Gesture Recognition

Atharva Musale - amusale, Radhika Singh - rsingh57 , Swapnil Khare - skhare

April 2024

1 Introduction

Our project focuses on gesture recognition, a field within computer vision and human-computer interaction that aims to interpret human gestures via Deep Learning.

For our study, we have chosen the Jester dataset as it provides a comprehensive collection of hand gesture videos. The Jester dataset is particularly well-suited for our purposes due to the following reasons:

- **Large-scale and Diverse**: The Jester dataset contains a large number of videos featuring diverse hand gestures, providing a rich source of data for training and evaluation of gesture recognition algorithms.
- **Real-world Gestures**: The gestures captured in the Jester dataset represent common human actions and interactions, making it applicable to a wide range of practical applications.
- **Annotated Data**: The dataset comes with pre-defined labels for each gesture, that facilitates supervised learning and evaluation of gesture recognition models.
- **Community Benchmark**: The availability of the Jester dataset has led to it being widely used as a benchmark for evaluating gesture recognition algorithms, allowing for comparison and advancement of research in the field.

By leveraging the Jester dataset, our project aims to develop and evaluate state-of-the-art gesture recognition models that can accurately interpret and classify human gestures in real-time.

2 Dataset Setup

2.1 Dataset Overview

The Jester dataset is a popular dataset used for gesture recognition research. It consists of a large collection of short video clips depicting hand gestures.

- The Jester gesture recognition dataset includes 148,092 labeled video clips of humans performing basic, pre-defined hand gestures in front of a laptop camera or webcam.
- Each sample in the Jester dataset is a short video clip, typically a few seconds long, capturing a person performing a hand gesture. The videos are recorded in various settings and backgrounds, adding diversity to the dataset.
- The dataset covers a wide range of hand gestures, encompassing both simple and complex actions. These gestures may include common hand signals, expressions, or actions typically associated with human interactions. Examples of gestures found in the Jester dataset may include waving, pointing, clapping, thumbs up, thumbs down, and more. The diversity of gestures makes the dataset suitable for training models to recognize a broad spectrum of hand movements
- The clips cover 27 different classes of human hand gestures, split in the ratio of 8:1:1 for training, development and testing. The dataset also includes two "no gesture" classes to help the network distinguish between specific gestures and unknown hand movements.

2.2 Data Preprocessing

The video data is provided as one large TGZ archive, split into parts of 1 GB maximum. The total download size is 22.8 GB. The archive contains directories numbered from 1 to 148092. Each directory corresponds to one video and contains JPG images with a height of 100px and variable width. The JPG images were extracted from the original videos at 12 frames per seconds. The filenames of the JPGs start at 00001.jpg. The number of JPGs varies as the length of the original videos varies.

