Gesture Recognition Case Study:

Group: Harshul Agarwal and Mikhil Varshney

This is the write-up for all models built to recognize the gestures correctly

Below experiments were performed to train models for Gesture Recognition,

Following are the results:

<table>
<tr><th>Experiment Number</th><th>Model</th><th>Result</th><th>Decision + Explanation</th></tr>
<tr>
<td>1</td>
<td><b>Conv3D</b>
[Vanilla]
<table>
<tr><td>Image-size</td><td>120X120</td></tr>
<tr><td>Layers</td><td>64-128 neurons with max pooling -> Flatten() -> Dropout (0.50) -> Dense (256) -> Dropout (0.50) -> Dense(5,softmax)</td></tr>
<tr><td>Dense Layer</td><td>256 neurons</td></tr>
<tr><td>[Con3D filter size/ strides/ padding</td><td>(3,3,3) / (1,1,1) / same
(3,3,3) / (1,1,1) / same</td></tr>
<tr><td>Max pool size / strides / padding</td><td>(2,2,2) / (2,2,2) / valid
(2,2,2) / (2,2,2) / valid</td></tr>
<tr><td>Optimizer</td><td>SGD</td></tr>
<tr><td>Batch size</td><td>60</td></tr>
</table>
</td>
<td>Out of Memory Error</td>
<td>
<ul>
<li>Created a vanilla model with 2 Conv3D layers.</li>
<li>Batch size was taken as 60 which gave Out of Memory Error.</li>
<li>Next Step is to reduce the batch size to 40.</li>
</ul>
</td>
</tr>
<tr>
<td>2</td>
<td><b>Conv3D</b>
[Vanilla]
<table>
<tr><td>Image-size</td><td>120X120</td></tr>
<tr><td>Layers</td><td>64-128 neurons with max pooling -> Flatten() -> Dropout (0.50) -> Dense (256) -> Dropout (0.50) -> Dense(5,softmax)</td></tr>
<tr><td>Dense Layer</td><td>256 neurons</td></tr>
<tr><td>[Con3D filter size/ strides/ padding</td><td>(3,3,3) / (1,1,1) / same
(3,3,3) / (1,1,1) / same</td></tr>
<tr><td>Max pool size / strides / padding</td><td>(2,2,2) / (2,2,2) / valid
(2,2,2) / (2,2,2) / valid</td></tr>
<tr><td>Optimizer</td><td>SGD</td></tr>
<tr><td>Batch size</td><td>40</td></tr>
</table>
</td>
<td>
<table>
<tr><td><b>Trainable Parameters</b></td><td><b>88,701,701</b></td></tr>
<tr><td><b>Training Accuracy</b></td><td><b>87.30 %</b></td></tr>
<tr><td><b>Validation Accuracy</b></td><td><b>57.14 %</b></td></tr>
</table>
</td>
<td>
<ul>
<li>This vanilla model with 2 conv3D layers performs well on training accuracy but have very low validation accuracy.</li>
<li>Overfitting as well.</li>
<li>Next step is to change the filter size to a small value (2,2,2) to have better validation accuracy</li>
</ul>
</td>
</tr>
</table>

