Deep Learning Course Project - Gesture Recognition

**Team:** Arunima Dasgupta, Lakshmi J , Seema Gopalakrishnan, Sumit Pandey

**Architecture I: Convolutional 3D Neural Network (with a temporal dimension)**

**Mechanism:** 3D convolutions are a natural extension to the 2D convolutions in which case, the input to a 3D conv is a video (which is a sequence of 30 RGB images in this case). In 3D conv, you move the filter in three directions (x, y and z)

**Detailed Experiments:**

1. We started Conv3D with 4 layers with kernel size of 3\*3 with Adam as optimizer and with an image resolution of 80x80 and got “Out of memory” error.

**Outcome:**

The number of trainable parameters was around 9 digits. We experimented with different number of feature maps, and then reduced the number of feature map such that number of trainable parameters was around 8 digits.

1. The model was not trainable after doing the above changes

**Outcome**:

The number of trainable parameters was around 8 digits. Reduced the number of feature maps and the number of trainable parameters was reduced to 6 digits and increased the dropout.

1. Model was trainable but accuracy was very low around 0.20.

**Outcome:**

The accuracy started improving. This may be because

* Cropping the images were leading to loss of information as some images had the main person at one extreme side of the frame instead of being at the center.
* We chose 120\*120 as that was the smaller of the image sizes provided to us. Resizing to a size bigger than the smallest image sizes might have distorted the image or made it too grainy to be interpreted correctly.
* Normalization of the images were also done by dividing the dimensions by 255.

1. Model accuracy was increased to 0.40 but it was still low

**Outcome:**

Hyperparameters needed more fine tuning. Addition of kernel regularizer along with increase in the batch size and epoch size to 25 was done.

1. Tried with more nodes per conv3D layer starting with 16 and going up to 128 (in addition to the Dense layers)

**Outcome:**

The number of trainable parameters grew very large but did not improve accuracy significantly.

1. Added a dropout of 0.25 at each of the conv layers with no regularization at any layer

**Outcome:**

The model started underfitting. The validation loss was considerably higher than the training loss and same with accuracies (~.56-.60)

1. Removed dropouts from the initial conv layers ( 16 and 32 nodes) and retained them only for layers with 64 and more nodes

**Outcome:**

The model was not underfitting, but the validation accuracy didn’t improve as such (~ 0.55-0.62)

1. Adam optimizer gave an accuracy around 0.80 but with high validation loss

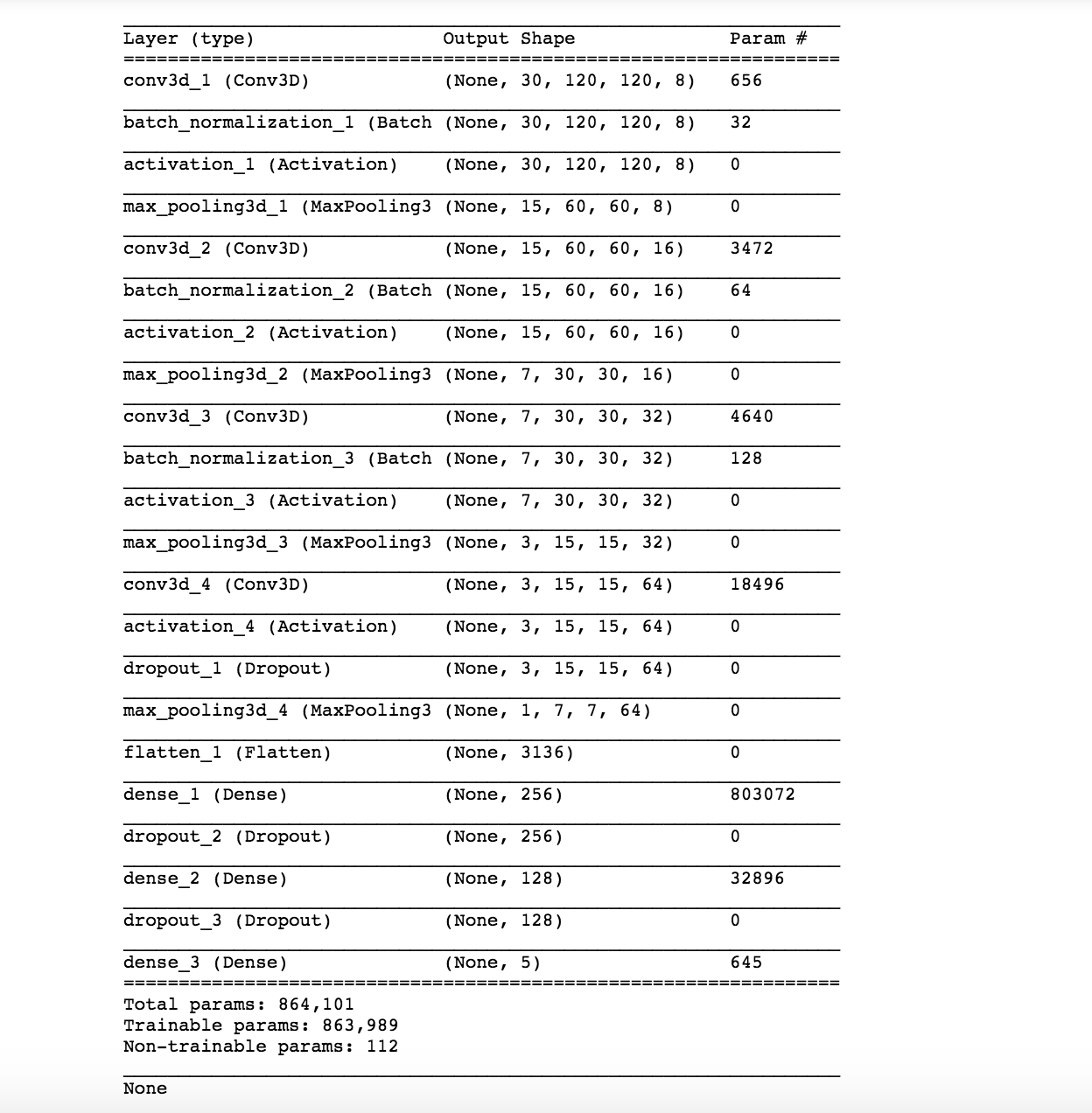
**Outcome**:

