Final Report

Dynamic Hand Gesture Recognition

A.R. Shrivastava[1], P. Gupta [2] and S. Sharma[3]

[1] *B21AI003, AI&DS, Department of CSE, IIT Jodhpur*

[2] *B21AI027, AI&DS, Department of CSE, IIT Jodhpur*

[3] *B21CS066, CS, Department of CSE, IIT Jodhpur*

SUMMARY: Our project focuses on creating a streamlined deep learning model for Hand Gesture Recognition (HGR) to facilitate computer control. By employing advanced techniques like CNNs, RNNs, and transformers, we aim to enable real-time, hands-free interactions with digital devices. Our goal is to contribute to a more intuitive and accessible computing experience, impacting areas such as virtual reality, gaming, healthcare, and accessibility.

1. INTRODUCTION

In our rapidly advancing technological landscape, the fusion of artificial intelligence and human-computer interaction has become increasingly essential. Our project endeavors to contribute to this intersection by focusing on the development of a sophisticated deep learning model for dy Hand Gesture Recognition (HGR), aimed at enhancing computer control interfaces. Hand gestures, as a natural and intuitive form of communication, offer a promising avenue for more seamless interactions with digital devices.

The primary goal of our project is to create an efficient and accurate deep learning model capable of recognizing and interpreting diverse hand gestures to facilitate control of computers. By leveraging state-of-the-art deep learning techniques, we aim to empower users with a more intuitive and hands-free computing experience. This technology holds significant potential for applications in diverse fields, ranging from virtual reality and gaming to healthcare and accessibility.

Our approach involves harnessing the power of convolutional neural networks (CNNs), transformers and recurrent neural networks (RNNs) to process and understand the dynamic nature of hand gestures. We will explore datasets containing a wide variety of hand gestures, ensuring the model's robustness and adaptability across different user scenarios. Through meticulous design and training, we aspire to create a model capable of real-time gesture recognition, allowing users to control computers effortlessly through natural hand movements.

Our project's success will not only hinge on the precision of gesture recognition, but also its feasibility, promptness, and appeal to users. Our goal is to advance interactive computing, creating a more seamless and user-friendly experience for people in all fields.

2. Codebase

LINK - Colab

3. Dataset

3.1. Jester Dataset

3.1.1. Introduction

Our project leverages the Qualcomm Jester Gesture Recognition dataset (1) to train and evaluate our deep learning model for Hand Gesture Recognition (HGR). The Jester dataset comprises 148,092 labeled video clips featuring humans performing basic, predefined hand gestures in front of a camera.

3.1.2. Dataset Details:

The tabular details of dataset is shown in *Table 1.* The dataset covers 27 different classes of human hand gestures, such as swiping left or right, thumb up or down, and turning hands clockwise or counterclockwise. To ensure model robustness, the dataset in-

Table 1: Dataset Details

Total Number of Videos	148,092
Training Set	118,562
Validation Set	14,787
Test Set (w/o labels)	14,743
Labels	27
Quality	100px
FPS	12

Fig. 1: Jester Dataset

cludes two "no gesture" classes, differentiating specific gestures from unknown hand movements.

3.2. Custom Dataset

3.2.1. Introduction

The creation of a custom dataset provides a unique opportunity to delve into various data processing techniques. This includes preprocessing steps, normalization, and handling of specific data characteristics that are essential for our problem domain. Exploring data processing techniques involves investigating methods such as data augmentation, feature engineering, and the application of domain-specific transformations to enhance the quality and informativeness of the dataset.

3.2.2. Dataset Details:

The dataset is created from webcam of `LENOVO Ideapad 3 15ADA05`, WebCam: `480p 4:3 30fps` The dataset is available at https://drive.google.com/drive/folders/1TN6Rwt_I_VUzJoLpquba4bOkF5fS1VeS?usp=drive_link