<table>
<tr>
<td><b>3</b></td>
<td><b>Conv3D</b>
[Vanilla] (changed filter size to (2,2,2), Otherwise same as model 1</td>
<td><table>
<tr><td><b>Trainable Parameters</b></td><td><b>88,483,717</b></td></tr>
<tr><td><b>Training Accuracy</b></td><td><b>85.71 %</b></td></tr>
<tr><td><b>Validation Accuracy</b></td><td><b>75 %</b></td></tr>
</table></td>
<td><ul>
<li>As expected smaller filter size improved both accuracies.</li>
<li>Overfitting got reduced by a large extent.</li>
<li>Next step is to increase filter size to (5,5,5) to validate that bigger filter will perform bad.</li>
</ul></td>
</tr>
<tr>
<td><b>4</b></td>
<td><b>Conv3D</b>
[Vanilla] (changed filter size to (5,5,5), Otherwise same as model 1</td>
<td><table>
<tr><td><b>Trainable Parameters</b></td><td><b>89,523,333</b></td></tr>
<tr><td><b>Training Accuracy</b></td><td><b>28.71 %</b></td></tr>
<tr><td><b>Validation Accuracy</b></td><td><b>31 %</b></td></tr>
</table></td>
<td><ul>
<li>As expected bigger filter size worsened both accuracies.</li>
<li>Next step is to try for lower image size for reducing training time, reverting to filter size of 2</li>
</ul></td>
</tr>
<tr>
<td><b>5</b></td>
<td><b>Conv3D</b>
[Vanilla] (changed image size 80X80)
<table>
<tr><td>Image-size</td><td>80X80</td></tr>
<tr><td>Layers</td><td>64-128 neurons with max pooling -> Flatten() -> Dropout (0.50) -> Dense (256) -> Dropout (0.50) -> Dense(5,softmax)</td></tr>
<tr><td>Dense Layer</td><td>256 neurons</td></tr>
<tr><td>[Con3D filter size/ strides/ padding</td><td>(2,2,2) / (1,1,1) / same
(2,2,2) / (1,1,1) / same</td></tr>
<tr><td>Max pool size / strides / padding</td><td>(2,2,2) / (2,2,2) / valid
(2,2,2) / (2,2,2) / valid</td></tr>
<tr><td>Optimizer</td><td>SGD</td></tr>
</table></td>
<td><table>
<tr><td><b>Trainable Parameters</b></td><td><b>88,492,101</b></td></tr>
<tr><td><b>Training Accuracy</b></td><td><b>83.34 %</b></td></tr>
<tr><td><b>Validation Accuracy</b></td><td><b>71.43 %</b></td></tr>
</table></td>
<td><ul>
<li>Smaller image size as expected performed worse than larger.</li>
<li>Switch back to (120X120) size as model perform best on it.</li>
<li>Next Step is to reduce overfitting by trying different optimized (Adam)</li>
</ul></td>
</tr>
</table>

<table>
<tr>
<td>6</td>
<td><b>Conv3D</b>
[Vanilla] (changed optimizer to Adam)
<table>
<tr><td>Image-size</td><td>120X120</td></tr>
<tr><td>Layers</td><td>64-128 neurons with max pooling -> Flatten() -> Dropout (0.50) -> Dense (256) -> Dropout (0.50) -> Dense(5,softmax)</td></tr>
<tr><td>Dense Layer</td><td>256 neurons</td></tr>
<tr><td>[Con3D filter size/ strides/ padding</td><td>(2,2,2) / (1,1,1) / same
(2,2,2) / (1,1,1) / same</td></tr>
<tr><td>Max pool size / strides / padding</td><td>(2,2,2) / (2,2,2) / valid
(2,2,2) / (2,2,2) / valid</td></tr>
<tr><td>Optimizer</td><td>Adam</td></tr>
</table>
</td>
<td>
<table>
<tr><td><b>Trainable Parameters</b></td><td><b>88,542,405</b></td></tr>
<tr><td><b>Training Accuracy</b></td><td><b>40.65 %</b></td></tr>
<tr><td><b>Validation Accuracy</b></td><td><b>28.57 %</b></td></tr>
</table>
</td>
<td>▪ Performed worse than SGD optimized.
▪ Will continue with SGD optimized.
▪ Next Step:- Add batch Normalization layer and revert back to SGD optimizer</td>
</tr>
<tr>
<td>7</td>
<td><b>Conv3D</b>
[Vanilla + Batch Normalization]
<table>
<tr><td>Image-size</td><td>120X120</td></tr>
<tr><td>Layers</td><td>64-128 layer neurons with Batch Norm and max pooling -> Flatten() -> Dropout (0.50) -> Dense (256) -> Dropout (0.50) -> Dense(5,softmax)</td></tr>
<tr><td>Dense Layer</td><td>256 neurons</td></tr>
<tr><td>[Con3D filter size/ strides/ padding</td><td>(2,2,2) / (1,1,1) / same
(2,2,2) / (1,1,1) / same</td></tr>
<tr><td>Max pool size / strides / padding</td><td>(2,2,2) / (2,2,2) / valid
(2,2,2) / (2,2,2) / valid</td></tr>
<tr><td>Optimizer</td><td>SGD</td></tr>
<tr><td></td><td></td></tr>
</table>
</td>
<td>
<table>
<tr><td><b>Trainable Parameters</b></td><td><b>88,548,573</b></td></tr>
<tr><td><b>Training Accuracy</b></td><td><b>89.34 %</b></td></tr>
<tr><td><b>Validation Accuracy</b></td><td><b>75.43 %</b></td></tr>
</table>
</td>
<td>▪ Batch normalization improved the model performance and accuracy slightly.
▪ Next step, try adding dropout (0.25) layers between 64-128 neurons layer of model.</td>
</tr>
<tr>
<td>8</td>
<td><b>Conv3D</b>
[Vanilla + 3rd layer + Conv3D Dropout between 2nd and 3rd layer]
<table>
<tr><td>Image-size</td><td>120X120</td></tr>
<tr><td>Layers</td><td>32 neurons with max pooling and batch Norm -> Dropout (0.25)-> 64 neurons with max pooling -> Dropout (0.25)-> 128 neurons with max pooling -> Flatten () -> Dropout (0.50) -> Dense (256) -> Dropout (0.50) -> Dense(5,softmax)</td></tr>
<tr><td>Dense Layer</td><td>256 neurons</td></tr>
<tr><td>[Con3D filter size/ strides/ padding</td><td>(2,2,2) / (1,1,1) / same
(2,2,2) / (1,1,1) / same</td></tr>
<tr><td>Max pool size / strides / padding</td><td>(2,2,2) / (2,2,2) / valid
(2,2,2) / (2,2,2) / valid</td></tr>
<tr><td>Optimizer</td><td>SGD</td></tr>
<tr><td></td><td></td></tr>
</table>
</td>
<td>
<table>
<tr><td><b>Trainable Parameters</b></td><td><b>92,436,531</b></td></tr>
<tr><td><b>Training Accuracy</b></td><td><b>87.36 %</b></td></tr>
<tr><td><b>Validation Accuracy</b></td><td><b>79.67 %</b></td></tr>
</table>
</td>
<td>▪ Adding additional layer to improve training accuracy and Adding dropouts between 2nd and 3rd layer to reduce the overfitting, increased the model validation accuracy slightly</td>
</tr>
</table>

