Gesture Recognition - Project Write Up

Generator function evaluation

We have developed the Generator function based on the steps provided in the starter code. We have analyzed the frame per video for each gesture. We did the resizing of the frames per video and normalized it.

Ablation Runs:-
Since running the whole data while performing hyper-parameter tuning, took large amounts of time. We choose to run ablation runs with an ablation size of 100 and 10 epochs for selecting the hyper parameters.

GPU Limitations:-
We reached a constraint on GPU which depended on batch_size, image_size, number of images per video, and number of filters in the first CNN layer.
We got to the values that used the maximum GPU available to us, and when we wanted to increase one of these values to further optimize, we had to reduce another value.

Model Building:-

We first starting with a two layer 3D CNN layer followed by a single dense layer and output layer. In order to select the number of filters in the CNN layer we performed the following ablation runs:

Ablation Model Observations:

Experiment Number	**Model**	**Result**	**Decision + Explanation**
Ablation - 1	Conv3D CNN layer 1-8 filters CNN layer 2-16 filters Dense Layer-32 neurons	Val_Categorical_Accuracy: 0.3100	Increase filter size
Ablation - 2	Conv3D CNN layer 1-16 filters CNN layer 2-32 filters Dense Layer-64 neurons	Val_Categorical_Accuracy: 0.3400	Accuracy has improved. Increase filter size
Ablation - 3	Conv3D CNN layer 1-32 filters CNN layer 2-64 filters Dense Layer-128 neurons	Val_Categorical_Accuracy: 0.4200	Accuracy has improved. Increase filter size
Ablation - 4	Conv3D CNN layer 1-64 filters CNN layer 2-128 filters Dense Layer-256 neurons	Val_Categorical_Accuracy: 0.3500	Accuracy has decreased. Ablation – 3 has the best filter values.

Note: The above ablation runs used batch_size = 50, lr = 0.001, and optimizer = SGD.

We choose the model from the Ablation run 3. We further performed ablation runs in order to select the best batch_size, image size, normalization technique, number of images per video, activation function. Given below are the different values tried for these hyper parameters and the best values for each:

- **Batch_size:** We tried 25, 50, 100. The best value was 50.
- **Image_size:** We tried 80*80, 120*120, 160*160. The best value was **120*120**.
- **Number of images per video:** We tried 30(all images), 15(alternate images), 10(1 out of 3 images). The best value was **15 i.e., alternate images**.
- **Normalization technique:** We tried dividing by 255, dividing by max-min, and diving by $95^{th} – 5^{th}$ percentile . The best technique was **Dividing by max-min**.
- **Activation function:** We tried "elu" and "relu". The best function was **"relu"**.

We the above selected model we further performed hyper parameter tuning to select the best learning rate and optimizer. Given below are the ablation run for this:

Ablation Model Observations:

Experiment Number	**Model**	**Result**	**Decision + Explanation**
Ablation - 1	Conv3D Optimizer: SGD LR: 0.001	Val_Categorical_Accuracy: 0.4200	Try Adam Optimizer.
Ablation – 2	Conv3D Optimizer: Adam LR: 0.001	Val_Categorical_Accuracy: 0.3900	Try RMSProp Optimizer.
Ablation – 3	Conv3D Optimizer: RMSProp LR: 0.001	Val_Categorical_Accuracy: 0.3700	Increase the LR.
Ablation – 4	Conv3D Optimizer: SGD LR: 0.002	Val_Categorical_Accuracy: 0.3500	Reduce the LR.
Ablation - 5	Conv3D Optimizer: SGD LR: 0.00075	Val_Categorical_Accuracy: 0.4000	The best hyper parameter are SGD(lr = 0.001)

Using the above mention model and hyper parameters we build our first model. We used the whole data, and ran 50 epochs. Given below are the observations:

Model Observations:

Model - 1

Parameters

1. Conv 3d filter size = (3,3,3)
2. Conv 3d layers = 2 (layer 1 - 32 filters, layer 2 – 64 filters)
3. Dropout = No
4. Batch size = 50
5. No. of epochs = 50
6. Batch Normalization = Provided at both the Conv 3d layers
7. Dense Layers = 1 dense layers (128 neurons)
8. Trainable Parameters = 4,212,293
9. Optimizer = SGD
10. LR = 0.001

Output

1. Train Categorical Accuracy = ~0.97
2. Validation Categorical Accuracy = ~0.68
3. Train Loss = ~0.08
4. Validation Loss = ~0.85

Observation

The train accuracy is much higher and train loss is much lower than their validation counterparts. Therefore the model is over fitting.

Solution

Hence we will add Dropouts in the dense layer to reduce the over fitting.

Model – 2

Similar to the previous model we performed ablation run for selection the hyper parameter Optimizer and LR. Given below hyper parameter values tried and the best hyper parameters achieved:

- **Optimizer:** We tried SGD, Adam and RMSProp. And the best optimizer was **Adam.**
- **LR:** We tried 0.001, 0.0008, 0.0005, and 0.0003. And the best LR was **0.0005.**

Parameters

1. Conv 3d filter size = (3,3,3)
2. Conv 3d layers = 2 (layer 1 - 32 filters, layer 2 – 64 filters)
3. Dropout = Yes. Dropout of 0.5 after the dense layers and flatten layer.
4. Batch size = 50
5. No. of epochs = 50
6. Batch Normalization = Provided at all the Conv 3d layers
7. Dense Layers = 1 dense layers (128 neurons)

8. Trainable Parameters = 4,212,293
9. Optimizer = Adam
10. LR = 0.0005

Output

1. Train Categorical Accuracy = ~0.6
2. Validation Categorical Accuracy = ~0.55
3. Train Loss = ~1.0
4. Validation Loss = ~1.25

Observation

The accuracy is low. The validation loss is very high.

