|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| 0 | Conv3D | loss: 13.4174  categorical\_accuracy: 0.4139  val\_loss: 11.0994  val\_categorical\_accuracy: 0.6000 | Batch size: 16.  Generator function uses 13 frames  Normalization function is (image – mean)/standard deviation.  Number of Epoch: 1  Model:  model.add(Conv3D(32, (3, 3, 3), padding='same', input\_shape=(13,100,100,3)))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(2, 2, 2)))  model.add(Conv3D(64, (3, 3, 3), padding='same', input\_shape=(13,100,100,3)))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(2, 2, 2)))  model.add(Conv3D(128, (3, 3, 3), padding='same', input\_shape=(13,100,100,3)))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(2, 2, 2)))  model.add(Flatten())  model.add(Dense(512,kernel\_regularizer=l2(0.01)))  model.add(Activation('relu'))  model.add(Dense(num\_classes))  model.add(Activation('softmax'))  Observation:  Model doesn’t have any drop outs. Validation accuracy being higher means underfitting happening. |
| 1 | Conv3D | loss: 8.6052  categorical\_accuracy: 0.9940  val\_loss: 9.3862  val\_categorical\_accuracy: 0.6700 | Same as experiment – 0 but number of epochs: 10. Accuracy is very high compared to validation accuracy. Overfitting is happening. We should be adding dropouts. |
| 2 | Conv3D | loss: 22.0634  categorical\_accuracy: 0.2136  val\_loss: 20.1440  val\_categorical\_accuracy: 0.2500 | Same as experiment – 1. Adding Dropout for each layer.  Model:  model.add(Conv3D(32, (3, 3, 3), padding='same', input\_shape=(13,100,100,3)))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(2, 2, 2)))  model.add(Dropout(0.25))  model.add(Conv3D(64, (3, 3, 3), padding='same', input\_shape=(13,100,100,3)))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(2, 2, 2)))  model.add(Dropout(0.25))  model.add(Conv3D(128, (3, 3, 3), padding='same', input\_shape=(13,100,100,3)))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(2, 2, 2)))  model.add(Dropout(0.25))  model.add(Flatten())  model.add(Dense(512,kernel\_regularizer=l2(0.01)))  model.add(Activation('relu'))  model.add(Dropout(0.5))  model.add(Dense(num\_classes))  model.add(Activation('softmax'))  Observation:  Underfitting is happening as validation accuracy is higher than categorical accuracy |
| 3 | Conv3D | loss: 8.9456  categorical\_accuracy: 0.9297  val\_loss: 10.3551  val\_categorical\_accuracy: 0.5100 | Running experiment – 2 with more epochs. Number of epochs: 10.  Observation:  Validation accuracy is way less. It looks like overfitting is happening. |
| 4 | Conv3D | loss: 7.4838  categorical\_accuracy: 0.9732  val\_loss: 8.5864  val\_categorical\_accuracy: 0.7200 | Running experiment -2 with even more epochs. Number of epochs: 25  Observation:  Validation accuracy has increased but still less than categorical accuracy. Some amount of overfitting is happening. Loss has decreased. |
| 5 | Conv3D | loss: 8.8658  categorical\_accuracy: 0.9116  val\_loss: 9.6661  val\_categorical\_accuracy: 0.6600 | Changing two parameters of experiment – 2. Batch size has been increased from 16 to 32. Number of epochs is set to 20.  Observation:  Overfitting is happening. Loss has increased from experiment – 4. |
| 6 | Conv3D | loss: 7.4990  categorical\_accuracy: 0.9345  val\_loss: 8.4444  val\_categorical\_accuracy: 0.6500 | Changing batch size to 16 and other parameters remain same as experiment – 5.  Observation:  Loss has gone down while accuracy remains more or less the same. |
| 7 | Conv3D | loss: 9.1704  categorical\_accuracy: 0.9354  val\_loss: 10.2243  val\_categorical\_accuracy: 0.6100 | Batch size is changed to 64. Number of epochs set to 25. Rest remains same as experiment – 6  Observation:  Loss has increased. Validation accuracy has reduced. Overfitting is still happening |
| 8 | Conv3D | loss: 2.4811  categorical\_accuracy: 0.9155  val\_loss: 3.5696  val\_categorical\_accuracy: 0.6800 | Batch size is set to 32. Generator function is changed to use 12 frames. Generator function had some bugs in generating the batches so fixed that bug. Number of epochs is set to 75.  LR function used - ReduceLROnPlateau(monitor=’val\_loss', factor=0.2, patience=5, min\_lr=0.0001)  Observation:  Loss has considerably reduced but validation accuracy is still not good. |