<table>
<tr><td colspan="4"><b><u>Conv3D Conclusion</u></b>
- <b>Dropouts are helping in reducing the overfitting by reducing training accuracy.</b>
- <b>(2,2,2) filter size is performing the best.</b>
- <b>2 Conv3D layers are sufficient for achieving high training accuracy. But the model is overfitting.</b>
- <b>3 Conv3D layers (model 8 with 256 layers in 3rd layer) can achieve a good fit and we can improve validation accuracy by using more training data</b></td></tr>
<tr><td><b>9</b></td><td><b>CNN(Conv2D) + RNN(GRU)</b>
Conv3D+BatchNorm+MaxPool repeated thrice followed by single GRU layer + 1 Dense layers. Dropout after each of the dense layers and GRU layer and then connect to the final output layer.</td><td><b>Trainable Parameters:</b> 2,429,285
<b>Val Accuracy :</b> 56.25%
<b>Training Accuracy</b>: 98.47%</td><td>▪ We tried to create a CNN network that can learn & extract features and then fed that into RNN to perform classification of the gesture
▪ The Current Model with 3 Con2D layer each followed by Maxpooling, batch Normalization & then by Flatten & dense layers with dropouts overfits & achieves a validation accuracy of 56.25%
▪ We can see that we need more dense CNN network so we will try transfer learning with predefined N/W</td></tr>
</table>

10	**CNN(Resnet) Earlier layers as fixed (last 5 layers trainable) + RNN(GRU)**	**Trainable Parameters:** 6,560,773 **Val Accuracy :** 37.5% **Training Accuracy:** 89.6%	▪ Trainable parameters increase with RESNET. ▪ We only train the last 5 layers to keep params and training time low. ▪ The Validation accuracy as observed is low. ▪ As feature extraction is responsibility of CNN, not training the weights of entire Resnet does not result in good validation accuracy.

11	**CNN(MobileNet)All parameters trainable)+ 2 RNN(GRU) layers having dropout layers** GRU (32, return_sequences = True) -> Dropout (0.5) -> GRU (64) -> Dropout (0.25)	**Trainable Parameters :** 3,331,653 **Val Accuracy :** 89.25% **Training Accuracy:** 99.09%	▪ Now lets use another Predefined network for transfer learning : MobileNet ▪ We will train all the weights as seen from previous models that works better. ▪ This is much better than Resnet. But too much fluctuation can be seen in later epochs and overfitting is also noticeable.

12. Final Model	**CNN(MobileNet)All parameters trainable) + 2 RNN(GRU) layers having dropout layers + L2 regularization in dense layers** GRU (32, return_sequences = True) -> Dropout (0.25) -> GRU (64) -> Dropout (0.25) + Add L2 regularization	**Trainable Parameters :** 3,331,653 **Val Accuracy :** 95.83% **Training Accuracy:** 99.41%	▪ Now adding: ▪ Kernel regularizer = L2(0.01) ▪ Activity Regularizer = L2(0.01) ▪ This is the best model with less over fitting and the accuracy curve is converging.