Recognition of Dynamic Hand Gestures for the EMMA Research Platform

Project Work

Systems Neuroscience & Neurotechnology Unit

Saarland University of Applied Sciences
Faculty of Engineering

Submitted by: Madhava Reddy Pesala

Matriculation Number: 5000804

Course of Study: Neural Engineering, MSc

Supervisor: Benedikt Buchheit, M.Eng.

Abstract

This project dives into recognizing hand movements especially moving gestures using advanced deep-learning methods. As people look for more natural and intuitive ways to interact with computers and automobiles, it's becoming crucial to precisely understand hand gestures. These gestures can be used in every situation between humans and machines, from automobiles to helping those with disabilities, but it requires specialized hardware, such as cameras, and sufficient computational power to minimize the delay between performing and recognizing the gestures. The core of our project is a well-performing neural network designed to understand and recognize different hand gestures from a set of video sequences and more importantly, the quality of the dataset we use to train plays a major role. The initial process is to define meaningful and necessary state-of-the-art gestures for extensive human-machine interaction and reach out to publicly available datasets that closely align with defined gestures. We enhanced our dataset with varied techniques, used a top-notch network model, and followed a detailed training plan. Initial tests show encouraging results, matching the performance of many leading models in this field. This report takes a deep dive into the steps we took, our methods, and our findings, exciting possibilities of using deep learning for understanding hand gestures. This project is a part of the EMMA research platform – an HMI research pool utilized by the SNN-Unit and is the extension of the static gesture control.

Declaration

I hereby declare that I have authored this work independently, that I have not used other than the declared sources and resources, and that I have explicitly marked all material that has been quoted either literally or by content from the used sources. This work has neither been submitted to any audit institution nor been published in its current form.

Saarbrücken, September 13, 2023

Madhava Reddy Pesala

Contents:

1 Introduction

Hand gestures have always played a major role in human-computer interaction, serving as a natural and intuitive mode of communication. Today, with the advanced technology of having great computational resources and many deep learning pre-trained models and abundant data available publicly, we have the ability to train, recognize and interpret large number of gestures with good accuracy, applications ranging from augmented reality experiences and gaming to assistive technologies for the differently-abled by reducing hardware dependency when compared to conventional Human Machine Interactive Systems. Gesture recognition, especially in this project, dynamic hand gesture recognition, involves tracking the movement of the hand over time and recognition of predefined movements of the hand, called dynamic gestures.

Deep learning, a subset of machine learning, has revolutionized the field of computer vision in the past decade. By neural networks with multiple layers, deep learning models can automatically learn hierarchical representations from raw data, making them particularly suitable for tasks such as image and video recognition. When applied to the domain of hand gesture recognition, deep learning techniques have shown significant promise in enhancing accuracy and robustness against various challenges such as diverse backgrounds, lighting conditions, and occlusions.

The objective of this project is to identify and list state-of-the-art hand gestures utilized in automobiles, source publicly available datasets that closely align with these identified gestures, preprocess and format the data to extract hand landmarks using EMMAeye to make it compatible with the EMMA system, and subsequently design a model capable of effectively recognizing and classifying dynamic hand gestures in real-time.

2 State-of-the-art

Dynamic hand gesture recognition is an evolving field in computer vision and human-computer interaction. State-of-the-art in this domain has significantly improved over the past few years, with better improvements in deep learning, better datasets, and improved computing power. The state-of-the-art dynamic hand gesture datasets that are available publicly are:

Datasets:

➢ DHG-14/28 dataset: Contains 14 gestures and two subsets based on the number of gesture phases, but has depth video sequences.
URL: http://www-rech.telecom-lille.fr/DHGdataset/

➢ SHREC'17 dataset: Contains 14 gestures performed with a hand's full articulation, but has depth and skeletal dataset
URL: http://www-rech.telecom-lille.fr/shrec2017-hand/.

➢ Nvidia's Dynamic Hand Gesture dataset: Contains 25 gestures collected from 20 subjects in an automotive setting captured with SoftKinetic DS325 sensor that provided both color and depth video sequences.

https://research.nvidia.com/publication/2016-06_online-detection-and-classification-dynamic-hand-gestures-recurrent-3d

Deep Learning Models:

- ➢ 3D Convolutional Neural Networks (3D CNNs): These models process temporal sequences of spatial frames, making them suitable for dynamic gesture recognition.
- ➢ Two-Stream CNNs: Process spatial and temporal streams separately and then fuse the information.
- ➢ CNN combined with RNN/LSTM: CNNs handle the spatial features, while RNNs or LSTMs handle the temporal dynamics.
- ➢ Transformers: Although initially designed for natural language processing tasks, Transformers have been adapted for vision tasks, and their self-attention mechanisms can be useful for sequence data like dynamic gestures.