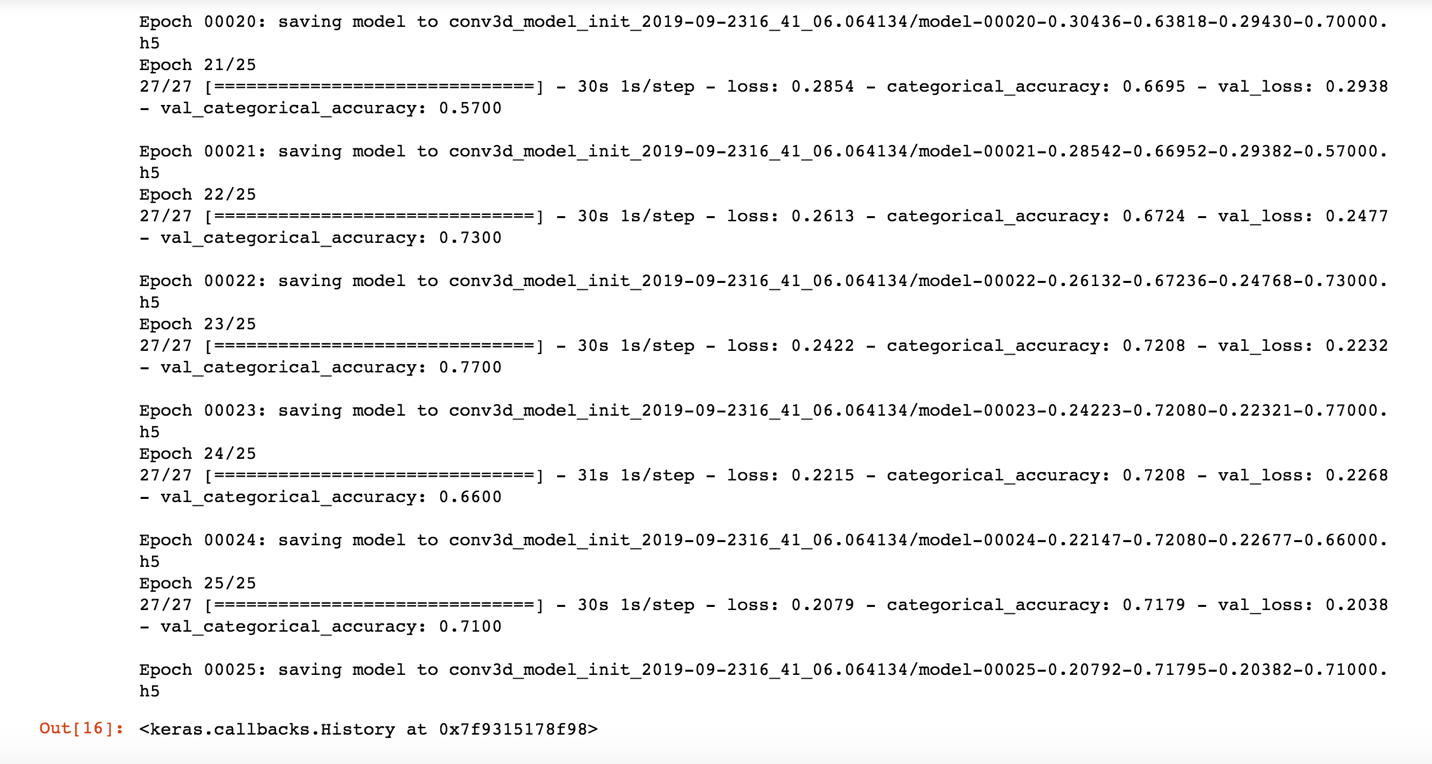
In order to decrease the validation loss, we experimented with different Optimizer like Nadam, sgd.

