often used to solve problems related to spatial data, such as images. RNNs are better suited for analyzing temporal, sequential data, such as text or video.[4] In this project I focus on the model performance on a static gesture dataset so I delved into the architecture of CNNs.

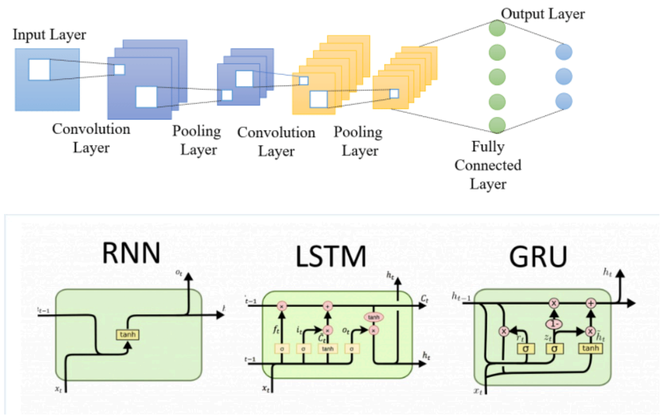


Figure 3-1 Architecture of CNN and RNN

First, image data should be preprocessed so that can be used to train model. Here I used the code for preprocess our dataset including read in and covert image to greyscale, also reshaped it to the correct size to fit in CNN model.[5] I split the dataset into training set (16000 samples), validation set (2000 samples), testing set (2000 samples).

```
print(x_train.shape)
print(x_validate.shape)
print(x_test.shape)
```

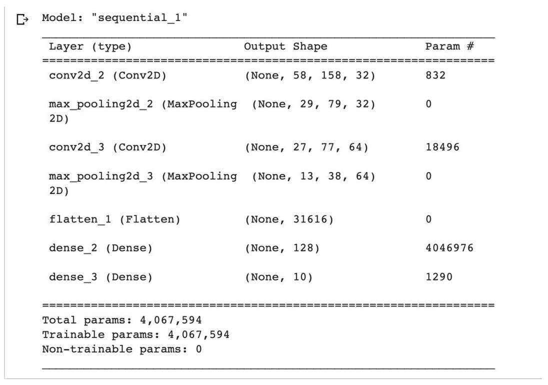
```
(16000, 120, 320, 1)
(2000, 120, 320, 1)
(2000, 120, 320, 1)
```

Figure 3-2 Training, Validation, Testing data

Then I built the base CNN model and trained different CNN models, adding more layers, changing activation function and batch size.

3.1 Base model

My basic convolutional neural network has architecture as follows:

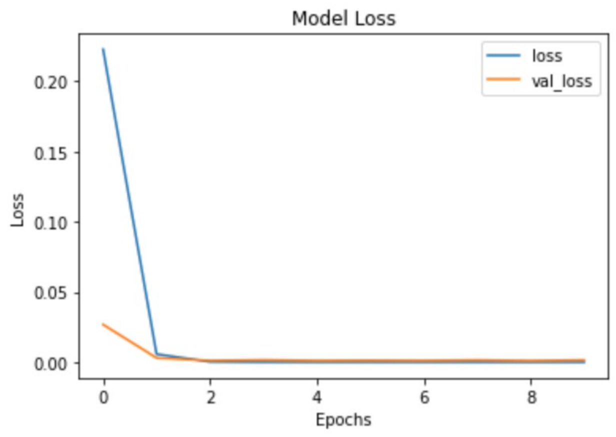


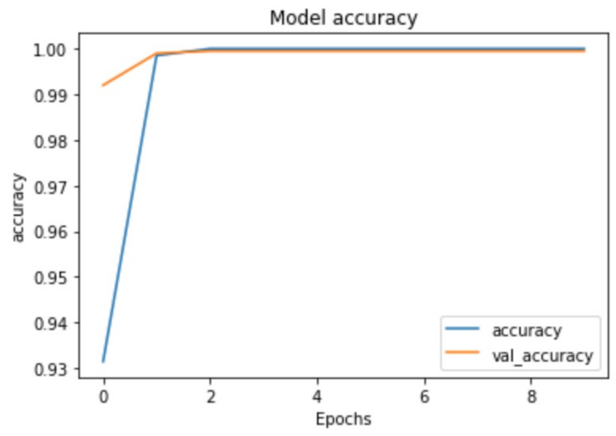
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 58, 158, 32)       832

 max_pooling2d_2 (MaxPooling  (None, 29, 79, 32)       0
 2D)

 conv2d_3 (Conv2D)           (None, 27, 77, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 13, 38, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 31616)             0

 dense_2 (Dense)             (None, 128)               4046976

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 4,067,594
Trainable params: 4,067,594
Non-trainable params: 0
_________________________________________________________________
```

Figure 3-3 Base CNN model

The CNN used ReLU activation function and was trained using the Adam Optimizer. The batch size was 64.

Training results:

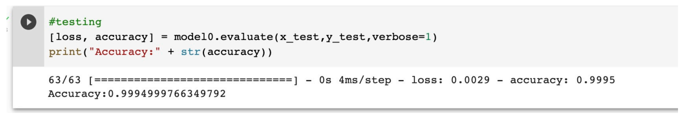




