**Gesture Recognition using CNN and RNN**

-Observations done by Ramkumar R and Rashmi N

**Aim**

To identify the best fit model for a TV company who tends to install a gesture recognition system by using two different deep learning model architectures.The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

* Thumbs up: Increase the volume
* Thumbs down: Decrease the volume
* Left swipe: 'Jump' backwards 10 seconds
* Right swipe: 'Jump' forward 10 seconds
* Stop: Pause the movie

The architectures used in this case study for gesture recognitions are:

* CNN + RNN
* Conv3D CNN

**Experiments using Conv3D CNN Architecture**

Conv3D involves using 3D convolutional layers to process volumetric data, such as medical images or video frames. The reason behind choosing this architecture is that we can achieve the efficiency of the model with less number of parameters compared to CNN + RNN architecture and it works specifically well on video frames. It has its own pros and cons that we can discuss later. The following are the experiments done while developing a Conv3D architecture.

\*\*Every model processes for 50 epochs and a constant learning rate in all the experiments mentioned below.

| **Experiment Number** | **Model Name** | **Metrics and Parameters** | **Results** | **Decisions and Explanations** |
| --- | --- | --- | --- | --- |
| 1 | **Conv3D** | * Batch size - 32 | Got image dimension error in the generator function | Changed the image dimensions and normalised the data |
| 2 | **Conv3D** | * Batch size - 32 * categorical\_accuracy: 1.0000 * loss: 0.0178 * val\_loss: 1.3512 * val\_categorical\_accuracy: 0.6806 * lr: 1.0000e-05 * Total params: 5,135,557 * Trainable params: 5,135,557 * Non-trainable params: 0 * Steps: 21 | Faced file name error in the model checkpoint during fitting the model | Adjusted the model checkpoint code to fix this error |
| 3 | **Conv3D** | * Batch size - 32 * categorical\_accuracy: 0.9773 * loss: 0.1125 * val\_loss: 0.8934 * val\_categorical\_accuracy: 0.7778 * lr: 1.0000e-05 * Total params: 2,840,197 * Trainable params: 2,840,197 * Non-trainable params: 0 * Steps: 21 | Better performance than the previous model with less number of parameters | Added a new layer to the previous model |
| 4 | **Conv3D** | * Batch size - 32 * categorical\_accuracy: 0.8866 * loss: 0.4431 * val\_loss: 3.2862 * val\_categorical\_accuracy: 0.6528 * lr: 1.0000e-05 * Total params: 2,840,645 * Trainable params: 2,840,421 * Non-trainable params: 224 * Steps: 21 | Slight increase in the number of parameters but the accuracies are reduced | Added batch normalisation to the previous model |
| 5 | **Conv3D** | * Batch size - 32 * categorical\_accuracy: 0.9738 * loss: 0.1191 * val\_loss: 0.4629 * val\_categorical\_accuracy: 0.8611 * lr: 1.0000e-05 * Total params: 2,841,157 * Trainable params: 2,840,677 * Non-trainable params: 480 * Steps: 21 | Slight increase in the number of parameters but this is the best model with better accuracies | Added Dropout value of 0.5 to the previous model |
| 6 | **Conv3D** | * Batch size - 32 * categorical\_accuracy: 0.9738 * loss: 0.1998 * val\_loss: 0.5882 * val\_categorical\_accuracy: 0.7639 * lr: 1.0000e-05 * Total params: 711,077 * Trainable params: 710,837 * Non-trainable params: 240 * Steps: 21 | Sudden decrease in the number of parameters. | Tuned the model by reducing the filter size |
| 7 | **Conv3D** | * Batch size - 64 * categorical\_accuracy: 0.9192 * loss: 0.3319 * val\_loss: 0.8183 * val\_categorical\_accuracy: 0.6700 * lr: 1.0000e-05 * Total params: 711,077 * Trainable params: 710,837 * Non-trainable params: 240 * Steps: 11 | Decrease in the accuracies with the same number of parameters as the previous model but the number of steps get reduced during the epochs | Changing the batch size to 64 in the previous model |
| 8 | **Conv3D** | * Batch size - 64 * categorical\_accuracy: 0.8930 * loss: 0.4359 * val\_loss: 0.8434 * val\_categorical\_accuracy: 0.7200 * lr: 1.0000e-05 * Total params: 1,284,549 * Trainable params: 1,284,437 * Non-trainable params: 112 * Steps: 11 | This made the model slightly better but comes with more number of parameters | Removed a layer and changed dropout value to 0.4 |
| 9 | **Conv3D** | * Batch size - 16 * categorical\_accuracy: 0.9151 * loss: 0.2941 * val\_loss: 0.6311 * val\_categorical\_accuracy: 0.7500 * lr: 1.0000e-05 * Total params: 476,517 * Trainable params: 476,021 * Non-trainable params: 496 * Steps: 42 | Decreasing batch size made an increase in the number of steps and a slight increase in the validation accuracy | Added 4 layers to the model with the dropout value of 0.5 , reduced filters and changing the batch size to 16 |
| 10 | **Conv3D** | * Batch size - 32 * categorical\_accuracy: 0.9442 * loss: 0.2100 * val\_loss: 0.5487 * val\_categorical\_accuracy: 0.8056 * lr: 1.0000e-05 * Total params: 476,517 * Trainable params: 476,021 * Non-trainable params: 496 * Steps: 21 | This gives the best model with even less number of parameters comparing the model in experiment 5 | Changing the batch size to 32 in a 4 layer model |
| 11 | **Conv3D** | * Batch size - 32 * categorical\_accuracy: 0.7016 * loss: 0.7977 * val\_loss: 0.9165 * val\_categorical\_accuracy: 0.6111 * lr: 1.0000e-05 * Total params: 78,565 * Trainable params: 78,197 * Non-trainable params: 368 * Steps: 21 | This model has the least number of parameters after adding the global average pooling on comparing all the other models but works poorly. Trying with different models may give better results. | Changing the flatten layer to global average pooling layer with a dropout value of 0.5 |