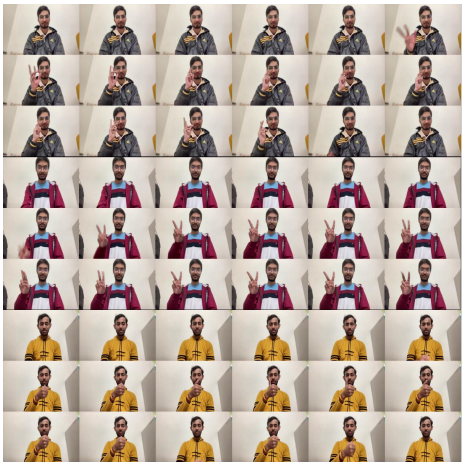


Fig. 2: Custom Dataset

4. Solution Strategy

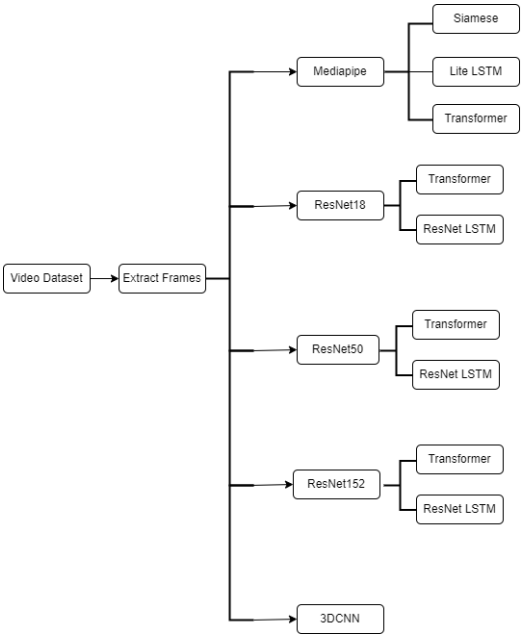
4.1. Assumptions

- Tensorflow uses Static Graphs for computation which were performing better than Pytorch Dynamic Graph in the dataset. So for some models we used Pytorch and for some we used Tensorflow
- As the dataset dimension is too low, the mediapipe is not working as expected due to this, so we also used ResNet for create embeddings
- Tensorflow provided production grade models, where as Pytorch provided development focused models

4.2. Motivation towards the strategy

We thought of two strategies to encounter this problem which are as follows:

- Use mediapipe and resnet encodings to train different models like LSTMs, transformers etc. on top of that.
- Take the extracted frames directly for training the models like 3D CNN.



4.3. Lite weight LSTM

In our exploration of temporal data, we began by implementing RNN models as our primary approach. Initially, we experimented with a lightweight LSTM model, which served as a basic foundation. Our preprocessing involved extracting key points from both hands using Mediapipe.

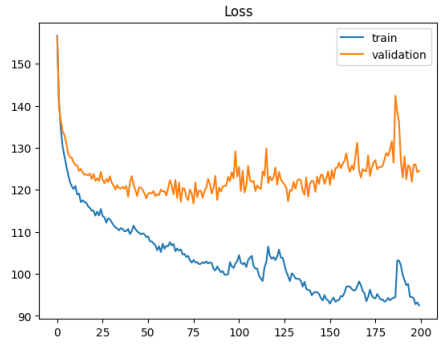
To establish a baseline accuracy, we employed the Cross Entropy loss function. Subsequently, we devised a custom loss function that assigns higher weights to the last 12 frames of each sequence, as opposed to considering the output of last frame only. This adjustment proved crucial in enhancing the model's accuracy.

The total loss (total_loss) is calculated using the following formula:

$$\text{total\_loss} = \sum_{i=0}^{24} \text{loss\_1}_i \times e^{\frac{i}{37}} + \sum_{i=0}^{11} \text{loss\_2}_i \times e^{\frac{i+25}{37}}$$

where:

- total_loss is the total loss calculated using the provided formula.
- loss_1_i is the cross-entropy loss of the i-th image in the first set of 25 images.
- loss_2_i is the cross-entropy loss of the i-th image in the last set of 12 images.
- e is the base of the natural logarithm, approximately equal to 2.71828.



4.4. ResNet LSTM

To address the limitations of using Mediapipe due to the small dimension of images in the input dataset, we opted to utilize the ResNet model for extracting embeddings.

We experimented with three variants of the ResNet model: ResNet18, ResNet50, and ResNet152. For ResNet152, we took initial 50% of the layers on ResNet152 and finetuned last layers of it. For fine-tuning, we added an adaptive pooling layer and fully connected layer to generate embeddings of a specific length, we named it as ResNet152_per50.

