**Gesture Recognition**

**Problem Statement:**

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. Below are the gestures:

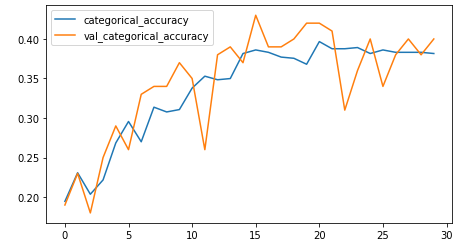
* Thumbs up: Increase the volume
* Thumbs down: Decrease the volume
* Left swipe: 'Jump' backwards 10 seconds
* Right swipe: 'Jump' forward 10 seconds
* Stop: Pause the movie

**Objective:**

The objective is to train multiple models to correctly identify each gesture. We have to build two kind of models which are 1) Conv3D and 2) Conv + LSTM/Conv + GRU. Accuracy metrics will be used to check which model is performing better. The model will be trained on training data and validate on validation data. Finally after building multiple models, h5 file of the best model needs to be saved which will be used to check the performance on test data.

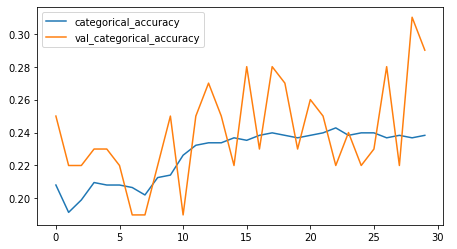
**Model Building**

**Model 1 (Base Model Conv3D):** Out of 30 images, most of the images are providing the similar information so chosen 11 images by skipping 2 image frames. Images are of 2 sizes so chosen 120\*120 by considering the computation cost. Initially the batch size was considered as 10 because we wanted to slightly increase the batch size to see the impact. Below is the accuracy metrics and graph:



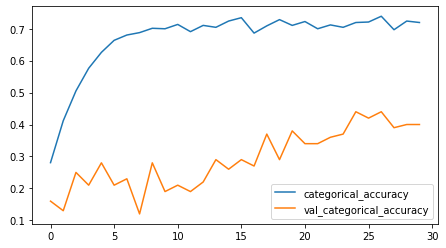
1. categorical\_accuracy: 0.3816
2. val\_categorical\_accuracy: 0.4000

**Model 2(Conv3D):** We have observed that model was **underfitting** so increased the batch size to 30 so see if it improves the performance of the model. Remaining parameters were remain constant as we wanted to modify the parameters one by one to see the impact and performance of the model.



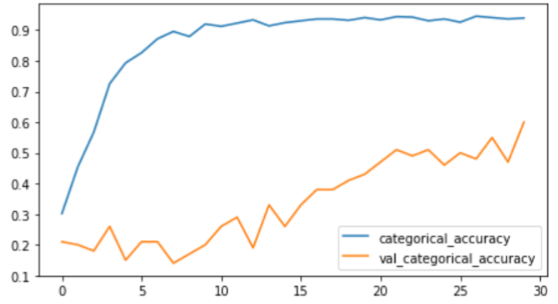
1. categorical\_accuracy: 0.2383
2. val\_categorical\_accuracy: 0.2900

**Model 3(Conv3D):** Model 2 was failed miserably as it was a clear case of **underfitting**. To improve the model performance, we changed the batch size to 20, included dropouts and batch normalization.



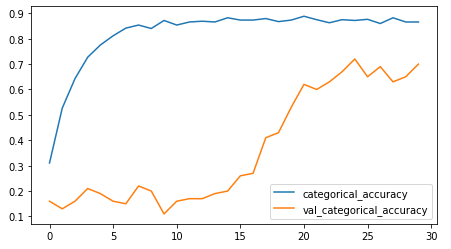
1. categorical\_accuracy: 0.7195
2. val\_categorical\_accuracy: 0.4000

**Model 4(Conv3D):** We got rid of underfitting but above model gave an **overfitting** model. So this time, we changed the learning rate from default to 0.01. Remaining parameter remained constant.



1. categorical\_accuracy: 0.9397
2. val\_categorical\_accuracy: 0.6000

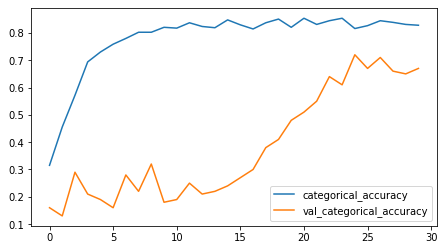
**Model 5(Conv3D):** The above model also seems **overfitting**. So to fix this issue, we increased the number of image frames from 11 to 15. Remaining parameters kept constant and the result was commendable.