After all these, I have done some more experiments which gave similar results. Some of the experiments are listed below.

* Reset to experiment 5 with global average pooling
  + Epoch 00050: Batch size -32,

Steps: 21/21 - 64s 3s/step - loss: 0.1582 - categorical\_accuracy: 0.9843 - accuracy: 0.9843 - val\_loss: 0.6429 - val\_categorical\_accuracy: 0.7778 - val\_accuracy: 0.7778 - lr: 1.0000e-05

* Previous model with batch size of 16
  + Epoch 00050: Batch size -16,

Steps: 21/21 - 64s 3s/step - loss: 0.1582 - categorical\_accuracy: 0.9843 - accuracy: 0.9843 - val\_loss: 0.6429 - val\_categorical\_accuracy: 0.7778 - val\_accuracy: 0.7778 - lr: 1.0000e-05

* Finally I tried to achieve the model in experiment 5 which has the best accuracy. But not able to recreate it,
  + Batch size -32, Steps: 21/21 - 42s 2s/step - loss: 0.1187 - categorical\_accuracy: 0.9773 - accuracy: 0.9773 - val\_loss: 0.4811 - val\_categorical\_accuracy: 0.8472 - val\_accuracy: 0.8472 - lr: 1.0000e-05.

**Conclusion**

Experiment 5 seems to strike a good balance between complexity (parameters) and performance (accuracy and loss), making it a potentially optimal choice. However, Experiment 10 also provides good performance with significantly fewer parameters, making it a suitable option for resource-constrained environments. It might be beneficial to explore further tuning of hyperparameters, especially for models with fewer parameters, to optimise both accuracy and efficiency.