```
Training Accuracy 0.9314374923706055
Validation Accuracy 0.9919999837875366
```

Figure 3-4 training results

Testing results:



```
#testing
[loss, accuracy] = model0.evaluate(x_test,y_test,verbose=1)
print("Accuracy:" + str(accuracy))
```

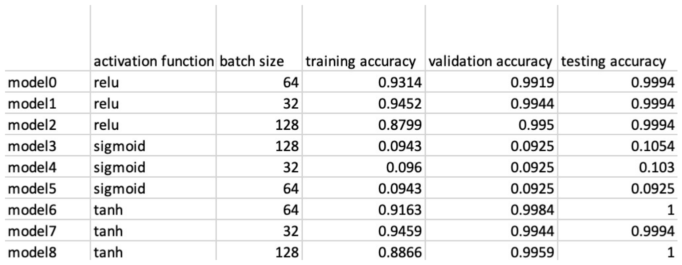
```
63/63 [==============================] - 0s 4ms/step - loss: 0.0029 - accuracy: 0.9995
Accuracy:0.9994999766349792
```

Figure 3-5 testing results

It already can reach almost 1.0 accuracy

3.2 Substitute hyperparameters on the base model

Although the base model already had the perfect performance, I still want to see the model performances after changing some hyperparameters.

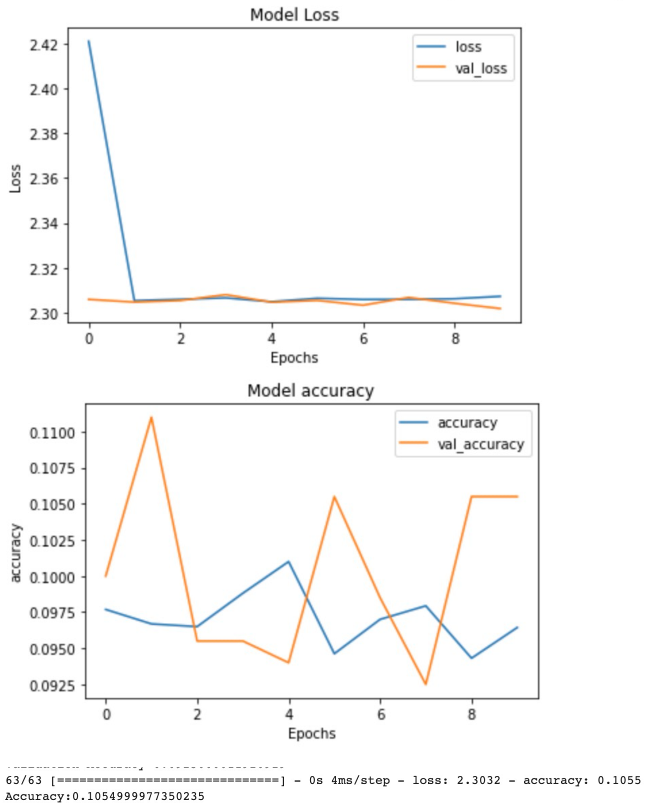


	activation function	batch size	training accuracy	validation accuracy	testing accuracy
model0	relu	64	0.9314	0.9919	0.9994
model1	relu	32	0.9452	0.9944	0.9994
model2	relu	128	0.8799	0.995	0.9994
model3	sigmoid	128	0.0943	0.0925	0.1054
model4	sigmoid	32	0.096	0.0925	0.103
model5	sigmoid	64	0.0943	0.0925	0.0925
model6	tanh	64	0.9163	0.9984	1
model7	tanh	32	0.9459	0.9944	0.9994
model8	tanh	128	0.8866	0.9959	1

Figure 3-6 training and testing results

The above is the results of the model after I replaced some hyperparameters. I also attached all code and results including histograms in GitHub. I didn't increase epochs since models with suitable hyperparameters already had very good performances.

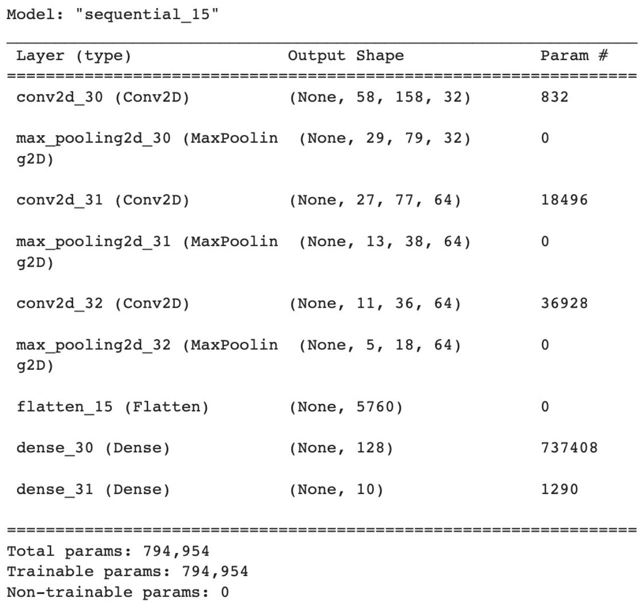
I noticed that CNN models with sigmoid activation function had very low training and testing accuracy.