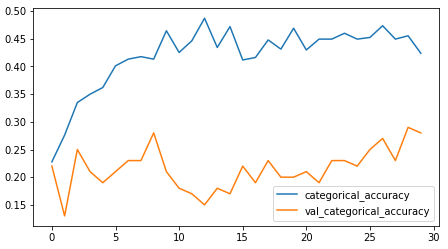
1. categorical\_accuracy: 0.8658
2. val\_categorical\_accuracy: 0.7000

**Model 6(Conv3D):** Now we can see that the model performed really well. Still it is slightly **overfitting** but a huge improvement in the mode. This time we tried to change the image size from 120\*120 to 84\*84 and see the model performance.

1. categorical\_accuracy: 0.8281
2. val\_categorical\_accuracy: 0.6700



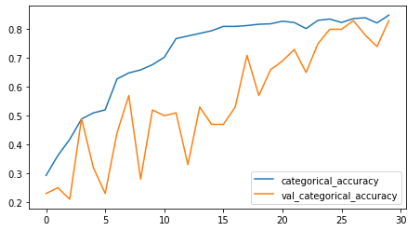
**Model 7(Conv3D):** Even after changing the image size, model was **overfitting**. So we tried to change the dropout from 0.25 to 0.5 to see if performance improves.



1. categorical\_accuracy: 0.4238
2. val\_categorical\_accuracy: 0.2800

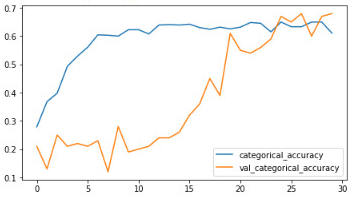
**Final Model**

**Model 8(Final Model Conv3D):** This was the clear case of **underfitting** model which means changing the dropout to 0.5 was not a good choice. So we changed the dropout value back to 0.25 and introduced momentum as 0.9 to see the model performance and the results were outstanding. We have considered it final model.



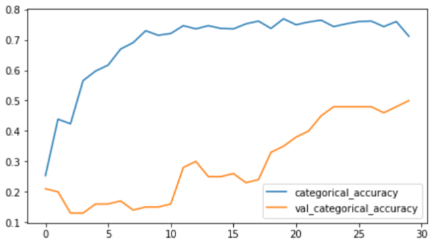
1. categorical\_accuracy: 0.8477
2. val\_categorical\_accuracy: 0.8100

**Model 9(Conv3D):** Though the above model performed really well and we considered it as **final model**, we wanted to improve it further. So changed the learning rate to 0.001 and kept the remaining parameters as constant to check the model performance.



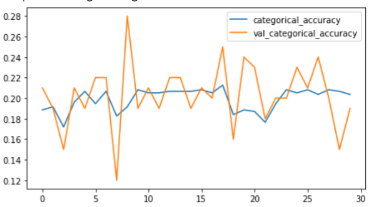
1. categorical\_accuracy: 0.6109
2. val\_categorical\_accuracy: 0.6800

**Model 10(Conv3D):** Again model started **underfitting** so changing the learning rate was not a good choice. This time we changed the number of layers from 5 to 7 to check the model performance.



1. categorical\_accuracy: 0.7119
2. val\_categorical\_accuracy: 0.5000

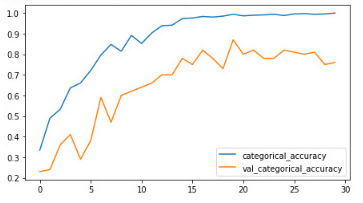
**Model 11(Conv3D):** Again the model is **overfitting** so changing the number of layers was not a good choice. This time we changed the optimizer from Relu to Adam and checked the model performance.



1. categorical\_accuracy: 0.2036
2. val\_categorical\_accuracy: 0.1900

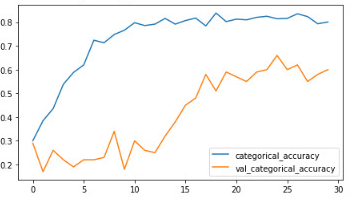
Looks like changing the optimizer to Adam was the worst decision and the model is **underfitting** miserably.

