# Gesture Detection

We experimented 100+ times using 500 premium compute units from Google Colab which gave us NVIDIA A100 GPU (40 GB memory) to experiment quickly.

**Note:** Listed accuracies are validation accuracies unless stated otherwise.

**Experiments:**

**Experiment 1**

**Action:** Created a simple 3D CNN Model inspired from VGGNet-16.

**Result:** Throws ResourceExhaustedError.

**Decision:** Reduce batch size.

**Experiment 2**

**Action:** Reduced batch size.

**Result:** Throws ResourceExhaustedError.

**Decision:** Pick only few images from the video sequence.

**Experiment 3**

**Action:** Picked 15 images from the video sequence.

**Result:** ValueError - One of the dimensions in the output is <= 0 due to downsampling in conv3d\_2.

**Decision:** Remove the layout causing problem.

**Experiment 4**

**Action:** Removed the layer which was reducing dimension to negative.

**Result:** Accuracy 42.86%

**Decision:** Increase epoch.

**Experiment 5**

**Action:** Increased epoch.

**Result:** Accuracy 47.62%

**Decision:** Add image augmentation to generator to more training data.

**Experiment 6**

**Action:** Added image augmentation logic to generator

**Result:** Accuracy 23.81%

**Decision:** Increase model complexity by adding new layers in VGGNet-16 way

**Experiment 7**

**Action:** Added a new convolutional layer with more kernel

**Result:** Accuracy 23%

**Decision:** Make the model more complex

**Experiment 8**

**Action:** Added multiple convolutional & dense layers

**Result:** Accuracy 32%, but overfitting with training accuracy 20.83%

**Decision:** Add Batch Normalization layer to reduce overfitting which can also allow generalization

**Experiment 9**

**Action:** Added Batch Normalization layer just after the first convolutional layer

**Result:** Accuracy 26%

**Decision:** Make model more complex

**Experiment 10**

**Action:** Added more layers to the model

**Result:** Accuracy 16%

**Decision:** This doesn’t make sense. Try training other types of models.

**Experiments 11-15**

**Action:** Created multiple basic models with TimeDistributed, ConvLSTM2d, LSTM and GRU.

**Result:** Accuracy ranged from 11% to 26%.

**Decision:** This doesn’t make sense. There’s got to be problem with data. Debug generator.

**Experiment 16**

**Action:** Initialized a sample generator & visualized batches.

**Result:** Images of a video aren’t in sequence.

**Decision:** Fix the image order issue.

**Experiment 17**

**Action:** It turns out that os.listdir() doesn’t sort based on file name. Added sort logic.

**Result:** Images a video are now in sequence.

**Decision:** Try training all models again.

**Experiment 18-21**

**Action:** Trained all 3D CNN models

**Result:** Accuracy ranged from 17% to 30%

**Decision:** Create new model with smaller number of kernels in the first convolutional layer & follow VGGNet-16 architecture

**Experiment 22**

**Action:** Created a new simpler 3D CNN model

**Result:** Accuracy 44%

**Decision:** Create new model with even smaller number of kernels in the first convolutional layer & follow VGGNet-16 architecture

**Experiment 23**

**Action:** Created a new simpler 3D CNN model

**Result:** Accuracy 21%

**Decision:** Add more layers to these two models to make them more complex

**Experiment 24**

**Action:** Made previous two models more complex

**Result:** 12% & 24% accuracies respectively

**Decision:** Try training other types of models on the new fixed sequence data

**Experiment 25-28**

**Action:** Trained all other models.

**Result:** Accuracies ranged from 18% to 23% (Best model: GRU)

**Decision:** Try transfer learning before making GRU model more complex

**Experiment 29-30**

**Action:** Created LSTM & GRU models with ResNet50

**Result:** GRU model gave 16% accuracy & LSTM model gave 21% accuracy.

**Decision:** It occurred to me that only two images from the sequence can tell what gesture it is. As starting & ending frames aren’t known, use some more frames, but not 15.

