**Brief Summary: Gesture Recognition Project**

**Team: Ashish Veera**

**Deepak Kattil Veettil**

We have started off experimenting with the batch size in the initial few experiments and identified that 40 would be an optimal batch size to train the model.

In the initial few experiments, we used the CNN 3D architecture to train our model. Our strategy was to start with a less complex model with small number of parameters and then increase the complexity of the model if required. Most the initial experiments resulted in overfit models which lead to the introduction of dropouts and batch normalization which lead to reduction in overfitting but also decreased both the train and validation accuracy.

In our 10th experiment, we then decided to introduce additional CNN layer with 128 filters and increase the dropout ratio in the dense layer which lead to a model with Training accuracy of 90.31% and Validation accuracy of 88.33%, with 589253 total no of parameters. We further wanted to improve the accuracy of this model by increasing the number of neurons in the dense layer from 256 to 512 which lead to a model with Training accuracy of 94.46% and Validation accuracy of 90% but increased the no of parameters to 885701. Because of this increase in model complexity we decided to go with a stable model with less no of parameters which is model 10, that is also highlighted in the below table.

We also tried out CNN + RNN architecture to train the model, we also leveraged transfer learning by leveraging the mobilenet architecture. Even though the results from this architecture was promising the model 10 from CNN3d architecture showed better generalizability and had less no of parameters because of which model 10 was selected as our final model which is color coded in green in the below table.

Please note that along with this write up, we are also submitting the jupyter notebook as an evidence to show the attempts made to finalize the best model and also the model file .h5 generate from the final model is being submitted four your validation purposes. **Also, note that we have commented out the other models in our jupyter notebook file. Only our final model which is model 10 has been left uncommented in our jupyter notebook submission file.**

The below table summarizes the results of our experiments which spans across CNN3D, CNN + RNN architectures and Transfer learning process using mobilenet:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Exp No** | **Model Type** | **Model Name** | **Result** | **Decision + Explanation** | **No of Params** |
| 1 | Conv3D | model1 | Throws ResourceExhaustedError: when batch size is 60 | **Experimenting to identify optimal batch size.** Throws an error when batch size is 60. Limited the number of epochs to 2 since this was only to identify the optimal batch\_size. | 1,933,765 |
| 2 | Conv3D | model2 | Runs successfully with batch\_size=40. 40 seems to be a viable batch\_size | **Reduced the batch\_size to 40 since there was an error when batch\_size was 60 in the previous experiment**. Limited the number of epochs to 2 since this was only to identify the optimal batch\_size. | 1,933,765 |
| 3 | Conv3D | model3 | **Train Accuracy: 95% Val Accuracy: 78.3%**  **Result Analysis:** The model seems to be clearly overfitting. We will try to introduce dropouts in the next experiment | The strategy here is to start with a less complex model with small number of parameters and then increase the complexity of the model if required. | 1,677,701 |
| 4 | Conv3D | model4 | **Train Accuracy: 82% Val Accuracy: 78.33%**  **Result Analysis:** The overfitting seems to have reduced with the usage of dropouts | Since the model was overfitting in the previous experiment, **we have introduced dropouts in Conv layers and FC layer to reduce overfitting.** | 1,677,701 |
| 5 | Conv3D | Model5 | **Train Accuracy: 96.19% Val Accuracy: 85%**  **Result Analysis:** Even though the validation accuracy has improved to 85%, there is overfitting in the model since the training accuracy is 96.19%. | In the previous experiment the overfitting has reduced due to the usage of dropouts in the Conv and FC layer. **In this experiment we wanted to see the impact of using Batch normalization in the Conv layers and retained the dropout in the FC layer and removed dropouts from conv layers.** | 1,678,149 |
| 6 | Conv3D | model 6 | **Train Accuracy: 71.97% Val Accuracy: 68.33%**  **Result Analysis:** Accuracy dropped for both train and val sets | With the usage of batch normalization there is a significant improvement in the training and validation accuracy as seen in the previous experiment (exp no 5), but the model is still overfitting. So, we have decided to **reintroduce dropouts in the Conv layer** and retain BC in Conv layer. | 1,678,149 |
| 7 | Conv3D | model7 | **Train Accuracy: 87.20% Val Accuracy: 76.67%**  **Result Analysis:**  By adding more convolutional layers the train and val accuracy has improved but the model is still overfitting on the train set. | Since, the training and validation accuracy dropped in the previous experiment, we have decided to **add more convolutional layers to the model design** from experiment 6. | 1,021,045 |
| 8 | Conv3D | model8 | **Train Accuracy: 86.51% Val Accuracy: 83.33%**  **Result Analysis:** Significant decrease in overfitting when compared to model5 | **Enhancements to model5** We have decided to build on top of model5. The experiment no 5 had good validation accuracy but was overfitting. So, **we decided to increase the dropout ratio at the dense layer from 0.25 to 0.50** | 1,678,149 |
| 9 | Conv3D | model9 | **Train Accuracy: 80.28% Val Accuracy: 78.33%**  **Result Analysis:** Significant decrease in overfitting when compared to model7 | **Enhancements to model7** Continuation of model 7 by **increasing drop out ratio at Dense Layer** | 1,021,045 |
| **10** | **Conv3D** | **model10** | **Train Accuracy: 90.31% Val Accuracy: 88.33%  Result Analysis:**  The model generalizes pretty well and the no of training parameters is almost reduced by half | **Enhancements to model 8 Added an additional CNN layer with 128 filters** | **589,253** |
| 11 | Conv3D | model11 | **Train Accuracy: 94.46% Val Accuracy: 90%**  **Result Analysis:**  The model generalizes pretty well and the increase in val accuracy comes at the cost of increase in no of parameters when compared to model10. But still the total number of parameters is significantly less that the models yielded from the previous experiments | **Increased the no of parameters by increasing the no of neurons in the dense layer from 256 to 512 so that accuracy can be increased.** | 885,701 |
| 12 | Conv3D | model12 | **Train Accuracy: 82.35% Val Accuracy: 83.33%**  **Result Analysis:**  There is no increase in accuracy by increasing the image size and also results in an increase in the no of parameters | **Increased the image size to 100X100 and increased the drop out ratio to 0.60 in the dense layer to counter overfitting** | 2,655,173 |
| 13 | CNN + RNN (Leveraging Transfer Learning) | model13 | **Train Accuracy: 79.58% Val Accuracy: 70%**  **Result Analysis:**  The model is overfitting | **We are trying out the CNN + RNN architecture in this experiment. We are also leveraging transfer learning by using MobileNet architecture** | 3,446,533 |
| 14 | CNN + RNN (Leveraging Transfer Learning) | model14 | **Train Accuracy: 92.39% Val Accuracy: 75%**  **Result Analysis:**  The model is overfitting | **Increased the number of dense neurons to 256 from 64** | 3,459,973 |
| 15 | CNN + RNN (Leveraging Transfer Learning) | model15 | **Train Accuracy: 74% Val Accuracy: 70%**  **Result Analysis:**  The model is overfitting | **Since the model from the previous experiment was overfitting we have decided to add drop out in the CNN layer and changed the learning rate to 0.001** | 3,459,973 |