Future research is likely to focus on improving accuracy, ensuring real-time processing, handling a wider variety of gestures, and expanding the applications of dynamic hand gesture recognition.

3 Method

The quality of a dataset is important for the success of any machine learning or deep learning project. In this dynamic hand gesture recognition, the dataset's quality directly influences the model's ability to accurately and consistently recognize gestures in diverse real-world scenarios. High-quality datasets contain a broad spectrum of variations, ensuring that the trained model is robust, adaptable, and less prone to errors.

3.1 Collecting State-of-the-Art Gestures

In the rapidly evolving technology, state-of-the-art dynamic hand gestures often set the benchmark for user interactions. For our project, prioritized capturing these cutting-edge gestures, specifically those that have found applications in the automotive industry. The automotive sector has increasingly integrated gesture-based controls, offering drivers and passengers a seamless and advanced user experience. By focusing on these gestures, aimed to align the dataset with industry standards and address real-world applications directly. The selection of these gestures was based on their usage in modern vehicles, ensuring that our dataset remains relevant.

Detailed List of Collected Gestures:

	Gesture	**Description**	**Contextual information**	**Automobile applications**
G1	Palm swiping left	Swiping towards left, right, down, and up hand movements to navigate screens	Used for scrolling through options or lists	BMW, Audi, Volkswagen, Jaguar, Ford
G2	Palm swiping right			
G3	Palm swiping up			

G4	Palm swiping down			
G5	CW Rotation	Circular hand movement to adjust audio volume or settings	Used for controlling volume	BMW, Ford
G6	CCW Rotation			
G7	Pointing gesture	Extending index finger to select an item or option	Accepting	BMW, Volkswagen, Ford, Mercedes
G8	Two-finger peace gesture	Extending index and middle fingers in a peace sign gesture	Used for answering or rejecting phone calls	BMW, Mercedes Benz, Ford
G9	Thumb-pointing left gesture	Extending the thumb and pointing it toward the left side	Used for indicating forward	BMW
G10	Thumb-pointing right gesture	Extending the thumb and pointing it toward the right side	Used for indicating backward	BMW
G11	Pinching gesture	Bringing thumb and index finger together to zoom in or out	Used for zooming in or out on maps or screens	BMW
G12	Palm wave gesture	Open hand movement to accept or dismiss notifications	Used for accepting or dismissing notifications	*suggested by pesala*
G13	Thumb-up gesture	Thumb raised as a sign of approval or liking	Used for approving or liking a feature	*suggested by pesala*
G14	Thumb-down gesture	Thumb pointed downwards to indicate disapproval	Used for rejecting or disliking a feature	*suggested by pesala*
G15	Pushing gesture	Forward hand movement to confirm a selection or action	Used for confirming choices or actions	*suggested by pesala*

G16	Pulling gesture	Backward hand movement to cancel or undo an action	Used for canceling or undoing previous actions	*suggested by pesala*

Dataset Collection for Selective Gestures Having identified the state-of-the-art dynamic hand gestures pivotal to our project, the step involves choosing selective gestures depending on our project needs. Given the expansive nature of available data and the precision required for our project, it's tough to select specific gestures prior, to ensuring that the collected data aligns seamlessly with our objectives. To optimize the data collection process and minimize both time and effort, the next step is to approach to explore the available datasets. A comprehensive search on platforms such as Google and specialized dataset repositories will be done. The aim is to find the datasets that closely match our requirements. If we encounter datasets that match our criteria in terms of gesture type, quality, and diversity, leveraging these existing resources can provide a significant head-start. This not only minimizes the project timeline but also ensures that we're building upon tried and tested datasets. However, if the found datasets do not provide the desired results, or if there are gaps in the available data, we are prepared to supplement with our own data collection efforts to ensure comprehensiveness.

3.2 Data Collection

During the online search for datasets, found three resources as stated in the ' State of the Art ’ section. Unfortunately, two of the datasets did not match with our specific project requirements. ‘DHG 14/28’ and ‘SHREC17’ datasets contain depth image frames which isn’t possible to extract hand landmarks as EMMA eye’s mediapipe framework only works for color image frames. However, a dataset that is available on Nvidia's official website matches our requirements. In total, 11 gesture classes from the requirement list matched with Nvidia’s 25 distinct gesture classes. This data was collected from a group of 20 subjects, by being conducted indoors within a car simulator. This ensures that the gestures are reflective of actual in-car interactions. SoftKinetic DS325 sensor is used to acquire front view color and depth videos and a top-mounted DUO 3D sensor to record a pair of stereo-IR streams This multi-faceted recording method provides left and right IR videos, color, and depth videos. Each gesture captured in a video segment lasts 8 seconds. The specifics and enumeration of gestures can be found in the preprocessing section.