**Experiment 31**

**Action:** Chose 6 frames from the sequence & tried 3D CNN model.

**Result:** Negative dimension error in the second layer.

**Decision:** Use ‘SAME’ padding

**Experiment 32**

**Action:** Trained simplest 3D CNN model by using ‘SAME’ padding

**Result:** Accuracy 45% with oscillating accuracies during training

**Decision:** Decrease learning rate to stabilize accuracies

**Experiment 33**

**Action:** Decreased learning rate

**Result:** Accuracy 46%

**Decision:** Try 5 frames instead of 6

**Experiment 34**

**Action:** Used 5 frames

**Result:** Accuracy 50%

**Decision:** Try different types of models on 5 frames

**Experiment 35**

**Action:** Trained TimeDistributed + ConvLSTM2D model

**Result:** Accuracy 23%

**Decision:** Make the model more complex

**Experiment 36**

**Action:** Added more layers to capture more information

**Result:** Accuracy 17%

**Decision:** Make the model more complex

**Experiment 37**

**Action:** Added more layers to capture more information

**Result:** Accuracy 23%

**Decision:** Make the model more complex

**Experiment 38**

**Action:** Added more layers to capture more information

**Result:** Accuracy 22%

**Decision:** Try other types of models

**Experiment 39**

**Action:** Trained vanilla TimeDistributed model

**Result:** Accuracy 9%

**Decision:** Try other types of models

**Experiment 40**

**Action:** Trained GRU model

**Result:** Accuracy 22%

**Decision:** Increase number of GRU units

**Experiment 41**

**Action:** Added more GRU units

**Result:** Accuracy 24%

**Decision:** Increase number of GRU units

**Experiment 42**

**Action:** Added more GRU units

**Result:** Accuracy 11%

**Decision:** Try ResNet50 pre-trained model

**Experiment 43**

**Action:** Trained ResNet50 + LSTM model

**Result:** Accuracy 22%

**Decision:** Increase number of LSTM units

**Experiment 44**

**Action:** Added more LSTM units

**Result:** Accuracy 22%

**Decision:** Try GRU with ResNet50

**Experiment 45**

**Action:** Trained ResNet50 + GRU model

**Result:** Accuracy 18%

**Decision:** This doesn’t make sense. Try more images in training.

**Experiments 46-50**

**Action:** Chose 15 images in sequence for training again, but this time picked more frames from beginning & end than middle because middle frames don’t seem to add value. Trained 5 most promising models.

**Result:** Accuracies ranged from 16% to 38%. ConvLSTM2D performed best.

**Decision:** Try using all frames in the sequence.

**Experiment 51**

**Action:** Chose all 30 frames & trained ConvLSTM2D model.

**Result:** Accuracy 20%.

**Decision:** There’s no benefit. Bring back the old sequence & debug generator again.

**Experiments 52-61**

**Action:** Removed cropping from 1:1 images because they can be resized perfectly. Trained 10 models.

**Result:** Accuracies range from 13% to 47%. Best model is ResNet50 (30 layers trainable) + LSTM

**Decision:** The problem seems to be with data quality. Remove augmentation & try it later after visualizing it well.

**Experiment 62**

**Action:** Removed data augmentation logic from generator. Trained best performing 3D CNN model.

**Result:** Accuracy 69% but it’s overfitting with training accuracy 84.62%

**Decision:** Add Batch Normalization to fix overfitting

**Experiment 63**

**Action:** Added Batch Normalization to the previous model.

**Result:** Accuracy 9%

**Decision:** Remove Batch Normalization & add Dropout layers.

**Experiment 64**

**Action:** Removed Batch Normalization & added Dropout layers

**Result:** Accuracy 22%

**Decision:** Try other models

**Experiment 65-82**

**Action:** Trained 18 models.

**Result:** Accuracies ranged from 18% to 61%. Best performing model is ResNet50 (30 layers trainable) + 128 units of LSTM

