|  |  |  |  |
| --- | --- | --- | --- |
| Experiment Number | Model | Result | Decision + Explanation |
| 1 | Conv3D | Cannot handle this data type: (1, 1, 3), <f4 | The functions imresize() cannot accept inputs in the **float32** format. PIL library expects RGB images to be in the unit8 format. Therefore, the input was converted to **uint8** format and multiplied with 255 so as to normalize the images and then passed to the imresize() function. |
| 2 | Conv3D | OOM when allocating tensor with shape[618,16,5,160,160] and type float on /job:localhost/replica:0/task:0/device:GPU:0 by allocator GPU\_0\_bfc | Tried execution with only 5 training inputs. The number of parameters is too high for the model to handle. Therefore, added more dense layers and maxpooling layers to reduce the parameters to a reasonable size. |
| 3 | Conv3D | Input to reshape is a tensor with 384000 values, but the requested shape requires a multiple of 6272 | Cannot pass images of size (160, 160) due to it not being a multiple of 6272. Passing images of size (100, 100) solved the issue. |
| 3 | Conv3D | IndexError: index 663 is out of bounds for axis 0 with size 663 | Passing the length of t **(t=np.random.permutation(folder\_list))** instead of the source\_path solved the issue. Source\_path contains all the folders (in this case the train folder length was equal to 663). Passing source\_path caused the batch\_size to vary from one value to another (batch\_size should be constant). Passing the length of t, as mentioned above, fixed that as well. |
| 4 | Conv3D (Model 0) | categorical\_accuracy: 0.2899  val\_categorical\_accuracy: 0.4000 | We got the mentioned accuracies when the parameters were:  x = 15  y = 100 (*image height*)  z = 100 (*image width*)  batch\_size = 30  epochs = 15  It’s likely the model is unable to learn properly due to limited training images. Also the fact that the validation accuracy is more than the training accuracy could be of the following reason:  *When training, a percentage of the features are set to zero (25% in our case since we are using Dropout(0.25)). When testing, all features are used (and are scaled appropriately). So, the model at test time is more robust - and can lead to higher testing accuracies. (Source – stackoverflow [most upvoted answer])* |
| 5 | Conv3D (Model 1) | categorical\_accuracy: 0.4058  val\_categorical\_accuracy: 0.3250 | We got the mentioned accuracies when the parameters were:  x = 30 (*Changed from 15; passing the entire training images from each folder*)  y = 100  z = 100  batch\_size = 30  epochs = 15  A small step in the right direction. More training data was fed and therefore the training accuracy went up, but as a result validation accuracy went down. |
| 6 | Conv3D (Model 2) | categorical\_accuracy: 0.2667  val\_categorical\_accuracy: 0.4000 | We got the mentioned accuracies when the parameters were:  x = 30  y = 100  z = 100  batch\_size = 45 (*changed from 30*)  epochs = 20 (*changed from 15*)  Using a larger batch size can help in faster execution (due to parallel processing). Making it too large or too small impacts performance negatively. The validation accuracy turned out to be more than the training accuracy again. Therefore, more parameters need to be tuned. |
| 7 | Conv3D (Model 3) | categorical\_accuracy: 0.4444  val\_categorical\_accuracy: 0.4667 | We got the mentioned accuracies when the parameters were:  x = 30  y = 100  z = 100  num\_epochs = 25 (*changed from 20*)  batch\_size = 45  The accuracies obtained are very close to each other this time but the overall accuracy is low. More tuning of parameters is required. |
| 8 | Conv3D (Model 4) | categorical\_accuracy: 0.9343  val\_categorical\_accuracy: 0.8167 | We got the mentioned accuracies when the parameters were:  x = 30  y = 120 (*changed from 100*)  z = 120 (*changed from 100*)  batch\_size = 40 (*changed from 45*)  num\_epochs = 20 (*Reduced from 25 back to 20*)  Increasing the image size gives the model more accurate data to train on. Making it too large caused an OOM error. Therefore, we chose 120, 120 as the largest size that the model could handle.  Number of epochs was reduced from 25 to 20 as the loss was increasing after the 20th epoch (in model 4).  While the validation and training accuracy have improved considerably, the model is overfitting. |
| 9 | Conv3D (Model 5) | categorical\_accuracy: 0.7647  val\_categorical\_accuracy: 0.7000 | We got the mentioned accuracies when the parameters were:  x = 30  y = 120  z = 120  batch\_size = 40  num\_epochs = 20  The only difference in this model and the previous is that the dropout value is increased from 0.25 to 0.5 to reduce the overfitting faced by the previous model. Although the training and validation accuracy have noticeably gone down, the overfitting issue is resolved. |
| --------------- | ------------------ | ---------------------------- | ---------------------------------------------------------------------- |
| 10 | ConvGRU | categorical\_accuracy: 0.7197  val\_categorical\_accuracy: 0.7500 | Inputs:  x = 30  y = 120  z = 120  batch\_size = 40  num\_epochs = 20  Used the same parameters as the final 3D model’s as they are the best parameters. Used GRU as it has fewer number of gates compared to LSTM (also the number of trainable parameters is fewer).  The results are very close to the final 3D model’s (model 5) results. |