Let E_i represent the embeddings extracted from the ResNet model for the ith video in the dataset which will be of shape (37, 512). These embeddings were then used as input to the LSTM model for further processing and classification.

4.5. Siamese LSTM model

The architecture of a Siamese LSTM model (2) is similar to the Lite Weight LSTM model. The only difference is in the loss function. Siamese LSTM models use contrastive loss function. Formula for contrastive loss is given as follows:

$$L(W, Y, X_1, X_2) = (1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}(\max(0, m - D_W))^2$$

where

$$D_W(X_1, X_2) = ||G_W(X_1) - G_W(X_2)||_2$$

$$Y = \begin{cases} 0, & \text{if } Y_1 = Y_2. \\ 1, & \text{otherwise.} \end{cases}$$

Y is either 0 or 1 depending on whether we are comparing similar or dissimilar items. In our case, if we are comparing a hand gesture of "Swiping down" to another hand gesture of "Swiping down", Y will be 0, otherwise, if we are comparing the hand gesture of "Swiping down" to a different hand gesture, then Y will be 1. D_w above refers to the Euclidean

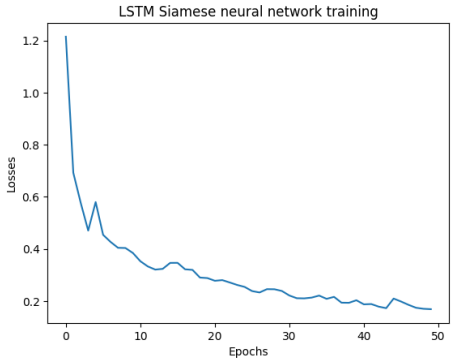


Fig. 3: Contrastive loss on mediapipe encodings

distance between 2 vectors, i.e. when the model processes 2 videos, both are converted into n-dimensional vectors. The closer the distance between 2 vectors, the more likely both gestures are same and will yield a closer Euclidean distance as opposed to a digit of 1 compared with 0, where vectors borne from different digits would yield a larger Euclidean distance.

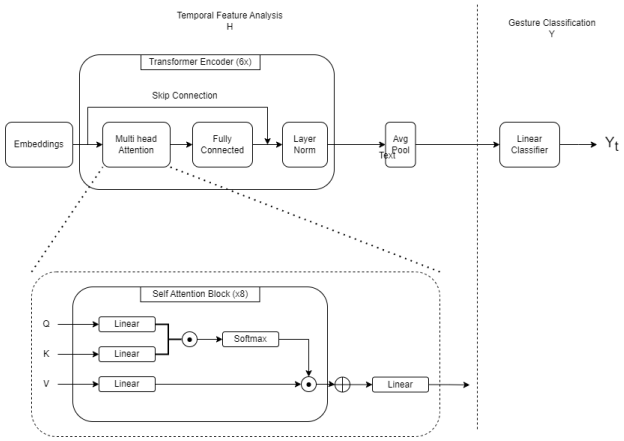
Therefore, the loss will be less when the vectors of the same gesture have low Euclidean distance while vectors of different gestures have greater Euclidean distance.

Since the number of classes is finite, we can use contrastive loss to make our model distinguish between videos of different gestures. In the prediction part, it takes the test frames/video, processes its vector and compares it with the representation vectors of gestures from all the classes (one from each). These images are chosen randomly with the restriction of one image per class. The class, whose image has the least Euclidean distance with the test vector will be the predicted label of the given gesture video.

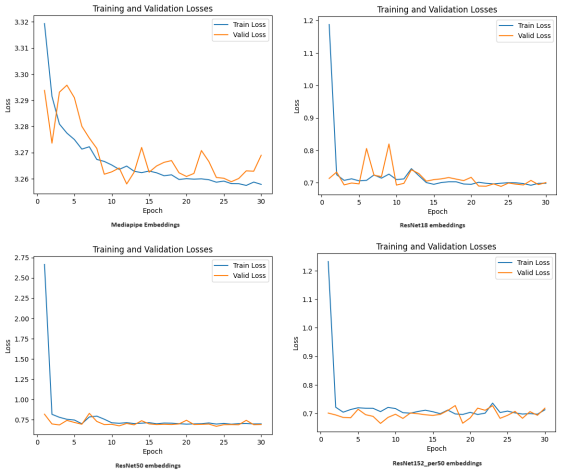
One of the drawbacks of Siamese model is that it takes much more time to predict in comparison to LightWeight LSTM. The prediction time depends on the number of classes in the dataset. For eg: If there are n classes in the dataset, Siamese will take time n times greater than the lightweight LSTM.