**Model 12(Conv2D+LSTM):** Built the Conv2D model and passed to RNN using GRU to see if the model performs better. Below are the results:



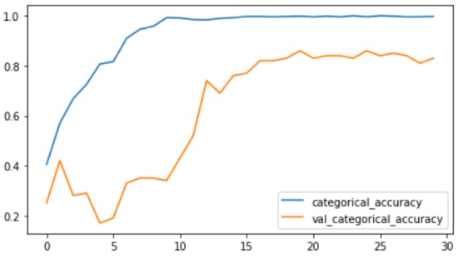
1. categorical\_accuracy: 1.0000
2. val\_categorical\_accuracy: 0.7600

**Model 13(Conv2D+LSTM):** We can clearly see from the above results that the model is **overfitting**. To avoid it, we included the dropouts in each layer. But the results were not satisfactory.



1. categorical\_accuracy: 0.8009
2. val\_categorical\_accuracy: 0.6000

**Model 14(Conv2D+GRU):** The LSTM didn’t perform even after including the dropouts as the model is still **overfitting**. We have also built the model using GRU where the convolutional 2D model is passed to GRU. The parameters remained same as LSTM and Conv3D final model. Below are the results:



1. categorical\_accuracy: 0.9970
2. val\_categorical\_accuracy: 0.8300

The above all the models are either underfitting or overfitting. So we have concluded that CNN+RNN is not performing in our case and we have considered the Model 8 as final model.

Below is the summarized models in the tabular format.