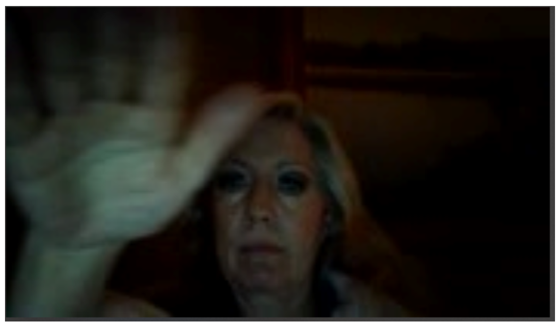


Figure 1: Swipe down

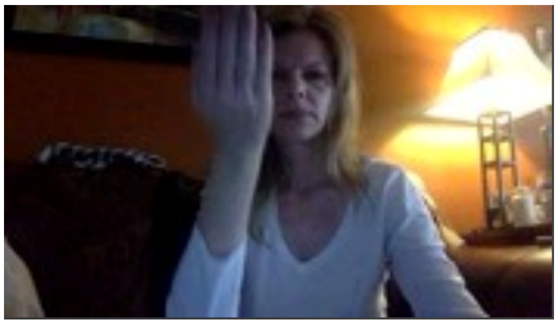


Figure 2: Swipe Up

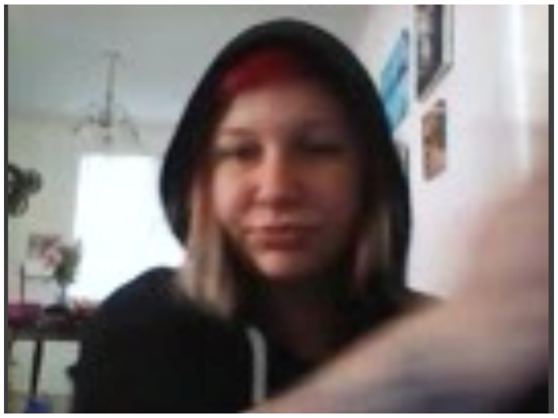


Figure 3: Swipe Left

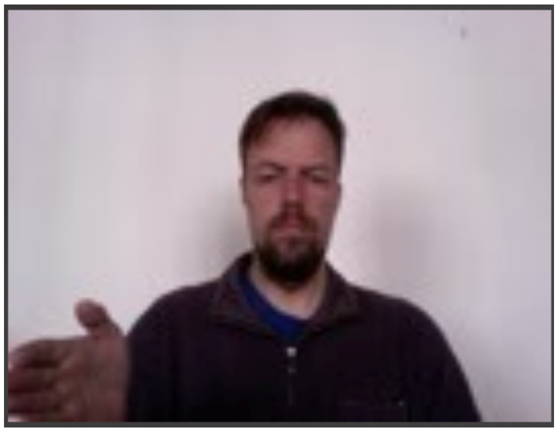
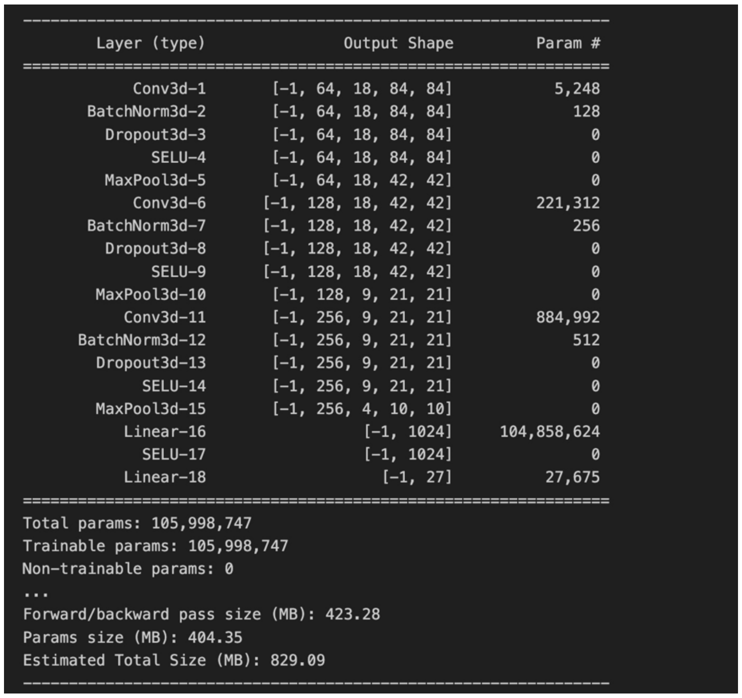


Figure 4: Swipe Right

2.3 Data Split

The dataset consists of 16686 annotations. The dataset is divided into three parts. They are: training, validation and testing dataset. The dataset is divided into 80:10:10.