1. The optimizer SGD was chosen

**Outcome:**

The erratic behavior of the training accuracy and loss reduced. The training and validation losses started going down with each epoch (screenshot of model summary and last few epoch runs below)





**Architecture II: Transfer Learning with VGG16 Model that feeds into a Recurrent Neural Network**

**Mechanism:** Pass the images of a video through a CNN which extracts a feature vector for each image, and then pass the sequence of these feature vectors through an RNN.

**Detailed Experiments:**

1. While building the CNN-RNN (Simple RNN) model, the input to the RNN was incorrect and throwing this error:

*‘Input 0 is incompatible with layer lstm\_1: expected ndim=3, found ndim=2’*

**Explanation & Fix:**

Inputs to the RNN (LSTM) needs to be reshaped to be [samples, time steps, features]. We flattened the output from the Maxpool layers of the CNN and reshaped it and then fed it to the RNN. We specified the timesteps to be 30 while initializing the Simple layer.

1. Batch Data Generator Error while running Ablation experiment on this model. It was expecting an input of 4 dimensions (batch size, num\_of\_rows, num\_of\_cols, num\_of\_channels) but the generator was passing an extra dimension of the num\_of frames.

**Explanation & Fix:**

* Inputs to the 2D Conv Model does not need the extra temporal dimension of frames per video. It just needs the image shapes. We changed the batch generator code to handle data generation for our 2 different architectures (conv3D and CNN-RNN).
* For the CNN-RNN model, the batch data generator outputs data of this shape (batch\_size, no\_of\_rows, no\_of\_cols, no\_of\_channels) along with the batch labels.

1. The accuracy of vanilla 2D Conv + Simple RNN was very low (with SGD optimizer)

**Explanation & Fix:**

We were getting a best validation accuracy of around ~0.3 in this model with a high validation loss of ~0.8. This is because the vanilla model was not able to extract features from the images properly. The network also had 0.25 dropouts at after each conv layer which might have led to the network not learning as much as it should.

* We removed dropouts from the initial layers and kept them only for the layers with 64 and more nodes.
* We used Adam optimizer

The accuracy was still low.

1. We substituted LSTM layer in-place of the Simple RNN layer and added a L2 regularization param of 0.01, added back a dropout of 0.5 to the LSTM layer.

**Outcome:**

While trying to overfit on a smaller number of samples, the accuracy didn’t improve significantly, and it took a long time to run each epoch as the number of parameters had gone up. The validation accuracy did not go above 0.3. On removing the dropout at the LSTM layer, the accuracy improved slightly ~0.45

1. We substituted GRU layer in-place of the LSTM layer and added a L2 regularization param of 0.01, added back a dropout of 0.5 to the GRU layer.

