This is a sample write-up. The write-up need not be in tabular form.

It doesn’t state that ConvLSTM will give you better results than Conv3D. The explanation should be as detailed as possible so that the logic behind the decision is conveyed. Also, there are a lot of things you can experiment with in the generator function and elsewhere. Please do not forget to specify the exact metric values, here Accuracy which drives your decision.

You can draw inspiration from the concepts taught in the Industry demo in CNNs to experiment with the data and different architectures.

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D**  **Batch Size: 64**  **Ablation: 10%** | Throws Generator error | OOM when allocating tensor with shape[64,512,2,15,30]  Try reducing Batch Size to 48 |
| **2** | **Conv3D + Batch Size = 48**  **Ablation: 10%** | Model overfits  categorical\_accuracy: **0.8939**  val\_categorical\_accuracy: **0.5000** | Attempt to reduce overfitting by reducing number of model parameters, add augmentation on training set, add dropout |
| **3** | **Conv3D + Augumentation**  **Ablation: 10%** | Model overfits  categorical\_accuracy: **0.8333**  val\_categorical\_accuracy: **0.1000** | Performance has degraded.  Next, increase the amount of trainable data, add dropouts |
| **4** | **Conv3D (configuration changes) + Augumentation + More data**  **Ablation (training set only): 30%** | Model is still overfitting  categorical\_accuracy: **0.6970**  val\_categorical\_accuracy: **0.2000** | Performance is not increasing. Model is not learning well enough.  Next: Try increasing amount of training data, reduce complexity, add dropouts, change configurations |
| **5** | **Conv3D (configuration changes) +**  **GlobalAveragePooling3D + Augumentation + Full data + More epoch** | Model is still overfitting  categorical\_accuracy: **0.5792**  val\_categorical\_accuracy: **0.2300** | Performance is still very bad.  Next, reduce complexity a lot, add more Batch Normalizxation and add even more dropouts, change configurations |
| **6** | **Conv3D + Augumentation + More Dropout + Full Data** | categorical\_accuracy: **0.7738**  val\_categorical\_accuracy: **0.7** | Performance is lot better than before. Epoch 35 has best performance so far.  Try ConvLSTM as Conv3D not giving desired accuracy |
| **7** | **ConvLSTM2D + Conv2D** | categorical\_accuracy: **0.5234**  val\_categorical\_accuracy: **0.52** | Performance isn’t very good. Try changing optimizer to Adam |
| **8** | **ConvLSTM2D + Conv2D + Adan** | categorical\_accuracy: **0.7888**  val\_categorical\_accuracy: **0.7500** | Performance is better.  Next try GRU based Model. Also use VGG16 and imagine weights as the base model. |
| **9** | **VGG16 + GRU** | categorical\_accuracy: **0.3152**  val\_categorical\_accuracy: **0.4100** | Performance isn’t satisfactory.  Perhaps SGD optimizer isn’t configured properly. Next, try using Adam optimizer instead. |
| **10** | **VGG16 + GRU + Adam  (Augumentation is enabled)** | categorical\_accuracy: **0.9744**  val\_categorical\_accuracy: **0.86** | Performance is a lot better than other models so far.  Try adding more Dropout and reducing model parameters. |
| **11** | **VGG16 + GRU + Adam**  **(Augumentation is enabled)** | categorical\_accuracy: **0.7617**  val\_categorical\_accuracy: **0.8000** | Performance didn’t improve from previous model. Perhaps the Dropout was either too high or too many parameters were dropped.  For experiment only, try to train all parameters including VGG16 base model parameters. |
| **12** | **VGG16 + GRU + Train base layers** | categorical\_accuracy: **0.8507**  val\_categorical\_accuracy: **0.88** | Since we also trained all layers of VGG16 the training took a very long time.  But the performance of the model is the very good. |
| **Final Model** | **Experiment 10 (VGG16 + GRU + Adam)**  **Epoch 42**  **model-00042-0.18459-0.97436-0.42152-0.86000.h5** | categorical\_accuracy: **0.9744**  val\_categorical\_accuracy: **0.86** | This model has the best performance in terms of training size and accuracy among all the models created in this project.  This model uses VGG16 model with some added Dense layers as feature extractor.  Then GRU based RNN is used for learning from the sequence.  Potentially the model performance can be further improved by making it little bit less complex. But during Experiment 11 the performance which reduces model complexity of Experiment 10 reduced model performance. More experiments can be run to find better hyper-parameters in order to perform better than both Experiment 10 and 11.  Even though Experiment 12 performs better than Experiment 10 models, the model in this experiment was trained by retraining all VGG16 parameters and it takes a long time to train models in this way. Experiment 10 trains a lot faster because VGG16 layers were frozen and only custom Dense and GRU layers were trained.  Test Accuracy is calculated at **80.0%** |