3.3 Data Selection

Given that the Nvidia dataset closely matches our project's requirements, we decided to use it, thereby reducing time and effort. However, even with high-quality datasets, clear checking is crucial to ensure the highest standards of results. The Nvidia dataset offers IR, color, and depth video segments for every gesture sample. For our specific use case, since the Mediapipe framework primarily detects hands from segments with color information, extracted only the color video segments from the dataset. It was observed that there was a similarity between few gesture classes and two gesture classes which are static to be removed as we are focussing only on dynamic gestures. To streamline our model and avoid potential bad results during recognition, merging these similar classes is performed. From the initial 25 classes, it is now reduced to 17 distinct gesture classes. The merged gesture classes exhibit slight variations, yet they are very similar in action.

Comprehensive list of gestures:

	Initial gestures	**Corresponding gesture in State of the Art gestures list**	**Renamed Gestures after merging**	**Final No. of Gestures**
1	Palm swiping left	G1	Horizontal swiping	1
2	Palm swiping right	G2		
3	Palm swiping up	G3	Swiping Up	2
4	Palm Swiping up(2)	G3		
5	Palm swiping down	G4	Swiping down	3
6	Palm swiping down(2)	G4		
7	V - Swiping left	Doesn't match	V - Swiping left	4
8	V - Swiping right	Doesn't match	V - Swiping right	5
9	V - Swiping up	Doesn't match	V - Swiping up	6
10	V - Swiping down	Doesn't match	V - Swiping down	7
11	Pointing	G7	Pointing	8
12	Click	G7		
13	Pulling	G16	Pulling	9
14	Pulling	G16		
16	Palm shake	G12	Palm shake	11
17	Peace sign	G8	Peace sign	12
18	Peace sign	G8		
15	Palm Opening	Doesn't match	Palm Opening	10
19	Three fingers open	Doesn't match	Three fingers open	13
20	CW Rotation	G5	CW Rotation	14
21	CCW Rotation	G6	CCW Rotation	15
22	Pushing	G15	Pushing	16
23	Five fingers closure	Doesn't match	Five fingers closure	17

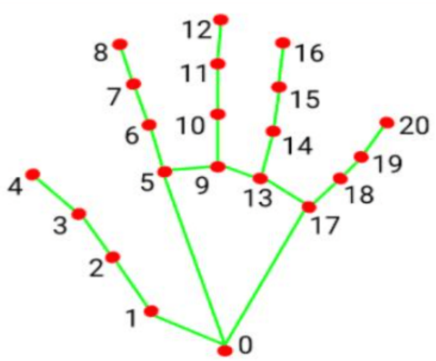
24	Thumbs up	G13	***removed*** (static gesture)
25	OK sign	Doesn't match	***removed*** (static gesture)

3.4 Preprocessing

Within each 8-second video segment of our dataset, the actual gesture performance averages around 2 seconds in duration. This indicates that a significant portion of each video, approximately 6 seconds, doesn't contribute meaningful gesture data. Such lengthy non-gesture segments can introduce noise, leading to potential misclassifications and negatively impacting the model's predictive performance. To enhance the clarity and relevance of our dataset, it's essential to trim these video segments, focusing primarily on the video frames that have hands detected. By using EMMA eye, which was developed by SNN-Unit and is based on python MediaPipe framework, the frame length of each video is trimmed to the frames that have only detected hands.

A challenge arose after trimming, the frame lengths of video segments across the entire dataset became inconsistent, complicating batch processing during model training and dimensionality issues. To address this, an average frame length across the entire dataset is identified, which amounted to 60 frames. The next step then involved standardizing all video segments to this length. Segments with frame lengths equal to or exceeding 60 were downsampled to this benchmark. Meanwhile, segments with fewer than 60 frames were excluded to maintain uniformity. Each sample was carefully checked, and any that contained inaccurate or misleading gestures were removed. Again EMMA eye's Mediapipe framework is used to extract hand landmarks from the data samples before any other preprocessing steps. This extraction resulted in clear hand landmarks set against a black background, ensuring that only the essential features were retained for training. It has 21 landmark detected points corresponding to 21 hand-knuckle coordinates, as illustrated below. Additionally, EMMA eye is designed to draw a line between the tips of the thumb and index finger for enhanced clarity and understanding.

Hand Land Marks



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Source: Google developers. Mediapipe
URL: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

Following the extraction of only the color video segments from the original dataset and filtering to include only video segments with 60 frames or more, significant reduction in data sampleswere observed. In fact, more than half of the original data samples were excluded. This reduced dataset size was insufficient for effective model training. Given the dataset is action-based, it's crucial that any applied augmentation techniques don't disrupt the spatial and temporal iinformation of the data. Therefore, modest rotation of about 5 degrees and translation methods were applied. These techniques were carefully chosen to enhance the dataset size without compromising the quality of the gesture representations.

