**Hand Gesture Recognition with CNN and RNN**

**By Shu-Ren Chang**

**Project goal:** In this project, the task is to build a 3D Conv model that will be able to predict the 5 gestures correctly. Imagine you are working as a data scientist at a home electronics company which manufactures state of the art smart televisions. You want to develop a cool feature in the smart-TV that can recognize five different gestures performed by the user which will help users control the TV without using a remote. Five gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

1. **Thumbs Up**: Increase the volume.
2. **Thumbs Down**: Decrease the volume.
3. **Left Swipe**: 'Jump' backwards 10 seconds.
4. **Right Swipe**: 'Jump' forward 10 seconds.
5. **Stop**: Pause the movie.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Exp. No.** | **Models** | **No. of Trainable**  **Parameters** | **Result/Accuracy** | **Model summary and actions** |
| 1. | **Conv3D**   * Image frames=20 * Batch\_size=50 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3) | 1,116,325 | categorical\_accuracy:94%  val\_categorical\_accuracy:25% | The Model 1 above is obviously overfitting with the validation accuracy of 25% and training accuracy of 94%. more dropout layers will be added to improv the model performance. |
| 2. | **Conv3D**   * Image frames=20 * Batch\_size=30 * Epoch =25 * Activation function = ‘relu’ * Kernel\_size=(3,3,3) * Dense\_neurons=256 * Dropout=0.5 | 3,637,477 | categorical\_accuracy:85%  val\_categorical\_accuracy:25% | The val\_loss did not improve much from 3.72. Thus, the model training did not succeed and stops early. The model training Epoch stop at the step of 12 /25. Best weights are saved automatically. This is an obvious overfitting with the validation accuracy of 25% and training accuracy of 85%. Next the filter size and image resolution could be reduced to see if it performs better. Since the loss is oscillating, the learning rate could be set up lower to 0.0002. |
| 3. | **Conv3D**   * Image frames=16 * Batch\_size=20 * Epoch =15 * Activation function = ‘relu’ * Kernel\_size=(2,2,2) * Dense\_neurons=256 * Dropout=0.5 * Image res 120x120 | 1,761,109 | categorical\_accuracy:76%  val\_categorical\_accuracy:21% | The Model yielded the poor validation accuracy at 21% and training one at 76%, although the parameter size is reduced by half compared to the earlier model. Next more layers will be added. |
| 4. | **Conv3D**   * Image frames=16 * Batch\_size=15 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3) * Dense\_neurons=256 * Dropout=0.5 * Image res 120x120 | 2,554,549 | categorical\_accuracy:63%  val\_categorical\_accuracy:78% | After adding more layers, the performance was improved with much better 63% of validation accuracy and 78% of training one. Dropout layers could be added at the convolutional layers |
| 5. | **Conv3D**   * Image frames=16 * Batch\_size=15 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3) * Dense\_neurons=256 * Dropout=0.25 * Image res 120x120 | 2,554,549 | categorical\_accuracy:88%  val\_categorical\_accuracy:37% | The overfitting still persists and it doesn't improve after adding dropouts. The validation accuracy reduced to 37% as the training accuracy increased to 88%. More than 1 million parameters are used for all experiments above. It's the time to reduce the model size to seek the performance improvement. |
| 6. | **Conv3D**   * Image frames=16 * Batch\_size=15 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3) * Dense\_neurons=256 * Dropout=0.25 * Image res 120x120 | 695,653 | categorical\_accuracy:83%  val\_categorical\_accuracy:33% | The model performance doesn’t improve and get even lower validation accuracy at 33% and training accuracy at 83%. |
| 7. | **Conv3D**   * Image frames=16 * Batch\_size=15 * Epoch =15 * Activation function = ‘relu’ * Kernel\_size=(3,3,3) * Dense\_neurons=64 * Dropout=0.25 * Image res 120x120 * Learning rate=0.0002 | 503,973 | categorical\_accuracy:77%  val\_categorical\_accuracy:66% | After the number of parameters were reduced, the model performance looked a little better with an increased validation accuracy up to 66% and decreased training accuracy down to 77%. |
| 8. | **CNN+LSTM**   * Image frames=18 * Batch\_size=15 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3) * Dense\_neurons=128 * Dropout=0.25 * Image res 120x120 * Learning rate=0.0002 | 1,656,453 | categorical\_accuracy:92%  val\_categorical\_accuracy:49% | With the new tryout using the CNN-LSTM model, the overfitting issues still existed with better training accuracy at 92%, but having poor validation accuracy at 49%. To address the overfitting issues, the data could be augmented with *\*slight rotation\** applied to the same models again. |
