Gesture Recognition Case study

Below is the list of experiments conducted with 3 kinds of models and their outcomes. We have taken steps to enhance the model at each step. The decisions are based on some additional intermediatory steps that have not been listed in the document.

	Model	Configuration	Number of Parameters	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Observations	Decision
1.	CNN+LSTM	Epochs = 10, Feature map = [8, 16, 32, 64, 128] Dense = [1000,500,5] Batch size = 10 Epochs = 10 Learning rate patience = 5	3,974,609	1.1772	58.21%	1.6759	38%	The model starts overfitting after a few models	We will increase the data through augmentation.
2.	CNN+LSTM	Epochs = 10, Feature map = [8, 16, 32, 64, 128] Dense = [1000,500,5] Batch size = 10 Epochs = 10 Learning rate patience = 5	3,974,609	1.6063	27.54%	1.6405	19.17%	We see the accuracy is poor and loss is high.	We add batch normalization Layers after convolutions.
3.	CNN+LSTM	Epochs = 10, Feature map = [8, 16, 32, 64, 128] Dense = [1000,500,5] Batch size = 10 Epochs = 20 Learning rate patience = 5 Add Batch Normalizations	3,974,705	1.6027	26.57%	1.4743	37.50%	Slight improvement but the model is still not performing well.	We will reduce the dense layer neurons to original 256,128
4.	CNN+LSTM	Epochs = 10, Feature map = [8, 16, 32, 64, 128] Dense = [256,128,5] Layer = 4 Batch size = 10 Epochs = 10 Learning rate patience = 5	982,613	1.2777	45.41%	1.1535	57%	Model Stopped Learning after a few iterations. There was no improvement in loss.	We will try to increase the learning rate. In fact reduce the learning patience.

5.	CNN+LSTM	Epochs = 10, Feature map = [8, 16, 32, 64, 128] Dense = [256,128,5] Batch size = 10 Epochs = 10 Learning rate patience = 2	982,709	0.7258	70.98%	0.8363	67%	This is a significant improvement in model performance.	We will now experiment with the GRU model.
CNN+GRU									
6.	CNN+GRU	Epochs = 10 Feature map = [8, 16, 32, 64, 128, 256] Dense = [1000,500,5] Layer = 5 Batch size = 10 Epochs = 10 Optimizer Adam	1,943,345	1.2311	47.83%	0.9083	62.50%	Training and validation losses are high, but the model performs a decent job.	Maybe the dropouts are cancelling more layers than required. We try removing dropout layers
7.	CNN+GRU	Epochs = 10 Feature map = [8, 16, 32, 64, 128, 256] Dense = [1000,500,5] Layer = 5 Batch size = 10 Epochs = 10 Optimizer Adam	1,943,345	0.5730	81.59%	0.8050	69.67%	Model performance has improved significantly on training data after removing dropout layers, but validation accuracy has only improved slightly	We can try decreasing image size to 84X84, increasing epochs
8.	CNN+GRU	Epochs = 10 Feature map = [8, 16, 32, 64, 128, 256] Dense = [1000,500,5] Layer = 5 Batch size = 10 Epochs = 20 Optimizer Adam	1,303,345	0.5008	83.58%	0.6299	74.33%	Model performance has increased significantly after reducing the image size.	We try adding one additional dense layer for better classification
9.	CNN+GRU	Epochs = 10 Feature map = [8, 16, 32, 64, 128, 256] Dense = [1000, 500, 250, 5] Layer = 5	1,303,345	0.9942	66.5%	0.9089	67.33%	The model performance has reduced significantly, and the losses have increased	We try reducing the learning patience

		Batch size = 10 Epochs = 20 Optimizer Adam							
10.	CNN+GRU	Epochs = 10 Feature map = [8, 16, 32, 64, 128, 256] Dense = [1000, 500, 5] Layer = 5 Batch size = 10 Epochs = 20 Optimizer Adam	1,972,595	0.6762	76.95%	0.8439	66.5%	The performance has still not improved after reducing the learning patience.	We can try building model using Conv3D
CONV 3D									
11.	Conv3D	Feature map = [8,16,32,64] Dense = [256, 128, 5] Kernel = (3, 3, 3) Layers = 4 Batch size = 10 Epochs = 10 Optimizer - SGD	863,989	1.1530	54.02%	0.9234	56%	The model performs with low accuracy.	We try increasing number of epochs and changing optimizer
12.	Conv3D	Feature map = [8, 16, 32, 64] Dense = [256, 128, 5] Kernel = (3, 3, 3) Layers = 4 Batch size = 10 Epochs = 20 Optimizer - Adam	863,989	0.6139	77.11%	0.6639	73%	The model performance has improved significantly. The loss has reduced, and accuracy has increased.	We try adding more perceptron in the dense layer
13.	Conv3D	Feature map = [8, 16, 32, 64] Dense = [1000, 500, 5] Kernel = (3, 3, 3) Layers = 4 Batch size = 10 Epochs = 20 Optimizer - Adam	3,667,381	0.3780	85.07%	0.5135	81%	The model performance has again improved significantly. Training loss is much less, and accuracy has improved quite significantly	We try adding one more layer with dropout
14.	Conv3D	Feature map = [8, 16, 32, 64, 128] Dense = [1000, 500, 5]	6,877,237	0.4495	82.09%	0.4586	80%	The model has performed somewhat similarly to the	We try adding one more layer with Batch normalization

		Kernel = (3, 3, 3) Layers = 5 Batch size = 10 Epochs = 20 Optimizer - Adam						previous model. There is no significant loss of gain in accuracy.	
15.	Conv3D	Feature = [8, 16, 32, 64, 128, 256] Dense = [1000, 500, 5] Kernel = (3, 3, 3) Layers = 6 Batch size = 10 Epochs = 20 Optimizer - Adam	13,444,533	0.4912	79.60%	0.5731	79%	The model has performed poorly as compared to previous 2 models. The loss has increased, and accuracy has dropped.	We will try to reduce the filter size for looking at the efficiency.
16.	Conv3D	Feature = [8, 16, 32, 64, 128, 256] Dense = [1000, 500, 5] Kernel = (3, 3, 3) Layers = 6 Batch size = 10 Epochs = 20 Optimizer - Adam	13,441,645	1.0849	57.49%	0.9909	55%	The accuracy decreased significantly.	We try training a new model using transfer learning
17.	MobileNet	Dense = [128, 5] Batch size = 10 Epochs = 20 Optimizer – Adam Trainable = False	609,541	1.0205	60.70%	1.7563	23%	Model is overfitting. It has a significant difference between training and validation accuracy.	We are making the layers as trainable
18.	MobileNet	Dense = [128, 5] Batch size = 5 Epochs = 20 Frames = 30 Optimizer – Adam Trainable = True	3,816,517	-	-	-	-	The model gave resources exceeded error	We reduce the number of frames
19.	MobileNet	Dense = [128, 5] Batch size = 5 Epochs = 20 **Frames = 16** Optimizer – Adam Trainable = True	3,816,517	47.44%	86.13%	0.3834	87.67%	This model gave the best performance so far.	This is the final submission.