The entire project, including the preprocessing code, was developed using the Python programming language. With the help of Scikit-learn,the dataset is partitioned into training, testing, and validation, which constitute 70%, 15%, and 15% of the data respectively. The chosen deep learning framework for this project is PyTorch for its flexibility and efficiency. Using PyTorch's DataLoader, the datasets are loaded and batched, ensuring smooth and optimized processing during the training and evaluation phases. The batch size was set to 5, considering the available computational resources.

4 Model Training

For an action recognition dataset, which combines spatial and temporal information, the choice of the neural network model is crucial. Given the depth of deep learning models that are available, it's often good to use pretrained models, not only to lessen the training process but also to make use of the knowledge from vast datasets. In the field of deep learning, there's a level of unpredictability. Often, the best model for a specific task isn't immediately evident until used into trial and experimentation. So, it is decided to go with the ResNet3D pretrained model at first from the chosen state of the art models. This model has been trained on the Kinetics dataset, a large-scale, high-quality dataset of YouTube video URLs, which covers a wide range of human actions. The Kinetics dataset encompasses around 400 action classes with at least 400 video clips for each action, each 10 seconds long, making it a robust choice for initial transfer learning models. The richness of this dataset, combined with the depth of the ResNet3D architecture, offers a solid foundation for recognizing dynamic hand gestures. The ResNet architecture, with its unique residual connections, effectively for the vanishing gradient problem. To be precise, ResNet3D-18 architecture was chosen for this project, which gives a balance between computational efficiency and the size of the data we have.

ResNet3D Network Overview:

1. Stem:

- ➔ Conv3d: 3 input channels, 64 output channels, kernel size (3, 7, 7), stride (1, 2, 2)
- ➔ BatchNorm3d: 64 channels
- ➔ ReLU

2. Layer1:

BasicBlock (0):

- ➔ Conv3DSimple: 64 input-output channels, kernel size (3, 3, 3)
- ➔ BatchNorm3d: 64 channels
- ➔ ReLU
- ➔ Conv3DSimple: 64 input-output channels
- ➔ BatchNorm3d: 64 channels
- ➔ ReLU

BasicBlock (1):

- ➔ Conv3DSimple: 64 input-output channels
- ➔ BatchNorm3d: 64 channels
- ➔ ReLU
- ➔ Conv3DSimple: 64 input-output channels
- ➔ BatchNorm3d: 64 channels
- ➔ ReLU

3. Layer2:

BasicBlock (0):

- ➔ Conv3DSimple: 64 input channels, 128 output channels
- ➔ BatchNorm3d: 128 channels
- ➔ ReLU
- ➔ Downsample: Conv3d from 64 to 128 channels

BasicBlock (1):

- ➔ Conv3DSimple: 128 input-output channels
- ➔ BatchNorm3d: 128 channels
- ➔ ReLU

4. Layer3:

BasicBlock (0):

- ➔ Conv3DSimple: 128 input channels, 256 output channels
- ➔ BatchNorm3d: 256 channels
- ➔ ReLU
- ➔ Downsample: Conv3d from 128 to 256 channels

BasicBlock (1):

- ➔ Conv3DSimple: 256 input-output channels
- ➔ BatchNorm3d: 256 channels
- ➔ ReLU

5. Layer4:

BasicBlock (0):

- ➔ Conv3DSimple: 256 input channels, 512 output channels
- ➔ BatchNorm3d: 512 channels
- ➔ ReLU
- ➔ Downsample: Conv3d from 256 to 512 channels

BasicBlock (1):

- ➔ Conv3DSimple: 512 input-output channels
- ➔ BatchNorm3d: 512 channels
- ➔ ReLU

6. AdaptiveAvgPool3d: Output size (1, 1, 1)

7. Fully Connected layer: 512 input features, 5 output features

Device Configuration:

The computational device is selected based on the availability of CUDA. If CUDA is available, it makes that a GPU is present and can be utilized for computation, ensuring faster training and validation. If not, computations default to the CPU.

Loss Function:

Cross Entropy Loss is selected as the loss criterion for our model. This decision is particularly apt for classification tasks, given that this loss function quantifies the difference between the predicted probabilities output by the model and the actual class labels of the data. In the context of gesture recognition, the use of Cross Entropy Loss ensures that the model is penalized appropriately for incorrect predictions, making it to produce more accurate results over subsequent training iterations.

Optimizer:

Stochastic Gradient Descent (SGD) is Chosen for its robustness and widespread use in training deep neural networks. Specific parameters include: Learning Rate of 0.0005, dictates the step size during optimization. Momentum of 0.9, helps accelerate the optimizer towards the correct direction and dampens oscillations and this value is commonly used. Weight Decay of 0.00001 Serves as a regularization technique, preventing the model from fitting too closely to the training data by penalizing large coefficients.

Learning Rate Scheduler:

The ReduceLROnPlateau scheduler is deployed, aimed at decreasing the learning rate when the model's performance plateaus. If the validation loss doesn't show improvement for a set number of epochs (4 epochs in this case), the learning rate is reduced by a factor of 0.1. This strategy aids in achieving a more refined model convergence.

