Deep Learning Course Project - Gesture Recognition

Group Members:

- Akshit Shetty
- Shyam Sunder Balaji
- Man Kit Chiu

Problem Statement:

➢ As data scientists at a home electronics company, we aim to develop a feature for smart TVs that can recognize five different gestures performed by users to control the TV without a remote. The gestures and their corresponding actions are:

Gesture	**Corresponding Action**
Thumbs up	Increase volume
Thumbs down	Decrease volume
Left swipe	Jump backward 10 seconds
Right swipe	Jump forward 10 seconds
Stop	Pause the movie

Objective:

➢ Train models on the 'train' folder to predict the action in each video sequence and perform well on the 'val' folder. The final model's performance will be evaluated on the 'test' set.

Understanding the Dataset:

➢ The training data consists of videos categorized into five classes, each containing a sequence of 30 frames. Each video is typically 2-3 seconds long, recorded by different people in front of a webcam.

Architecture Overview:

1. **3D Convolutional Neural Networks (Conv3D)**: Extract features from 30-frame video sequences using 3D convolutions.

2. **CNN + RNN Architecture**: Use CNN to extract image features and feed a sequence of these features into an RNN-based network.

Data Generator:

➢ Pre-process images by resizing, cropping, and normalizing them. Data augmentation techniques such as rotation are used to enhance the model's generalizability.

NN Architecture Development and Training:

- Experiment with various configurations, hyperparameters, and optimizers (SGD, Adam).

- Utilize Batch Normalization, pooling, dropout layers, and early stopping to prevent overfitting.

- Transfer learning with MobileNet architecture is explored to boost model accuracy.

Observations:

1. **Impact of Trainable Parameters and Training Time**: As the number of trainable parameters increases, the model's complexity and computational requirements also increase, leading to longer training times. This highlights the importance of balancing model complexity with computational resources and training time.
2. **Batch Size and GPU Memory:** The batch size is closely related to GPU memory and available compute resources. Choosing an optimal batch size is crucial to avoid GPU out-of-memory errors and ensure efficient training. Experimentation with batch size helped identify a balance between training speed and model accuracy.
3. **Trade-off Between Batch Size and Accuracy:** Increasing the batch size can significantly reduce training time but may negatively impact model accuracy. This trade-off underscores the importance of prioritizing between training time and accuracy based on project requirements.
4. **Role of Data Augmentation and Early Stopping:** Data augmentation and early stopping techniques proved effective in combating overfitting, enhancing the model's generalization ability. These strategies are essential for improving model performance and robustness.
5. **Performance of CNN+LSTM with GRU Cells:** The CNN+LSTM model with GRU cells outperformed the Conv3D architecture. This improvement could be attributed to various factors, including the data characteristics, model architecture, and hyperparameter tuning.
6. **Benefit of Transfer Learning with MobileNet:** Transfer learning with MobileNet architecture significantly boosted the model's overall accuracy. The choice of MobileNet, known for its lightweight design and high-speed performance, proved effective in achieving improved accuracy with low maintenance overhead.

Experiment	Model	Result	Decision + Explanation	Parameters
1	conv_3d1_1	Training Accuracy: 0.71 Validation Accuracy: 0.68	Investigate potential overfitting; consider reducing model complexity or using data augmentation techniques to improve generalization.	1940677
2	conv_3d1_2	Training Accuracy: 0.7 Validation Accuracy: 0.64	Model may be overfitting; explore regularization methods such as dropout or weight decay to mitigate overfitting.	1940677
3	conv_3d1_3	Training Accuracy: 0.69 Validation Accuracy: 0.57	Possible overfitting; revisit model architecture and training strategy to improve generalization.	1940677
4	conv_3d1_4	Training Accuracy: 0.67 Validation Accuracy: 0.69	Consider experimenting with different input sizes or architectures to enhance performance.	1105093
5	conv_3d1_5	Training Accuracy: 0.79 Validation Accuracy: 0.72	Model performance is promising; continue optimization efforts to further improve accuracy.	1105093
6	conv_3d1_6	Training Accuracy: 0.66 Validation Accuracy: 0.60	Investigate potential causes of overfitting and explore regularization techniques.	1105093
7	conv_3d1_7	Training Accuracy: 0.81 Validation Accuracy: 0.66	Model shows signs of overfitting; experiment with dropout rates or architectural modifications.	1105093
8	conv_3d1_8	Training Accuracy: 0.63 Validation Accuracy: 0.62	Overfitting may be occurring; consider reducing model complexity or increasing regularization.	1105093
9	conv_3d2_1	Training Accuracy: 0.66 Validation Accuracy: 0.33	Significant underperformance; review data preprocessing steps and model architecture for potential improvements.	1105093