| 9. | **CNN+LSTM+ Augment**   * Image frames=20 * Batch\_size=15 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3) * Dense\_neurons=256 * Dropout=0.5 * Image res 160x160 * Learning rate=0.0002 | 3,637,477 | categorical\_accuracy:82%  val\_categorical\_accuracy:32% | With the new tryout using augmentation strategy, the overfitting issues still existed with both decreasing training accuracy down to 82% and validation one down to 32%. To address the overfitting issues, more experiments using augmentations will be used again. |
| 10. | **Conv3D+ Augment**   * Image frames=16 * Batch\_size=30 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(2,2,2) * Dense\_neurons=256 * Dropout=0.5 * Image res 120x120 * Learning rate=0.0002 | 1,761,109 | categorical\_accuracy:71%  val\_categorical\_accuracy:12% | With the same augmentation strategy, the overfitting issues even got worse on both decreasing training accuracy down to 71% and validation one down to 12%. To address the overfitting issues, more experiments using augmentations will be used again. |
| 11. | **Conv3D+ Augment**   * Image frames=16 * Batch\_size=30 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3,) * Dense\_neurons=256 * Dropout=0.5 * Image res 120x120 * Learning rate=0.0002 | 2,554,549 | categorical\_accuracy:68%  val\_categorical\_accuracy:25% | With the same augmentation strategy, the overfitting issues still existed on both decreasing training accuracy down to 68% and validation one at 25%. To address the overfitting issues, more experiments using augmentations will be used again. |
| 12. | **Conv3D+ Augment**   * Image frames=16 * Batch\_size=30 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3,) * Dense\_neurons=256 * Dropout=0.25 * Image res 120x120 * Learning rate=0.0002 | 2,554,549 | categorical\_accuracy:65%  val\_categorical\_accuracy:12% | This model suffers much severe overfitting. With a dropout rate decreased from 0.5 to 0.25 and the same augmentation strategy, the overfitting issues still existed on both decreasing training accuracy down to 65% and validation one at 12%. To address the overfitting issues, more experiments using augmentations will be used again. |
| 13. | **Conv3D+ Augment**   * Image frames=16 * Batch\_size=30 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3,) * Dense\_neurons=128 * Dropout=0.25 * Image res 100x100 * Learning rate=0.0002 | 695,653 | categorical\_accuracy: 79%  val\_categorical\_accuracy:20% | The overfitting issues improved a little bit. With the number of dense\_neurons reduced from 256 to 128 and the same augmentation strategy, the overfitting issue got much better. It reflected on both increasing training accuracy up to 79% and validation one up to 20%. To address the overfitting issues, more experiments using augmentations will be used again. |
| 14. | **Conv3D+ Augment**   * Image frames=16 * Batch\_size=30 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3,) * Dense\_neurons=64 * Dropout=0.25 * Image res 100x100 * Learning rate=0.0002 | 503,973 | categorical\_accuracy:77%  val\_categorical\_accuracy:18% | The overfitting issues did not improve. With the number of dense\_neurons reduced from 128 to 64 and the same augmentation strategy, the overfitting issue still existed. It reflected on both decreasing training accuracy down to 77% and validation one down to 18%. To address the overfitting issues, more experiments using augmentations will be used again. |
| 15. | **CNN LSTM with GRU**   * Image frames=18 * Batch\_size=30 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3,) * Dense\_neurons=128 * Dropout=0.25 * Image res 120x120 * Learning rate=0.0002 | 2,573,445 | categorical\_accuracy:91%  val\_categorical\_accuracy:25% | The same overfitting is still dramatical when the augmentation, dropout, LSTM with GRU were applied, but the number of dense\_neurons were added from 64 to 128. The overfitting issues did not improve much. It reflected on both increasing training accuracy down to 91% and validation one down to 25%. Adding a greater number of dense\_neurons from 64 to 128 might be the culprit to cause the overfitting persistence. To address the overfitting issues, more experiments using transfer learning will be used again. |
| 16. | **Transfer learning LSTM**   * Image frames=16 * Batch\_size=15 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3,) * Dense\_neurons=128 * Dropout=0.25 * Image res 120x120 * Learning rate=0.0002 | 609,541 | categorical\_accuracy:98%  val\_categorical\_accuracy:86% | The model performance excelled all others with a high validation accuracy at 86% and an excellent training one at 98%, even the weights using the mobilenet hasn't been well trained yet. More training will be conducted in the next round to check if the model performance will be improved. |
| 17. | **Transfer learning with GRU**   * Image frames=16 * Batch\_size=15 * Epoch =10 * Activation function = ‘relu’ * Kernel\_size=(3,3,3) * Dense\_neurons=128 * Dropout=0.25 * Image res 120x120 * Learning rate=0.0002 | 3,669,317 | categorical\_accuracy:99%  val\_categorical\_accuracy:97% | The model performance excelled again with a high validation accuracy at 97% and an excellent training one at 99%. |