Training Loop:

The model undergoes training for 35 epochs. In each epoch, the model is set to training mode. Data is fetched batch-wise from the training data loader. The forward pass computes the model's predictions, followed by the calculation of the loss using the pre-defined criterion. Backpropagation computes the necessary gradients, and the optimizer updates the model parameters accordingly. The training accuracy is computed for the epoch by comparing the model's predictions against the true labels. Post-training in each epoch, the model's performance is validated using the validation dataset. The model transitions to evaluation mode, ensuring that certain layers like dropout behave differently than during training. Validation loss and accuracy metrics are determined, providing insights into the model's performance on unseen data.

Fine Tuning:

In the preliminary stages of the project, typical default values were employed for hyperparameters such as batch size, learning rate, and weight decay. However, as the training process evolved, several rounds of hyperparameter tuning were performed to optimize model performance. One significant constraint encountered was the limitation of computational resources. This restricted the batch size to a maximum of 5, a value that typically might be increased to capitalize on parallel processing capabilities and achieve smoother gradient descent. As the training progressed, it indicated overfitting, that the model was too closely fitting the training data by significant difference with validation data, which might perform poorly on unseen data. To mitigate this, regularization techniques were explored. L2 regularization, a form of weight decay, was incorporated to constrain the network's weights, discouraging overly complex models that might overfit. Furthermore, early stopping was introduced, providing a mechanism to halt training if the model's validation performance ceased to improve, preventing potential overfitting. Upon inspecting the model's architecture, the absence of dropout layers was noticed. Dropout, which randomly deactivates a subset of neurons during training, was then introduced as an additional measure against overfitting. However, this appeared causing slight underfitting. Recognizing this, the dropout layers were subsequently removed to strike a balance. Due to the computational limitations, significant alterations in hyperparameters were challenging. The batch size, for instance, remained at 5 throughout the experimentation phase.

5 Results

Regarding the results, due to limited computational resources, for the time being, the model was trained on just 5 out of the 17 gesture classes, and only the rotation augmentation technique was employed. However, the final code provided has both rotation and translation augmentation techniques and is designed to accommodate all 17 gesture classes.

Selected gestures were: {Palm Opening, Palm Shake, Pulling, Pointing, CCW_Rotation}

```
Epoch [1/35] - Train Loss: 1.5130, Train Accuracy: 34.44%, Validation Loss: 1.3295, Validation Accuracy: 44.33%
Epoch [2/35] - Train Loss: 1.3001, Train Accuracy: 48.89%, Validation Loss: 1.0082, Validation Accuracy: 68.04%
Epoch [3/35] - Train Loss: 1.0628, Train Accuracy: 60.89%, Validation Loss: 0.7057, Validation Accuracy: 74.23%
Epoch [4/35] - Train Loss: 0.7775, Train Accuracy: 76.00%, Validation Loss: 0.4878, Validation Accuracy: 90.72%
Epoch [5/35] - Train Loss: 0.5857, Train Accuracy: 81.78%, Validation Loss: 0.4724, Validation Accuracy: 84.54%
Epoch [6/35] - Train Loss: 0.4691, Train Accuracy: 89.33%, Validation Loss: 0.4448, Validation Accuracy: 89.69%
Epoch [7/35] - Train Loss: 0.3266, Train Accuracy: 92.22%, Validation Loss: 0.2868, Validation Accuracy: 89.69%
Epoch [8/35] - Train Loss: 0.3518, Train Accuracy: 89.78%, Validation Loss: 0.3126, Validation Accuracy: 89.69%
Epoch [9/35] - Train Loss: 0.2358, Train Accuracy: 94.00%, Validation Loss: 0.2623, Validation Accuracy: 90.72%
Epoch [10/35] - Train Loss: 0.2217, Train Accuracy: 95.11%, Validation Loss: 0.3411, Validation Accuracy: 91.75%
Epoch [11/35] - Train Loss: 0.1323, Train Accuracy: 98.22%, Validation Loss: 0.1452, Validation Accuracy: 95.88%
Epoch [12/35] - Train Loss: 0.1471, Train Accuracy: 96.22%, Validation Loss: 0.1383, Validation Accuracy: 95.88%
Epoch [13/35] - Train Loss: 0.0991, Train Accuracy: 98.22%, Validation Loss: 0.0949, Validation Accuracy: 96.91%
Epoch [14/35] - Train Loss: 0.1289, Train Accuracy: 97.11%, Validation Loss: 0.1488, Validation Accuracy: 93.81%
Epoch [15/35] - Train Loss: 0.1123, Train Accuracy: 98.00%, Validation Loss: 0.2819, Validation Accuracy: 90.72%
Epoch [16/35] - Train Loss: 0.0719, Train Accuracy: 99.33%, Validation Loss: 0.0903, Validation Accuracy: 97.94%
Epoch [17/35] - Train Loss: 0.0534, Train Accuracy: 99.11%, Validation Loss: 0.0640, Validation Accuracy: 100.00%
Epoch [18/35] - Train Loss: 0.0656, Train Accuracy: 98.89%, Validation Loss: 0.0890, Validation Accuracy: 97.94%
Epoch [19/35] - Train Loss: 0.1319, Train Accuracy: 96.67%, Validation Loss: 0.0943, Validation Accuracy: 98.97%
Epoch [20/35] - Train Loss: 0.0642, Train Accuracy: 99.33%, Validation Loss: 0.0791, Validation Accuracy: 97.94%
Epoch [21/35] - Train Loss: 0.1029, Train Accuracy: 98.22%, Validation Loss: 0.0790, Validation Accuracy: 100.00%
Epoch [22/35] - Train Loss: 0.0827, Train Accuracy: 98.44%, Validation Loss: 0.0554, Validation Accuracy: 98.97%
Epoch [23/35] - Train Loss: 0.0755, Train Accuracy: 98.22%, Validation Loss: 0.1134, Validation Accuracy: 96.91%
Epoch [24/35] - Train Loss: 0.0432, Train Accuracy: 99.56%, Validation Loss: 0.0639, Validation Accuracy: 96.91%
Epoch [25/35] - Train Loss: 0.0541, Train Accuracy: 99.56%, Validation Loss: 0.0526, Validation Accuracy: 98.97%
Epoch [26/35] - Train Loss: 0.0428, Train Accuracy: 99.33%, Validation Loss: 0.0978, Validation Accuracy: 95.88%
Epoch [27/35] - Train Loss: 0.0911, Train Accuracy: 97.11%, Validation Loss: 0.1585, Validation Accuracy: 92.78%
Epoch [28/35] - Train Loss: 0.0628, Train Accuracy: 99.11%, Validation Loss: 0.0611, Validation Accuracy: 97.94%
Epoch [29/35] - Train Loss: 0.0678, Train Accuracy: 98.44%, Validation Loss: 0.0961, Validation Accuracy: 98.97%
Epoch [30/35] - Train Loss: 0.0537, Train Accuracy: 98.89%, Validation Loss: 0.0790, Validation Accuracy: 96.91%
Early stopping! Best validation loss: 0.0526
```