3 Baseline Model Architecture



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv3d-1       [-1, 64, 18, 84, 84]           5,248
       BatchNorm3d-2       [-1, 64, 18, 84, 84]             128
         Dropout3d-3       [-1, 64, 18, 84, 84]               0
              SELU-4       [-1, 64, 18, 84, 84]               0
         MaxPool3d-5       [-1, 64, 18, 42, 42]               0
            Conv3d-6      [-1, 128, 18, 42, 42]         221,312
       BatchNorm3d-7      [-1, 128, 18, 42, 42]             256
         Dropout3d-8      [-1, 128, 18, 42, 42]               0
              SELU-9      [-1, 128, 18, 42, 42]               0
        MaxPool3d-10       [-1, 128, 9, 21, 21]               0
           Conv3d-11       [-1, 256, 9, 21, 21]         884,992
      BatchNorm3d-12       [-1, 256, 9, 21, 21]             512
        Dropout3d-13       [-1, 256, 9, 21, 21]               0
             SELU-14       [-1, 256, 9, 21, 21]               0
        MaxPool3d-15       [-1, 256, 4, 10, 10]               0
           Linear-16                 [-1, 1024]     104,858,624
             SELU-17                 [-1, 1024]               0
           Linear-18                   [-1, 27]          27,675
================================================================
Total params: 105,998,747
Trainable params: 105,998,747
Non-trainable params: 0
...
Forward/backward pass size (MB): 423.28
Params size (MB): 404.35
Estimated Total Size (MB): 829.09
----------------------------------------------------------------
```

Figure 5: Model Architecture

The explanation of each layer is as belows:

1. **Conv3d** Performs 3D convolution for extracting features from 3D data. The output shape [-1, C_out, D_out, H_out, W_out] represents batch size, number of output channels, depth, height, and width respectively.

2. **BatchNorm3d** Normalizes the activations of the previous layer, stabilizing learning and reducing training time.

3. **Dropout3d** Prevents overfitting by randomly setting a fraction of input units to 0 during training.

4. **SELU** Activation function that helps in faster learning and better performance by maintaining consistent variance across layers.

5. **MaxPool3d** Reduces spatial dimensions by applying a max filter, thus downsampling the input.

6. **Linear** Fully connected layer that combines features for classification or regression, connecting every input neuron to every output neuron.

4 Model Pipeline

The model pipeline consists of seven layers.

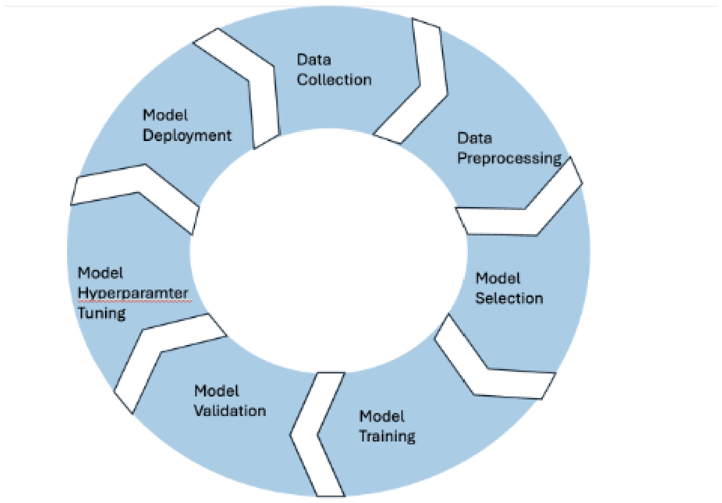


Figure 6: Lifecycle of Gesture Detection Lifecycle

5 Training Pipeline

The training pipeline consists of the steps from model selection, model training, model validation and model hyperparamter tuning.

5.1 Training Process

- *Data Collection* - We have gathered the dataset from the Qualcomm's dataset website. We extract the images from the tar file by using the command "cat 20bn-jester-v1-folder-name — tar zx".

- *Data Preprocessing* - We have performed preprocessing steps like normalization, standarization on the image dataset.

- *Model Selection* - We have made a fully connected three convolutional layers. This is our base model.

- *Model Training* - The model is trained for five classes. The AdamW gradient descent is used.
- *Model Validation* - The 10 percent of the dataset is kept for the calculation of the validation metrics. The validation loss is used as an evaluation metrics.
- *Model Hyperparamter Tuning* -In the next phase of the development, we will ve working on to incorporate changes in the base model by changing the hyperparamter to get less training and validation loss.
- *Model Deployment*- After the completion of the above steps, we are planing to deploy the model by creating a production pipeline (yml file) in the production server.

5.2 Evaluation Metrics

6 Initial Results

The training vs validation loss with respect to epochs is plotted below. The below graph explains that the loss is getting decreased with the increase in the epochs for the training and the validation dataset.

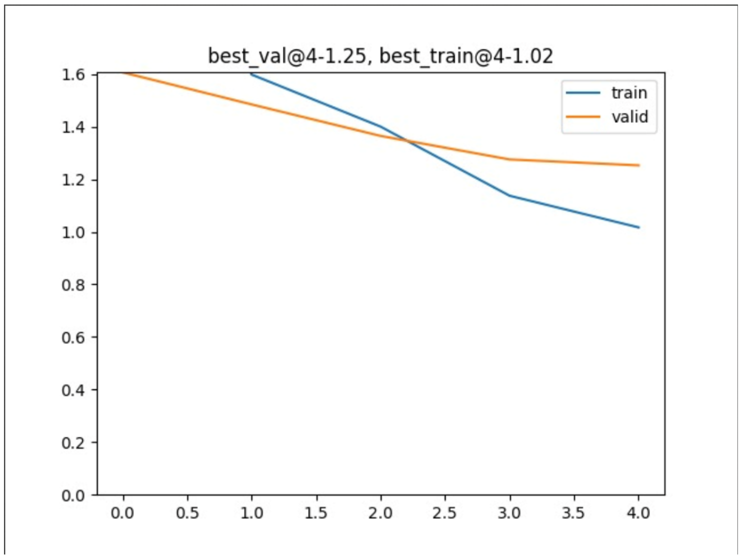