**Outcome:**

While trying to overfit on a smaller number of samples, the validation accuracy improved to ~0.48 and the runtime of each epoch decreased significantly. This is because the GRU had lesser parameters (due to lesser gates as opposed to LSTM). We removed the dropout at the GRU layer and the validation accuracy increased to ~0.5, occasionally reaching ~0.6

1. We tried different optimizers like Nadam, Adam and SGD. With Nadam we used lr=0.002, beta\_1=0.9, beta\_2=0.999, epsilon=1e-08, schedule\_decay=0.004, we used lr=0.01, decay=1e-6, momentum=0.9, nesterov=True for SGG and used Adam with default params. We also used ReduceLROnPlateau to with the following params (monitor='val\_loss', factor=0.2, patience=2, cooldown=1, verbose=1)

**Outcome**:

Though Nadam increased validation accuracy while training the entire sample (~0.61-0.66), the validation loss was high. SGD gave good validation loss of around ~0.3 though the validation accuracy was around ~0.55-0.6. We decided to use SGD for the next model trials as it is more generic and simpler optimizer as opposed to Nadam and Adam which were more advanced and sometimes would fail to reach the optimal solution if not used with caution.

The ReduceLROnPlateau function helped a little in achieving better metrics. We experimented with its params by modifying the factor to 0.5 and adding min\_lr=0.0001 but they did not improve accuracy, so we decided to go with factor=0.2 without specifying a min learning rate.

1. Transfer learning with VGG16 model

**Outcome:**

We used transfer learning withVGG16 model as it had pre-trained weights and fed it into our GRU layer. Since VGG16 is of class Model() instead of Sequential(), we decided to copy all the layers (except the last few dense layers for predicting the 1000 classes) into a new Sequential model. This gave us the comfort of adding layers to it easily as we were doing till now. We added the GRU layer after the VGG16 block5\_pool (MaxPooling2D) then added the dense softmax layer for our video categorization. We tried overfitting it on a smaller sample size and got categorical\_accuracy of up to 1 and validation categorical accuracy of around ~0.84

1. Testing VGG16\_GRU model with full sample (Nadam Optimiser with lr=0.002, beta\_1=0.9, beta\_2=0.999, epsilon=1e-08, schedule\_decay=0.004).

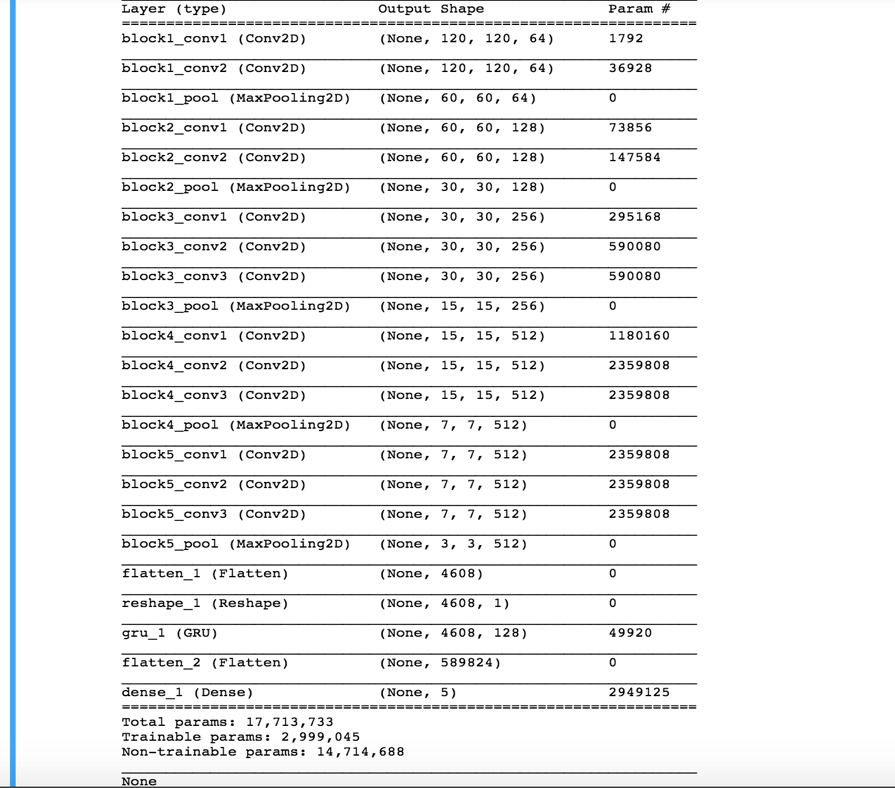
**Outcome**: The model started overfitting. We got categorical accuracy close to 1 (~0.97), a high loss of about ~0.88 with increasing validation loss of around ~1.04 and decreasing validation categorical accuracy of ~0.65- 0.67