Test Loss: 0.1404, Test Accuracy: 94.85%

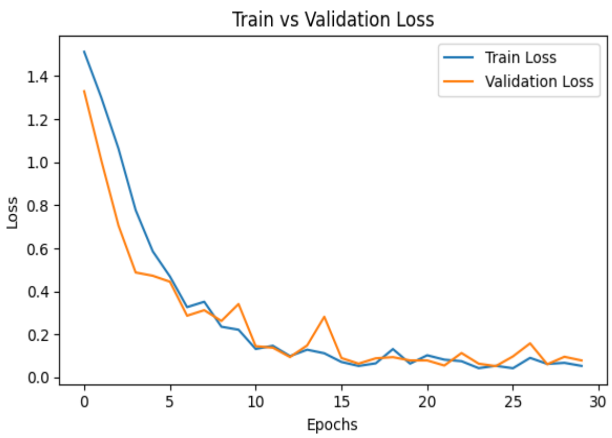
During the training phase of the hand gesture recognition model, 35 epochs were intended to be executed. However, due to the early stopping mechanism in place, the training was stopped at the 30th epoch to prevent overfitting, with the best validation loss recorded at 0.0526.

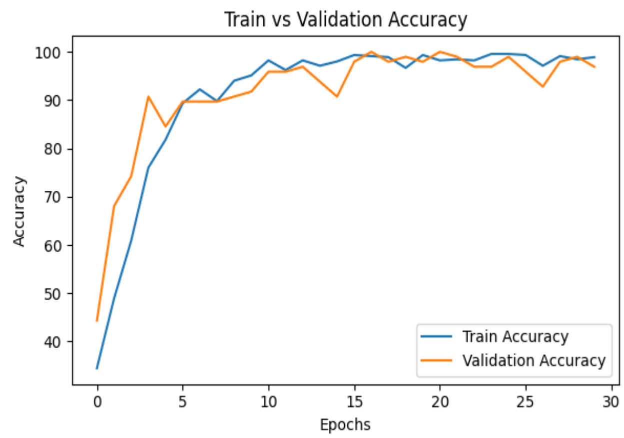
The training showed an improvement in accuracy over the epochs:

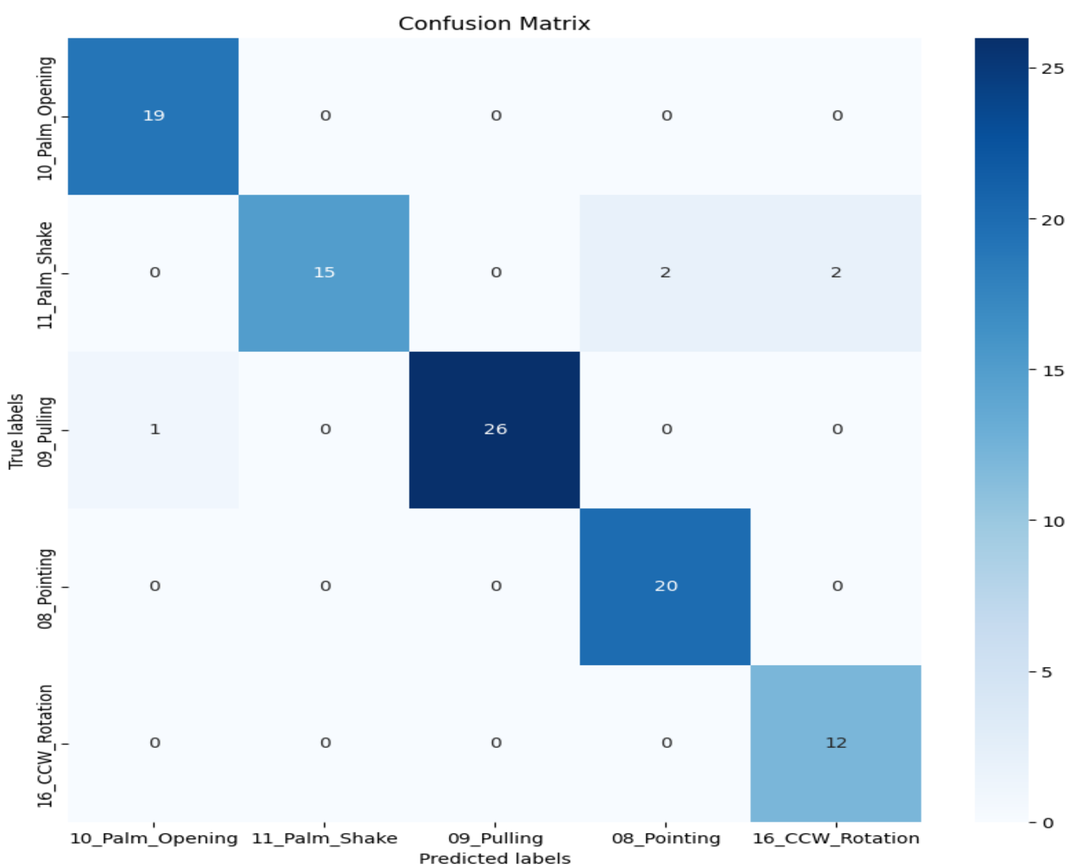
1. Training Performance: The training started with an accuracy of 34.44% in the first epoch, after then climbing to reach a high of 99.56% by the 25th epoch.
2. Validation Performance: For the validation set, accuracy began at 44.33% in the first epoch and increased significantly, touching a peak of 100% by the 21st epoch.

3. Test Performance: Once the training concluded, the model was evaluated on a test set, achieving an accuracy of 94.85% with a test loss of 0.1404.

Visual comparison of loss and accuracy metrics between the training and validation sets as shown:







The Palm shake gesture class has 4 mispredicted samples primarily due to the hand's orientation. When the hand is perpendicular to the camera, the landmarks appear to overlap or stack atop one another. As a result, some samples are misclassified as the 'pointing' gesture' while others, due to slight rotations, are incorrectly identified as the 'CCW rotation' gesture. This indicated the importance of sticking to specific guidelines and specifications during data generation. The necessary specifications are detailed in the 'Gesture Specifications' Excel sheet.

6 Challenges and limitations

One of the primary challenges encountered during the project was the inconsistency in the frame length across different gesture video segments in the dataset. This inconsistency posed a dimensionality challenge when interfacing with the neural network architecture, which expects a consistent input size. To address this, two potential solutions were identified:

Padding Technique: The first approach was to identify the video segment with the maximum frame length and then pad every other video to match this length. However, this method has a significant drawback. Padding essentially involves adding empty frames, which might not carry meaningful gesture information. This could introduce noise into the dataset and potentially mislead the model during training.

Downsampling: The alternative was to compute the average frame length across the entire dataset. For our specific dataset, this average was found to be 60 frames per second. Hence, all videos with frame lengths equal to or exceeding this average were selected. These selected videos were then downsampled to a consistent 60 frames. This method ensured that the videos retained meaningful gesture information without introducing artificial data. However, this approach resulted in the exclusion of some data samples, as gesture videos with frame lengths shorter than 60 were not considered. Moreover, the process of downsampling videos with a significantly higher number of frames to a consistent 60 frames can cause certain gesture videos to appear accelerated. This rapid pacing can introduce substantial variability in gesture dynamics, complicating the model's training due to the increased variations in gesture speed and motion. The choice between these strategies required careful consideration of their implications on the model's training and performance.