10	conv_3d3_1	Training Accuracy: 0.56 Validation Accuracy: 0.30	Explore potential causes of underperformance and experiment with different architectures or hyperparameters.	1105093
11	conv_3d3_2	Training Accuracy: 0.82 Validation Accuracy: 0.70	Model performance is promising; further optimization may lead to improved accuracy.	1105093
12	conv_3d4_1	Training Accuracy: 0.76 Validation Accuracy: 0.66	Investigate potential overfitting and experiment with regularization techniques.	484709
13	conv_3d5_1	Training Accuracy: 0.75 Validation Accuracy: 0.74	Model performance is promising; continue optimization efforts to further improve accuracy.	484709
14	conv_3d6_1	Training Accuracy: 0.75 Validation Accuracy: 0.70	Explore potential causes of underperformance and experiment with different architectures or hyperparameters.	484709
15	conv_3d6_2	Training Accuracy: 0.81 Validation Accuracy: 0.68	Investigate potential overfitting and experiment with regularization techniques.	911973
16	conv_3d6_3	Training Accuracy: 0.77 Validation Accuracy: 0.63	Model may be underfitting; consider increasing model complexity or training for more epochs.	911973
17	conv_3d6_4	Training Accuracy: 0.71 Validation Accuracy: 0.59	Significant underperformance; review data preprocessing steps and model architecture for potential improvements.	911973
18	conv_3d7_1	Training Accuracy: 0.73 Validation Accuracy: 0.66	Model shows signs of overfitting; experiment with dropout rates or architectural modifications.	865893
19	conv_3d8_1	Training Accuracy: 0.65 Validation Accuracy: 0.34	Significant underperformance; review data preprocessing steps and model architecture for potential improvements.	865893
20	conv_3d9_1	Training Accuracy: 0.7 Validation Accuracy: 0.68	Investigate potential overfitting and experiment with regularization techniques.	1933765
21	conv_3d9_2	Training Accuracy: 0.76 Validation Accuracy: 0.57	Explore potential causes of underperformance and experiment with different architectures or hyperparameters.	1933765
22	conv_3d9_3	Training Accuracy: 0.45 Validation Accuracy: 0.33	Significant underperformance; review data preprocessing steps and model architecture for potential improvements.	1105093
23	test_conv_3d5_1	Training Accuracy: 0.83 Validation Accuracy: 0.72	Model performance is promising; continue optimization efforts to further improve accuracy.	484709
24	test_conv_3d5_1	Training Accuracy: 0.75 Validation Accuracy: 0.72	Model performance is promising; continue optimization efforts to further improve accuracy.	484709
25	cnn_rnn_1	Training Accuracy: 0.92 Validation Accuracy: 0.70	Investigate potential overfitting; consider reducing model complexity or using data augmentation techniques to improve generalization.	1405813

26	cnn_rnn_2	Training Accuracy: 0.86 Validation Accuracy: 0.76	Model performance is promising; further optimization may lead to improved accuracy.	1405813
27	cnn_rnn_3	Training Accuracy: 0.82 Validation Accuracy: 0.75	Model performance is promising; continue optimization efforts to further improve accuracy.	1405813
28	cnn_rnn_4	Training Accuracy: 0.92 Validation Accuracy: 0.84	Model performance is promising; further optimization efforts may lead to improved accuracy.	337717
29	cnn_lstm_1	Training Accuracy: 0.73 Validation Accuracy: 0.76	Model performance is promising; continue optimization efforts to further improve accuracy.	415413
30	cnn_lstm_2	Training Accuracy: 0.76 Validation Accuracy: 0.80	Model performance is promising; continue optimization efforts to further improve accuracy.	1005541
31	rnn_cnn_tl_ mobilenet	Training Accuracy: 0.97 Validation Accuracy: 0.95	Model shows outstanding performance; further fine-tuning may lead to marginal improvements.	3840453
32	cnn_lstm_3	Training Accuracy: 0.71 Validation Accuracy: 0.79	Model performance is promising; further optimization efforts may lead to improved accuracy.	1005541
33	cnn_lstm_4	Training Accuracy: 0.79 Validation Accuracy: 0.79	Model performance is promising; continue optimization efforts to further improve accuracy.	1657445

After doing all the experiments, we finalized models which performed well.

1. conv_3d5_1 from Conv3D (Experiment Number 13)
2. cnn_rnn_4 for GRU (Experiment Number 28)
3. cnn_lstm_2 for CNN+LSTM (Experiment Number 30)
4. **rnn_cnn_tl_mobilenet (BEST MODEL!)**

Further suggestions for improvement:

- **Using Transfer Learning**: Using a pre-trained *ResNet50/ResNet152/Inception V3* to identify the initial feature vectors and passing them further to a *RNN* for sequence information before finally passing it to a softmax layer for classification of gestures. (This was attempted but other pre-trained models couldn't be tested due to lack of time and disk space in the nimblebox.ai platform.)
- **Using GRU:** A *GRU* model in place of *LSTM* appears to be a good choice. Trainable Parameters of a *GRU* are far less than that of a *LSTM*. Therefore would have resulted in faster computations. However, its effect on the validation accuracies could be checked to determine if it is actually a good alternative over LSTM.
- **Deeper Understanding of Data:** The video clips where recorded in different backgrounds, lightings, persons and different cameras where used. Further exploration on the available images could give some more information about them and bring more diversity in the dataset. This added information can be exploited in favour inside the generator function adding more stability and accuracy to model.
- **Tuning hyperparameters:** Experimenting with other combinations of hyperparameters like, activation functions (*ReLU, Leaky ReLU, mish, tanh, sigmoid*), other optimizers like *Adagrad()* and *Adadelta()* can further help develop better and more accurate models. Experimenting with other combinations of hyperparameters like the *filter size, paddings, stride_length, batch_normalization, dropouts* etc. can further help improve performance.