4.6. Transformer

To enhance the embeddings generated by the pretrained model and capture temporal features for hand gesture prediction, we utilize a Transformer-based architecture (3). The architecture consists of the following components:



Training and Validation Loss on 4 different types of embeddings, that includes Mediapipe, ResNet18, ResNet50, ResNet152_per50



4.7. 3D CNN

To address the limitations of the embeddings generated by Mediapipe and ResNet models, we explored the use of 3D convolutional neural networks (CNNs) (4) (5). However, due to the enormous size of the 3D CNN model and the significant time required for training, we were only able to train it on a limited subset of the dataset comprising 100 videos.

Unfortunately, the results obtained from this limited training data were not promising. The model exhibited signs of overfitting, likely due to the insufficient training data and the complexity of the model architecture. As a result, the performance of the 3D CNN in classifying hand gestures did not meet our expectations.

While 3D CNNs have shown promise in capturing spatiotemporal features from video data, their effectiveness depends heavily on the availability of large and diverse training datasets. In our case, the limited training data hindered the ability of the 3D CNN to generalize well to unseen examples, leading to suboptimal performance.

5. Major innovations

5.1. Current Works

The majority of recent research in hand gesture recognition has focused on static gestures. This includes approaches that utilize convolutional neural network (CNN) models to predict hand gestures based on static images. CNN-based architectures, such as ResNet, VGG, and MobileNet, are widely adopted due to their ability to extract spatial features from hand images effectively.

Our aim is to shift the focus towards dynamic hand gesture recognition, where the temporal evolution of hand movements over time is analyzed and classified.

5.2. Innovation

In the context of innovation, to perform dynamic gesture recognition we did following:

1. **Custom Loss Function in Lite LSTM**: We introduced a custom loss function used in Lite LSTM model to improve its performance. By doing either polynomial or expoential weighted average of the loss from the 37 frames output by the model, we were able to improve the performance by 30% on test data.

2. **ResNet over traditional Mediapipe**: We extracted embeddings from three different variants of the ResNet model: ResNet18, ResNet50, and a modified ResNet152 with 50% of its layers and fine tune over dataset. These embeddings were then used as input to a LSTM and a transformer model to predict the label.

3. **Siamese Architecture**: We employed a Siamese architecture to incorporate temporal information into our model. By processing pairs of video frames and learning to distinguish between similar and dissimilar frames, the Siamese architecture facilitated the modeling of temporal dependencies and contributed to improved accuracy in dynamic hand gesture recognition.

4. ***Optimizing extracted keypoints from Mediapipe**: To further improve the accuracy of keypoints obtained through Mediapipe, we propose a simple yet effective method. By considering the maximum pixel value among 3-4 consecutive frames for each keypoint, we aim to reduce dataset size while minimizing errors caused by blurry frames. This approach selects the most prominent features across multiple frames, enhancing the reliability of the keypoints. Ultimately, this method optimizes hand gesture recognition systems by improving accuracy and efficiency, especially in real-world scenarios with environmental variability and noise.

$*$ => Future Scope

6. Results

Embeddings	Architecture	Accuracy
Mediapipe	Siamese	13.05
	Lite LSTM CE Loss	32
	Lite LSTM Custom Loss	62
	Transformer	3.47
ResNet18	Transformer	44.23
	ResNet LSTM	44
ResNet50	Transformer	48.08
	ResNet LSTM	80
ResNet152_per50*	Transformer	55.77
	ResNet LSTM	10
None	3DCNN	< 10

Table 2: Accuracy of different models using various architectures.