**Decision:** It seems data quality problem is still there. Debug cropping logic again.

**Experiment 83**

**Action:** Checked cropping logic & found it to be false. Fixed it & trained best performing 3D CNN model again.

**Result:** Accuracy 83% (No overfitting)

**Decision:** Try Batch Normalization with this model to see if the model generalizes further.

**Experiment 84**

**Action:** Trained previous model with Batch Normalization

**Result:** Accuracy 74% with huge overfitting with training accuracy 99.45%

**Decision:** Add Dropout layers

**Experiment 85**

**Action:** Added Dropout layers to the previous models

**Result:** Accuracy 8%

**Decision:** Try other 3D CNN model with small number of kernels

**Experiment 86**

**Action:** Trained a simpler 3D CNN model

**Result:** Accuracy 19%

**Decision:** Try ConvLSTM2D model

**Experiment 87**

**Action:** Trained TimeDistributed + ConvLSTM2D model

**Result:** Accuracy 18%

**Decision:** Try more complex version of ConvLSTM2D model

**Experiment 88**

**Action:** Trained a more complex version of TimeDistributed + ConvLSTM2D model

**Result:** Accuracy 58%

**Decision:** Try vanilla TimeDistributed model

**Experiment 89**

**Action:** Trained vanilla TimeDistributed model

**Result:** Accuracy 24%

**Decision:** Try GRU models

**Experiments 90-92**

**Action:** Trained 3 GRU models of varying complexities

**Result:** Accuracies ranged from 13% to 24%

**Decision:** Try ResNet50 models

**Experiment 93**

**Action:** Trained ResNet50 (No trainable layers) + LSTM model

**Result:** Accuracy 23%

**Decision:** Make 10 layers of ResNet50 trainable

**Experiment 94**

**Action:** Made 10 layers of ResNet50 trainable in the previous model

**Result:** Accuracy 90%

**Decision:** Add more LSTM units

**Experiment 95**

**Action:** Added more LSTM units to previous models

**Result:** Accuracy 16%

**Decision:** Make 20 layers of ResNet50 trainable

**Experiment 96**

**Action:** Made 20 layers of ResNet50 trainable in the previous model

**Result:** Accuracy 73% but high overfitting with training accuracy 98.58%

**Decision:** Add Dropout layers to fix overfitting

**Experiment 97**

**Action:** Added Dropout layers

**Result:** Accuracy 68% but still overfitting with training accuracy 95.44%

**Decision:** Increase Dropout value

**Experiment 98**

**Action:** Increased Dropout value

**Result:** Accuracy 59% but still overfitting with training accuracy 91.97%

**Decision:** Reduce number of LSTM units

**Experiment 99**

**Action:** Reduced number of LSTM units

**Result:** Accuracy 62% (no overfitting)

**Decision:** Try other combinations of ResNet50 + LSTM

**Experiments 100-112**

**Action:** Trained 13 combinations of ResNet50 + LSTM models

**Result:** Accuracies ranged from 15% to 62%. Best performing model used very high number of LSTM units with 30 trainable layers of ResNet50

**Decision:** Try ResNet50 with GRU

**Experiments 113-114**

**Action:** Trained two GRU models with 30 trainable layers of ResNet50

**Result:** Accuracies were 52% for less GRU units & 59% for more GRU units

**Decision:** Try augmentation again

**Experiments 115-120**

**Action:** Added augmentation after visualization. Trained 6 best performing models again.

**Result:** Accuracies ranged from 22% to 60%. Best performing model was 128 units of GRU + ResNet50 (30 layers trainable)

**Decision:** Choose the final model

**Final Model**

**Type:** 3D CNN

**Accuracy:** 83%

**No. of Trainable Parameters:** 5,534,501

**Data Used:** No Augmentation