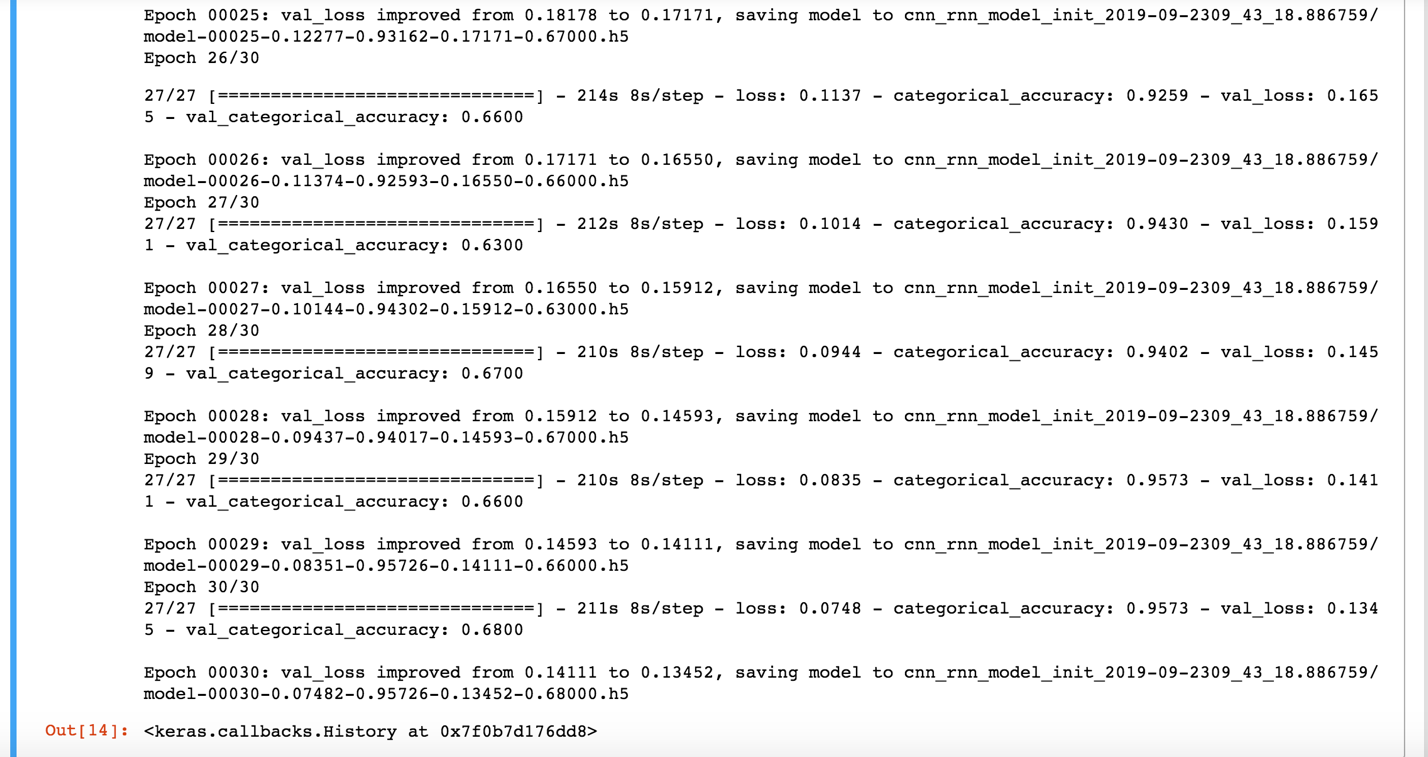
1. Added recurrent\_regularizer to GRU layer with L2=0.01

**Outcome**: The model stopped overfitting and both validation and training losses started going down with each epoch. The training accuracy stopped reaching the perfect 1.

1. Testing VGG16\_GRU model with full sample (SGD optimiser with lr=0.01, decay=1e-6, momentum=0.9, nesterov=True).

**Outcome**: The model became more stable and validation loss started going down. We ran 30 epochs and the validation loss and loss came down to 0.13 and 0.07 respectively. We got a great categorical accuracy of ~0.95 and validation categorical accuracy of ~0.68-0.7 (screen shot of model summary and last few epochs given below).





.h5 files link 🡪

https://drive.google.com/drive/folders/1pqSlayViUbcDYeQQfO\_qyqTj9cSdZAuK?usp=sharing