Solution

We decided to increase the number of conv3d layers by 1 with increased filter size to capture more patterns.

Model – 3

We added another 3D CNN layer with 128 filters. We also increased the dense layer neurons to 256.

We performed ablation run for selection the hyper parameter Optimizer and LR. Given below hyper parameter values tried and the best hyper parameters achieved:

- **Optimizer:** We tried SGD, Adam and RMSProp. And the best optimizer was **Adam.**
- **LR:** We tried 0.001, 0.0008, 0.0005, and 0.0003. And the best LR was **0.0005.**

Parameters

1. Conv 3d filter size = (3,3,3)
2. Conv 3d layers = 3 (layer 1 - 32 filters, layer 2 – 64 filters, layer 3 - 128)
3. Dropout = No
4. Batch size = 50
5. No. of epochs = 50
6. Batch Normalization = Provided at the Conv 3d layers.
7. Dense Layers = 1 dense layer (256 neurons)
8. Trainable Parameters =1,459,648
9. Optimizer = Adam
10. LR = 0.0005

Output

1. Train Categorical Accuracy = ~1.0

2. Validation Categorical Accuracy = ~0.8
3. Train Loss = ~0
4. Validation Loss = ~0.9

Observation

The validation Accuracy has increased from the earlier models. The model is also over fitting a lot, since the train accuracy is much higher than validation accuracy. But we will first focus on further increasing the validation accuracy.

Solution

We decided to increase the number of conv3d layers by 2 with increased filter size to capture more patterns.

Model – 4

We added 2 more 3D CNN layer with 256 filters each. We also increased the dense layer neurons to 512.

W performed ablation run for selection the hyper parameter Optimizer and LR. Given below hyper parameter values tried and the best hyper parameters achieved:

- **Optimizer:** We tried SGD, Adam and RMSProp. And the best optimizer was **RMSProp.**
- **LR:** We tried 0.001, 0.0008, and 0.0005. And the best LR was **0.0008.**

Parameters

1. Conv 3d filter size = (3,3,3)
2. Conv 3d layers = 5 (layer 1 - 32 filters, layer 2 – 64 filters, layer 3 – 128, layer 4 – 256 filters, layer 5 - 256 filters)
3. Dropout = No
4. Batch size = 50
5. No. of epochs = 50
6. Batch Normalization = Provided at the Conv 3d layers.
7. Dense Layers = 1 dense layer (512 neurons)
8. Trainable Parameters =4,118,213
9. Optimizer = RMSProp
10. LR = 0.0008

Output

1. Train Categorical Accuracy = ~0.99
2. Validation Categorical Accuracy = ~0.82

3. Train Loss = ~0.02
4. Validation Loss = ~0.8

Observation

The model is over fitting with the train accuracy much higher than the validation accuracy. We need to reduce the over fitting.

Solution

We decided to add a Dropout of 0.5 to the dense layers, so that some of the features are removed randomly and prevent over fitting.

Model - 5

Parameters

1. 1 Conv 3d filter size = (3,3,3)
2. Conv 3d layers = 5 (layer 1 - 32 filters, layer 2 – 64 filters, layer 3 – 128, layer 4 – 256 filters, layer 5 - 256 filters)
3. Dropout = Yes. Dropout of 0.5 after the dense layers and flatten layer.
4. Batch size = 50
5. No. of epochs = 50
6. Batch Normalization = Provided at the Conv 3d layers.
7. Dense Layers = 1 dense layer (512 neurons)
8. Trainable Parameters =4,118,213
9. Optimizer = RMSProp
10. LR = 0.0008

Output

1. Train Categorical Accuracy = ~0.98
2. Validation Categorical Accuracy = ~0.9
3. Train Loss = ~0.03
4. Validation Loss = ~0.4

Observation

The validation accuracy has increased significantly. There is still some over fitting happening which we will try to reduce.

Solution

We decided to add a Dropout of 0.2 to some of the 3D CNN layers, so that some of the features are removed randomly and prevent over fitting.

Model - 6

Parameters

1. 1 Conv 3d filter size = (3,3,3)
2. Conv 3d layers = 5 (layer 1 - 32 filters, layer 2 – 64 filters, layer 3 – 128, layer 4 – 256 filters, layer 5 - 256 filters)
3. Dropout = Yes. Dropout of 0.5 after the dense layers and flatten layer. And dropout of 0.2 in 3D CNN layers 2 to layer 5.
4. Batch size = 50
5. No. of epochs = 50
6. Batch Normalization = Provided at the Conv 3d layers.
7. Dense Layers = 1 dense layer (512 neurons)
8. Trainable Parameters =4,118,213
9. Optimizer = RMSProp
10. LR = 0.0008

Output

5. Train Categorical Accuracy = ~0.91
6. Validation Categorical Accuracy = ~0.75
7. Train Loss = ~0.1
8. Validation Loss = ~1.1

Observation

The train accuracy has reduced, but the validation accuracy has reduced much further.

Solution

Hence we will choose the Model 5 as our final model.

Conclusion:

There our final model is Model 5. The best .h5 file from the model has a validation loss of 0.4016 and validation categorical accuracy of 0.92.