Another limitation arises from the need to maintain both spatial and temporal information within the dataset. This constraint allowed us to employ only rotation and translation as augmentation techniques, restricting our ability to diversify the dataset further.

7 Future work

Model Exploration:

While the current research utilized the ResNet3D_18 architecture, there's room for broadening the scope. Future endeavors should include training on a variety of neural network architectures to ascertain optimal performance and possibly uncover nuances unique to each model.

Emphasis on Data:

The quality and volume of the dataset remain paramount in deep learning tasks. Regardless of the sophistication of the chosen architecture, the underlying data can make or break model performance. Going forward, the objective should be to enhance the dataset, both in terms of diversity and quantity, to ensure robust and generalizable outcomes.

Preprocessing-Background Context:

The current approach extracts hand landmarks against a black background, removing the original context. A promising avenue for future research would be to train the model on hand landmarks superimposed on their original backgrounds. This could provide richer information and potentially improve the model's ability to generalize across diverse real-world scenarios.

Fine Tuning:

Owing to constraints in computational resources, the model's fine-tuning was not done greatly during this project. In future work, a more comprehensive fine-tuning process will be done to potentially enhance the model's performance.

8 Summary:

This project focuses on enhancing human-machine interaction through precise hand gesture recognition, with a particular emphasis on automotive applications. Using deep learning, the ResNet3D architecture was employed to develop a dynamic hand gesture recognition model. Data was sourced mainly from Nvidia, capturing 17 relevant automotive gestures. The preprocessing involved hand landmark extraction with EMMAeye and frame length standardization. Despite computational limitations, the model achieved a test accuracy of 94.85% on a subset of 5 out of 17 gestures. Challenges included inconsistent video frame lengths and limited augmentation techniques. Future work will explore other architectures, expand the dataset, and employ advanced fine-tuning. This research paves the way for more intuitive human-machine interactions.

References

1. Oudah M, Al-Naji A, Chahl J. Hand Gesture Recognition Based on Computer Vision: A Review of Techniques. Journal of Imaging
https://doi.org/10.3390/jimaging6080073

2. Hussain Z, Gimenez F, Yi D, Rubin D. Differential Data Augmentation Techniques for Medical Imaging Classification Tasks.
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5977656/

3. Bhatt, Dulari, Chirag Patel, Hardik Talsania, Jigar Patel, Rasmika Vaghela, Sharnil Pandya, Kirit Modi, and Hemant Ghayvat. 2021. "CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope"
https://doi.org/10.3390/electronics10202470

4. Sharma, Sakshi; Singh, Sukhwinder. "ISL recognition system using integrated mobile-net and transfer learning method." Expert Systems with Applications
https://doi.org/10.1016/j.eswa.2023.119772

5. Kumar, Vijay; Alnuaim, Abeer; Zakariah (2022). "Human-Computer Interaction with Hand Gesture Recognition Using ResNet and MobileNet." Computational Intelligence and Neuroscience
https://doi.org/10.1155/2022/8777355

6. M. L. Amit, A. C. Fajardo and R. P. Medina, "Recognition of Real-Time Hand Gestures using Mediapipe Holistic Model and LSTM with MLP Architecture,"
https://ieeexplore.ieee.org/abstract/document/10001800

7. Abu Saleh Musa Miah, Md. Al Mehedi Hasan, And Jungpil Shin, "Dynamic Hand Gesture Recognition Using Multi-Branch Attention Based Graph and General Deep Learning Model," School of Computer Science and Engineering
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10012305

8. Zhaofan Qiu, Ting Yao, Tao Mei; Proceedings of the IEEE International Conference on Computer Vision (ICCV)
https://www.microsoft.com/en-us/research/wp-content/uploads/2017/10/iccv_p3d_camera.pdf

9. Wang, S., Wang, K., Yang, T. *et al.* Improved 3D-ResNet sign language recognition algorithm with enhanced hand features.
https://www.nature.com/articles/s41598-022-21636-z#citeas

10. Bao W, Ma Z, Liang D, Yang X, Niu T. Pose ResNet: 3D Human Pose Estimation Based on Self-Supervision.
https://www.mdpi.com/1424-8220/23/6/3057

11. Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi; Dynamic Image neural network for action recognitionNetworks for Action Recognition
https://www.researchgate.net/publication/319306777_Action_Recognition_with_Dynamic_Image_Networks

12. T. Alshalali and D. Josyula, "Fine-Tuning of Pre-Trained Deep Learning Models with Extreme Learning Machine," 2018 International Conference on Computational Science and Computational Intelligence (CSCI)
https://ieeexplore.ieee.org/abstract/document/8947855