**Experiments using CNN + RNN Architecture**

| **Experiment Number** | **Model Name** | **Metrics and Parameters** | **Results** | **Decisions and Explanations** |
| --- | --- | --- | --- | --- |
| 1 | **CNN+LSTM** | * Batch size -32 * Epochs - 30 * Total params - 1,005,541 * loss - 0.8824 * categorical\_accuracy - 0.6678 * val\_loss - 1.9438 * val\_categorical\_accuracy - 0.1667 * lr - 1.0000e-05 * Trainable params - 1,004,549 * Non-trainable params -992 * Steps -17 | This model is having least number of parameters but no good accuracy | Two-stage model (CNN + LSTM)  CNN+LSTM have least number of parameters and used density[16,32,64,128,256] |
| 2 | **CNN+LSTM** | * Batch size -32 * Epochs - 30 * Total params - 415,413 * loss - 0.9535 * categorical\_accuracy - 0.6298 * val\_loss - 1.3055 * val\_categorical\_accuracy - 0.3333 * lr - 1.0000e-05 * Trainable params - 414,917 * Non-trainable params -496 * Steps -17 | There is an increase in the accuracy than the previous model but not much. | Two-stage model (CNN + LSTM)  CNN+LSTM have least number of parameters and used density [16,32,64,128]  Reduced number of frames from 30 to 16 |
| 3 | **CNN+LSTM** | * Batch size -32 * Epochs - 50 * Total params - 415,413 * loss - 0.7690 * categorical\_accuracy - 0.7682 * val\_loss - 0.7450 * val\_categorical\_accuracy - 0.7500 * lr - 1.0000e-05 * Trainable params - 414,917 * Non-trainable params -496 * Steps -21 | On increasing the epoch size to 50 there is great improvement in the val\_accuracy | Just changed the epoch to 50 |
| 4 | **CNN+LSTM** | * Batch size -32 * Epochs - 50 * Total params - 1,005,541 * loss - 0.8196 * categorical\_accuracy - 0.7255 * val\_loss - 0.7957 * val\_categorical\_accuracy - 0.8125 * lr - 1.0000e-05 * Trainable params - 1,004,549 * Non-trainable params -992 * Steps -21 | This model has least number of parameter and having better model till now with val\_accuracy - 0.8125 | Increased frame to 30 This gave better accuracy with comparatively less parameters and density [8,16,32,64,128] |
| 5 | **GRU** | * Batch size -32 * Epochs -30 * Total params - 337,717 * loss - 0.5289 * categorical\_accuracy - 0.8339 * val\_loss - 1.4213 * val\_categorical\_accuracy - 0.4167 * lr - 1.0000e-05 * Trainable params - 337,221 * Non-trainable params -496 * Steps -17 | Number of parameter is less but accuracy in very low | Initial GRU Model with density [8,16,32,64,128] with 120X120 image and frame number = 16 |
| 6 | **GRU** | * Batch size -32 * Epochs - 35 * Total params - 1,319,781 * loss - 0.7652 * categorical\_accuracy - 0.7233 * val\_loss - 1.0317 * val\_categorical\_accuracy - 0.6667 * lr - 1.0000e-05 * Trainable params - 1,319,301 * Non-trainable params -480 * Steps -17 | There is increase in the number of parameters and improvement in the accuracy | Reducing the density of the layer [16,32,64,128] and increased epochs to 35 |

**Conclusion**

#### **CNN-LSTM Models**

* **Model 1**: Shows decent training accuracy (0.6678) but very poor validation accuracy (0.1667), indicating significant overfitting.
* **Model 2**: Has a lower training accuracy (0.6298) and slightly better validation accuracy (0.3333) compared to Model 1, but still shows overfitting.
* **Model 3**: Improves significantly with 50 epochs, achieving a good balance between training accuracy (0.7682) and validation accuracy (0.7500), indicating good generalisation.
* **Model 4**: Shows the best validation accuracy among CNN-LSTM models (0.8125) with a high training accuracy (0.7255), suggesting it generalises well while being robust in training.

#### **GRU Models**

* **Model 1**: Achieves high training accuracy (0.8339) but struggles with validation accuracy (0.4167), indicating overfitting.
* **Model 2**: Shows a balanced performance with good training accuracy (0.7233) and reasonable validation accuracy (0.6667), suggesting a well-regularised model.
* **Model 3**: Improves upon Model 1 with more epochs, showing good training accuracy (0.7958) and improved validation accuracy (0.6667).
* **Model 4**: Demonstrates the highest training accuracy (0.8547) among all models and a reasonable validation accuracy (0.6167), indicating good learning but potential overfitting.

**CNN-LSTM Model 4**

* **Training Accuracy**: 0.7255
* **Validation Accuracy**: 0.8125
* **Conclusion**: This model has the best validation accuracy, indicating it generalises well to unseen data while maintaining high training accuracy. Despite having a higher number of parameters, it balances complexity and performance effectively.

**Recommendations**:

* Use CNN-LSTM Model 4 as the baseline for further improvements.
* Consider hyperparameter tuning, such as learning rate adjustments and exploring different architectures.
* Implement techniques like early stopping and data augmentation to prevent overfitting and enhance generalisation.