$*$ $\rightarrow$ ResNet152 with 50% of its layers

7. Analysis and Problems Faced

- **Effectiveness of Custom Loss Function:** The introduction of a custom loss function in the Lite LSTM model improved the accuracy to a great extent. By assigning higher weights to the last 12 frames of each sequence, we achieved a 30% increase in accuracy on the test data compared to using a standard cross-entropy loss function applying on last layer.

- **ResNet Outperforms Mediapipe:** Utilizing ResNet for extracting embeddings proved more robust compared to Mediapipe, especially considering the challenges posed by blurry images in the Mediapipe. ResNet embeddings resulted in improved accuracy across various models and architectures.

- **Siamese Architecture for Temporal Dependency:** The Siamese architecture demonstrated the ability to capture temporal dependencies effectively. By distinguishing between similar and dissimilar frames, the Siamese model contributed to enhanced accuracy in dynamic hand gesture recognition.

- **Non robustness of Mediapipe** The Jester dataset is created by individuals themselves, which means it contains various types of images, including some that are blurry and size of the images is also very small (around 100 pixels). Unfortunately, this blurriness poses a challenge for Mediapipe in accurately extracting hand keypoints from those images. Consequently, when we utilized Mediapipe embeddings as the training dataset for our models, we encountered significantly reduced accuracy.

- **CUDA memory exceeded in 3D CNN** During the training process of our 3D Convolutional Neural Network (CNN), a significant challenge

arose due to its considerable size, rendering it incapable of fitting within the memory constraints of the available GPUs, both online and offline. Consequently, we were compelled to restrict the training dataset to just 100 videos. This reduction in dataset size led to a notable issue of overfitting, primarily stemming from the limited representation of each class with only 3-4 videos available for training. This scarcity of data per class hindered the model's ability to generalize effectively, resulting in an overemphasis on the specific characteristics of the training samples and a diminished performance when presented with unseen data.

- **Overfitting on Custom Dataset**

  Initially, we attempted to train our models using a custom dataset tailored to our specific needs. However, due to the limited size of this dataset, our models tended to overfit. In response to this challenge, we opted to switch to the Jester dataset for improved model training. This larger and more diverse dataset provided a richer training experience, enabling our models to better generalize across different gesture categories. As a result, we observed a significant improvement in our models' performance compared to when using the smaller custom dataset.

8. Conclusion

Our findings demonstrate the effectiveness of custom loss functions, such as the one implemented in the Lite LSTM model, in improving accuracy by capturing temporal dependencies more effectively. Additionally, the superiority of ResNet embeddings over traditional methods like Mediapipe highlights the importance of robust feature extraction in achieving accurate gesture recognition.

The utilization of innovative architectures like the Siamese model further underscores the importance of considering contrastive loss while training so that model can differentiate well between different classes.

In summary, our project contributes to the growing body of research in dynamic hand gesture recognition (HGR) and lays the foundation for further advancements in this field. By leveraging the power of deep learning, we aim to create more intuitive and accessible computing experiences that empower users across diverse domains.

9. Resources

1. Nvidia A100 40 GB
2. Nvidia A5000 40GB
3. Nvidia P100 16GB
4. Nvidia 2xT4 16GB
5. Total training time 20 hours

10. Approach used

1. **Caviar approach** - In this approach, we train lots of models and pick the best one.
2. **Pandas approach** - In this approach, we train a single model and tune it to get the best performance.
3. We used the **Caviar approach** as all the research papers were using the **Pandas approach**, and we wanted to produce a unique approach.

REFERENCES

[1] Jester gesture recognition dataset. Qualcomm. [Online]. Available: https://developer.qualcomm.com/software/ai-datasets/jester-gesture-recognition

[2] K. Teo. (2018, Jul.) Long short-term memory (lstm) neural networks as an alternative to convolutional neural networks. Medium. [Online]. Available: https://medium.com/@kelvinteokelvinteo/long-short-term-memory-lstm-neural-networks-as-an-alternative-to-convolutional-neural-networks-24e5e118d6a1

[3] A. D'Eusanio, A. Simoni, S. Pini, G. Borghi, R. Vezzani, and R. Cucchiara, "A transformer-based network for dynamic hand gesture recognition," in *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020, pp. 623–632.

[4] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*.

[5] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, "Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4207–4215.