|  |  |  |  |
| --- | --- | --- | --- |
| **Exp No.** | **Model** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D**   * **Activation function :** Relu * **Image size:** 120\*120 * **Kernel Size:** (3,3,3) * **No. of image frames:** 11 * **Batch Size:** 10 | * categorical\_accuracy: 0.3816 * val\_categorical\_accuracy: 0.4000 | Images have been resized to 120\*120 with 11 images and clearly model is **underfitting**. |
| **2** | **Conv3D**   * **Activation function :** Relu * **Image size:** 120\*120 * **Kernel Size:** (3,3,3) * **No. of image frames:** 11 * **Batch Size:** 30 | * categorical\_accuracy: 0.2383 * val\_categorical\_accuracy: 0.2900 | Changed the number of batches from 10 to 30 and the model performed even badly. So the choice of increasing the number of batches was not correct. |
| **3** | **Conv3D**   * Introduced **Batch Normalization** * Included **dropout** 0.25 * **Activation function :** Relu * **Image size:** 120\*120 * **Kernel Size:** (3,3,3) * **No. of image frames:** 11 * **Batch Size:** 20 | * categorical\_accuracy: 0.7195 * val\_categorical\_accuracy: 0.4000 | Included Batch Normalization, dropouts 0.25 and changed the number of batches to 20. We can clearly see that we are getting **overfitting** model. |
| **4** | **Conv3D**   * **Learning Rate:** 0.01 * **Activation function :** Relu * **Image size:** 120\*120 * **Kernel Size:** (3,3,3) * **No. of image frames:** 11 * **Batch Size:** 20 | * categorical\_accuracy: 0.9397 * val\_categorical\_accuracy: 0.6000 | Changed the learning rate to see the accuracy. Training accuracy came to 94% while the validation accuracy is 60% which clearly shows that the model is overfitting. |
| **5** | **Conv3D**   * **Learning Rate:** 0.01 * **Activation function :** Relu * **Image size:** 120\*120 * **Kernel Size:** (3,3,3) * **No. of image frames:** 15 * **Batch Size:** 20 | * categorical\_accuracy: 0.8658 * val\_categorical\_accuracy: 0.7000 | Changed the number of image frames to 15. The model performed really well but still it is overfitting. We need to modify it further to get a decent model. |
| **6** | **Conv3D**   * **Learning Rate:** 0.01 * **Activation function :** Relu * **Image size:** 84\*84 * **Kernel Size:** (3,3,3) * **No. of image frames:** 15 * **Batch Size:** 20 | * categorical\_accuracy: 0.8281 * val\_categorical\_accuracy: 0.6700 | Reduced the image size to 84\*84 so see the model performance. We can see from the graph that the model is still slightly overfitting. |
| **7** | **Conv3D**   * **Dropouts:** 0.5 * **Learning Rate:** 0.01 * **Activation function :** Relu * **Image size:** 84\*84 * **Kernel Size:** (3,3,3) * **No. of image frames:** 15 * **Batch Size:** 20 | * categorical\_accuracy: 0.4238 * val\_categorical\_accuracy: 0.2800 | Introduced the dropouts as 0.25, 0.4, 0.5, 0.5 at each layer but we can clearly see that the model is underfitting. |
| **8** | **Conv3D**   * **Dropouts:** 0.25 * **Momentum:**0.9 * **Learning Rate:** 0.01 * **Activation function :** Relu * **Image size:** 84\*84 * **Kernel Size:** (3,3,3) * **No. of image frames:** 15 * **Batch Size:** 20 | * categorical\_accuracy: 0.8477 * val\_categorical\_accuracy: 0.8100 | This selection of features performed really well so far. Included momentum in the optimizer which helped in building the better model. |
| **9** | **Conv3D**   * **Dropouts:** 0.25 * **Momentum:**0.9 * **Learning Rate:** 0.001 * **Activation function :** Relu * **Image size:** 84\*84 * **Kernel Size:** (3,3,3) * **No. of image frames:** 15 * **Batch Size:** 20 | * categorical\_accuracy: 0.6109 * val\_categorical\_accuracy: 0.6800 | Changed the learning rate from 0.01 to 0.001 to see the affect but seems like the model is getting underfitted after doing the change. |
| **10** | **Conv3D**   * **No. of layers:** 7 * **Dropouts:** 0.25 * **Momentum:**0.9 * **Learning Rate:** 0.001 * **Activation function :** Relu * **Image size:** 84\*84 * **Kernel Size:** (3,3,3) * **No. of image frames:** 15 * **Batch Size:** 20 | * categorical\_accuracy: 0.7119 * val\_categorical\_accuracy: 0.5000 | Changed the number of layers from 5 to 7. But unfortunately model is underfitting. So we moved back to 5 layers and performed further testing. |
| **11** | **Conv3D**   * **Dropouts:** 0.25 * **Learning Rate:** 0.001 * **Activation function :** Adam * **Image size:** 84\*84 * **Kernel Size:** (3,3,3) * **No. of image frames:** 15 * **Batch Size:** 20 | * categorical\_accuracy: 0.2036 * val\_categorical\_accuracy: 0.1900 | Changed the optimizer from Relu to Adam so see the model accuracy. Clearly visible model has been failed miserably. Hence Adam won’t work. |
| **12** | **Conv2D+LSTM**   * **No. of layers:** 5 * **Momentum:**0.9 * **Learning Rate:** 0.001 * **Activation function :** Relu * **Image size:** 84\*84 * **Kernel Size:** (3,3,3) * **No. of image frames:** 15 * **Batch Size:** 20 | * categorical\_accuracy: 1.0000 * val\_categorical\_accuracy: 0.7600 | Performed Conv2D+LSTM with the same parameter as Conv3D but seems like the model is overfitting. To get rid from it, we included dropouts in all the layers. |
| **13** | **Conv2D+LSTM**   * **No. of layers:** 5 * **Dropouts: 0.5** * **Momentum:**0.9 * **Learning Rate:** 0.001 * **Activation function :** Relu * **Image size:** 84\*84 * **Kernel Size:** (3,3,3) * **No. of image frames:** 15 * **Batch Size:** 20 | * categorical\_accuracy: 0.8009 * val\_categorical\_accuracy: 0.6000 | To avoid the overfitting issue from LSTM, we have included dropouts in each layer but still the model is overfitting. The reason could be more number of parameters. |
| **14** | **Conv2D+GRU**   * **No. of layers:** 5 * **Momentum:**0.9 * **Learning Rate:** 0.001 * **Activation function :** Relu * **Image size:** 84\*84 * **Kernel Size:** (3,3,3) * **No. of image frames:** 15 * **Batch Size:** 20 | * categorical\_accuracy: 0.9970 * val\_categorical\_accuracy: 0.8300 | Finally, we have built the model using Conv2D+GRU with the same architecture but the model is still overfitting. Hence CNN+RNN is not suitable for this requirement. |
| **Final Model** | **Conv3D**   * **Dropouts:** 0.25 * **Momentum:**0.9 * **Learning Rate:** 0.01 * **Activation function :** Relu * **Image size:** 84\*84 * **Kernel Size:** (3,3,3) * **No. of image frames:** 15 * **Batch Size:** 20 | * categorical\_accuracy: 0.8477 * val\_categorical\_accuracy: 0.8100 | This selection of features performed really well so far. Included momentum in the optimizer which helped in building the better model. |