```
63/63 [==============================] - 0s 4ms/step - loss: 2.3032 - accuracy: 0.1055
Accuracy:0.1054999977350235
```

Figure 3-7 training and testing results

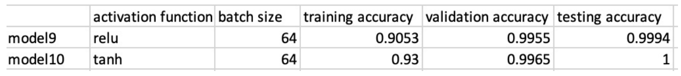
Then I added one more conv2D and max pooling layer to compare performances. The convolutional neural network has architecture as follows:



```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_30 (Conv2D)          (None, 58, 158, 32)       832

 max_pooling2d_30 (MaxPoolin  (None, 29, 79, 32)       0
 g2D)

 conv2d_31 (Conv2D)          (None, 27, 77, 64)        18496

 max_pooling2d_31 (MaxPoolin  (None, 13, 38, 64)       0
 g2D)

 conv2d_32 (Conv2D)          (None, 11, 36, 64)        36928

 max_pooling2d_32 (MaxPoolin  (None, 5, 18, 64)        0
 g2D)

 flatten_15 (Flatten)        (None, 5760)              0

 dense_30 (Dense)            (None, 128)               737408

 dense_31 (Dense)            (None, 10)                1290

=================================================================
Total params: 794,954
Trainable params: 794,954
Non-trainable params: 0
```

Figure 3-8 CNN model

I also trained and tested model with relu and tanh activation function with batch size 64. Here are the training and testing results.



	activation function	batch size	training accuracy	validation accuracy	testing accuracy
model9	relu	64	0.9053	0.9955	0.9994
model10	tanh	64	0.93	0.9965	1

Figure 3-9 training and testing results

4. Observation and Conclusion

As we can see, most models have excellent performances both on training, validation and testing accuracy except when I used sigmoid as activation function. Also, models with 32 or 64 batch size has better performances than those with 128 batch size. The best model in this project is the CNN model with tanh activation function and 64 batch size. Adding one more conv2D and max pooling layer almost didn't change training and testing results. The excellent performances are attributed to the high quality dataset. The images in the dataset I used are very clear and without the interference of background factors. On the one hand, we can easily see the efficiency of the CNN model in static image classification. On the other hand, for other more realistic problems, we need some fuzzy data to test the accuracy of the model. In other words, in simple situations, all models performed well, but complex and realistic situations can distinguish effective models.

5. Challenge and Future work

The biggest challenge I met in this project was to find a high quality dataset. Finally I found the suitable dataset and got CNN models with both high training and testing accuracy. While next step I will want to deal with more realistic problems with more fuzzy data. The background of images especially the light intensity and occlusion problems will hugely influence the accuracy of models. Besides, I will also want to use sequential image dataset so that we can compare the difference of CNN and RNN on dealing with video.

6. Reference

[1] Top 11 Speech Recognition Applications
https://research.aimultiple.com/voice-recognition-applications/

[2] Image Classification
https://huggingface.co/tasks/image-classification

[3] Hand Gesture Recognition Database
https://www.kaggle.com/datasets/gti-upm/leapgestrecog

[4] CNN vs. RNN: How are they different?
https://www.techtarget.com/searchenterpriseai/feature/CNN-vs-RNN-How-they-differ-and-where-they-overlap

[5] Hand Gesture Recognition Database with CNN
https://www.kaggle.com/code/benenharrington/hand-gesture-recognition-database-with-cnn

[6] A Complete Guide to Image Classification in 2022
https://viso.ai/computer-vision/image-classification/

[7] An Overview of Hand Gestures Recognition System Techniques
https://iopscience.iop.org/article/10.1088/1757-899X/99/1/012012

[8] Research on Gesture Recognition Method Based on Computer Vision https://www.researchgate.net/publication/329039460_Research_on_Gesture_Recognition_Method_Based_on_Computer_Vision

[9] Dynamic gesture recognition based on 2D convolutional neural network and feature fusion:
https://www.nature.com/articles/s41598-022-08133-z

[10] A Deep Convolutional Neural Network Approach for Static Hand Gesture Recognitionhttps://www.sciencedirect.com/science/article/pii/S1877050920312473