Figure 7: Training and validation loss over epochs.

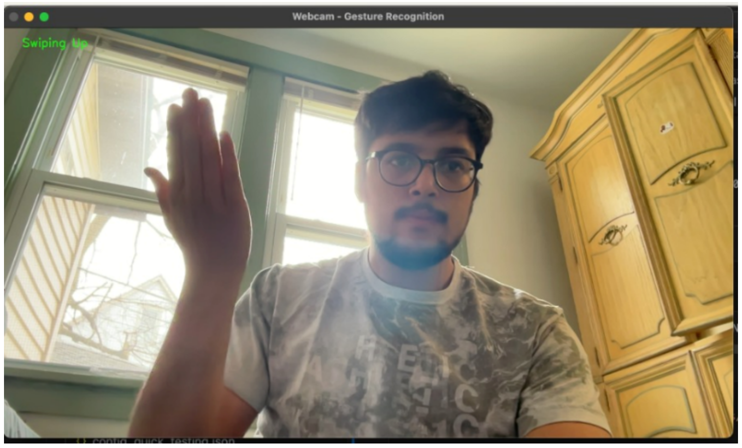


Figure 8: Testing

In training our model to recognize various gestures, including Swiping Left, Swiping Right, Swiping Down, Swiping Up, and Doing Other Things. Among these, the gesture for Swiping Up was recognized with the highest accuracy.

Firstly, it seemed that Swiping Up, as a gesture, might have characteristics that are more distinct or consistent, making it easier for our model to identify. This could be due to the motion patterns or spatial features unique to this gesture, which our 3D convolutional layers could capture more effectively.

It could be that the dataset for Swiping Up was richer or of better quality compared to the others.

The architecture of our model could also be playing a role. It's possible that the way our layers are structured, and the parameters we've chosen, are particularly well-suited to picking up the features of the Swiping Up gesture.

To address the discrepancies in gesture recognition, We are considering several strategies. This includes augmenting our dataset, especially for the gestures that are lagging behind in accuracy, and experimenting with tweaks in our model's architecture to see if we can improve its ability to capture the nuances of each gesture more effectively.

7 Discussion and Next Steps

Following the creation of the baseline model, several avenues for further exploration and improvement can be pursued. These include:

- **Training on all classes and complete dataset**: As a part of baseline model, we have traine d our model on only 5 classes which include -

Swiping Left Swiping Right, Swiping Down, Swiping Up, Doing other thing. Complete training on all the classes will include 27 classes,

- **Model Optimization**: Fine-tuning hyperparameters, experimenting with different architectures, or incorporating advanced techniques such as attention mechanisms or spatial-temporal modeling to enhance the performance of the gesture recognition model.
- **Data Augmentation**: Exploring techniques such as augmentation of the training data to increase its diversity and robustness, thereby improving the model's generalization capability.
- **Transfer Learning**: Investigating the applicability of transfer learning from pre-trained models on larger datasets to improve the performance of the gesture recognition model on the Jester dataset.
- **Real-time Inference**: Optimizing the model for real-time inference to enable efficient deployment in practical applications, such as interactive systems or assistive technologies.
- **User Interface Integration**: Integrating the gesture recognition model into user interfaces or interactive applications to enable intuitive and natural human-computer interaction.

These next steps aim to build upon the baseline model and address potential limitations, improving the precision of gesture recognition.

8 Conclusion

In conclusion, the baseline model developed using the Jester dataset represents a foundational step towards achieving accurate and reliable gesture recognition. Leveraging the rich source of hand gesture videos provided by the dataset, the model demonstrates the feasibility of interpreting and classifying human gestures in real-time. Moving forward, we will focus on refining the model, exploring advanced techniques, and integrating gesture recognition into practical applications to enhance human-computer interaction.

9 References

- https://developer.qualcomm.com/software/ai-datasets/jester
- https://github.com/udacity/CVND—Gesture-Recognition
- https://github.com/sukhijapiyush/Gesture-Recognition-using-the-CNN
- https://github.com/shikhinmehrotra/Deep-Learning-CNN-based-hand-gesture-recognition

- https://www.nature.com/articles/s41598-022-08133-z
- https://developer.qualcomm.com/software/ai-datasets/jester
- https://docs.python.org/3/library/os.html
- https://docs.python.org/3/library/sys.html
- https://docs.python.org/3/library/shutil.html
- https://docs.python.org/3/library/glob.html
- https://docs.python.org/3/library/pickle.html
- https://github.com/tqdm/tqdm