| 9 | Conv3D | loss: 7.0280  categorical\_accuracy: 0.9911  val\_loss: 8.0081  val\_categorical\_accuracy: 0.7200 | Changing the normalization function. Normalization function used:  Image = image/255.  LR function is changed –  ReduceLROnPlateau(monitor='categorical\_accuracy', factor=0.2, patience=5, min\_lr=0.0001)  Observation:  Validation accuracy shows some improvement, but overfitting is happening. |
| 10 | Conv3D | loss: 8.5756  categorical\_accuracy: 0.9866  val\_loss: 9.3711  val\_categorical\_accuracy: 0.7900 | Changing the model to use Dropout selectively. Number of epochs: 20  Model:  model.add(Conv3D(16, (3, 3, 3), padding='same', input\_shape=(12,100,100,3)))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(2, 2, 2)))  model.add(Conv3D(32, (3, 3, 3), padding='same'))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(2, 2, 2)))  model.add(Conv3D(64, (3, 3, 3), padding='same'))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(2, 2, 2)))  model.add(Dropout(0.25))  model.add(Conv3D(128, (2, 2, 2), padding='same'))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(1, 1, 1)))  model.add(Dropout(0.25))  model.add(Flatten())  model.add(Dense(512,kernel\_regularizer=l2(0.01)))  model.add(Activation('relu'))  model.add(Dropout(0.5))  model.add(Dense(num\_classes))  model.add(Activation('softmax'))  Observation:  Good improvement with validation accuracy. |
| 11 | Conv3D | loss: 7.0741  categorical\_accuracy: 0.9970  val\_loss: 7.5436  val\_categorical\_accuracy: 0.8700 | Changed model again (Playing with Dropout used). Number of epochs: 50  Model:  model.add(Conv3D(16, (3, 3, 3), padding='same', input\_shape=(12,100,100,3)))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(2, 2, 2)))  model.add(Conv3D(32, (3, 3, 3), padding='same'))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(2, 2, 2)))  model.add(Conv3D(64, (3, 3, 3), padding='same'))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(2, 2, 2)))  model.add(Conv3D(128, (2, 2, 2), padding='same'))  model.add(Activation('relu'))  model.add(BatchNormalization())  model.add(MaxPooling3D(pool\_size=(1, 1, 1)))  model.add(Dropout(0.5))  model.add(Flatten())  model.add(Dense(512,kernel\_regularizer=l2(0.01)))  model.add(Activation('relu'))  model.add(Dropout(0.5))  model.add(Dense(num\_classes))  model.add(Activation('softmax'))  Observation:  Best validation accuracy so far.  Model file:  <https://drive.google.com/open?id=1IhL2lnaHGbj5KMxLglDCEv2A-78jZTCZ> |
| 12 | CNN-RNN | loss: 0.1655  categorical\_accuracy: 0.9825  val\_loss: 0.3759  val\_categorical\_accuracy: 0.8900 | Batch size: 32  Generator function uses 12 frames.  Normalization: image/255  Model:  cnn = Sequential()  cnn.add(Conv2D(32, (3, 3), padding='same', input\_shape=(100,100,3)))  cnn.add(BatchNormalization())  cnn.add(Activation('relu')) #can use elu as well  cnn.add(MaxPooling2D(pool\_size=(2,2)))  cnn.add(Conv2D(64, (3, 3), padding='same', input\_shape=(100,100,3)))  cnn.add(BatchNormalization())  cnn.add(Activation('relu')) #can use elu as well  cnn.add(MaxPooling2D(pool\_size=(2,2)))  cnn.add(Conv2D(128, (3, 3), padding='same', input\_shape=(100,100,3)))  cnn.add(BatchNormalization())  cnn.add(Activation('relu')) #can use elu as well  cnn.add(MaxPooling2D(pool\_size=(2,2)))  cnn.add(Conv2D(128, (3, 3), padding='same', input\_shape=(100,100,3)))  cnn.add(BatchNormalization())  cnn.add(Activation('relu')) #can use elu as well  cnn.add(MaxPooling2D(pool\_size=(2,2)))  cnn.add(Dropout(0.25))  cnn.add(Conv2D(128, (3, 3), padding='same', input\_shape=(100,100,3)))  cnn.add(BatchNormalization())  cnn.add(Activation('relu')) #can use elu as well  cnn.add(MaxPooling2D(pool\_size=(2,2)))  cnn.add(Dropout(0.25))  #model.add(LSTM(100,return\_sequences=True))  cnn.add(Flatten())  cnn.add(Dense(256, activation='relu'))  cnn.add(Dropout(0.5))  cnn.add(Dense(256, activation='relu'))  model = Sequential()  model.add(TimeDistributed(cnn, input\_shape=(12, 100, 100,3)))  model.add(GRU(12))  model.add(Dropout(.2)) #added  model.add(Dense(5, activation='softmax'))  Observation:  CNN-RNN model gives a better validation accuracy than CNN3D model.  Model file:  <https://drive.google.com/open?id=1ZnogwXuwwsFmZPyEp5hUzYA7yLH6beAT> |
|  |  |  |  |