Gesture Recognition Project Report

Kaushik Deb (Tata Insights and Quant) Hrudaya Ranjan Sahoo (Huwei)
Hrishikesh Desai (Druva) Vijayaraghavan Amirisetty (Amazon)

BUSINESS PROBLEM 1

INPUT DATA 1

SUMMARY 2

DETAILS OF OUR APPROACH 3

Model Experiments **3**
3D CNN 3
2D CNN + LSTM 3
Transfer Learning + LSTM 3
Preprocessing 4
Tuning Image Size 4

Business Problem

We need to build a gesture recognition system for a Smart Television Company. There are five hand gestures which need to be identified. The hand gestures are provided as a video which comes from the smart TV camera. Likely due to the different models of the television, we have two different types of videos. Using these videos as an input to the model we should correctly identify the gesture that the customer has performed and take the appropriate action in the Smart TV. The accuracy of the model has to be high to have the correct commands and avoid customer frustration.

Input Data

We have two types of videos 360 x 360 pixels and a smaller one 120 x 160. Each video is provided as a sequence of 30 images. The input data has already been split into training and validation data.

Summary

We achieved a validation accuracy of 93% and a train accuracy of 97% using transfer learning using Resnet50 + LSTM. For training the model we used a Stochastic Gradient Descent Optimized with a learning rate of 0.1 and a Learning Rate Decay of 0.5. To reduce the variance while maintaining the model accuracy we used Dropout Regularization with a Dropout % of 25%. We also used an image size of 90x90 pixels and the inbuilt resnet preprocessing. Due to the hardware constraints (memory) we could not go beyond 90x90 pixels and Resnet with more layers.
In all we ran 23 experiments to conclude the optimal model. These experiments are summarized in the table below.

Experiment Number	Batch Size	Image Size	Layers	Learning Rate	Image Count Considered	Learing Rate Decay	Validation Accuracy After 5 Epochs	Validation Accuracy After 10 Epochs	Training Accuracy	Filter	Regularization Drop Out percentage	Alogithm	Image Pre Processing Algorithm
	3D CNN												
1	50	100*100	32 - 64 -128-512(Dense)	0.01	15	0.7	0.21	0.21	0.2	2*2*2	0.25-0.5	CONV3D	Resize+Normalization+Scaling
2	50	40*40	32-64	0.01	15	0.5	0.4	0.68	0.6	2*2*2	0.25-0.5	CONV3D	Resize+Normalization+Scaling
3	50	40*40	32-64-64(Dense)	0	15	0.6	0.3	0.3	0.3	2*2*2	0.25-0.5	CONV3D	Resize+Normalization+Scaling
4	50	50*50	32-64(Dense)	0.01	15	0.5	0.58	0.58	0.5	2*2*2	0.25-0.5	CONV3D	Resize+Normalization+Scaling
5	50	70*70	32-64-64(Dense)	0.01	15	0.5	0.37	0.37	0.3	2*2*2	0.25-0.5	CONV3D	Resize+Normalization+Scaling
6	50	100*100	32-64(Dense)	0.01	15	0.5	0.43	0.45	0.4	2*2*2	0.25-0.5	CONV3D	Resize+Normalization+Scaling
7	20	100*100	16-32(Dense)	0.01	15	0.7	0.35	0.35	0.3	2*2*2	0.25-0.5	CONV3D	Resize+Normalization+Scaling
8	50	40*40	64-64-64(Dense)	0.01	15	0.5	0.53	0.53	0.4	3*3*3	0.25-0.5	CONV3D	Resize+Normalization+Scaling
9	50	40*40	64-64-64(Dense)	0.1	15	0.6	0.4	0.4	0.4	3*3*3	0.3-0.4	CONV3D	Resize+Normalization+Scaling
	CONV2D + LSTM/GRU												
10	50	40*40	32-64-256(LSTM)	0.1	15	0.5	0.48	0.53	0.43	3*3*3	0.25-.25-.	Conv2D + LSTM	Resize+Normalization+Scaling
11	50	40*40	32-64-256(LSTM)	0.1	15	0.5	0.5	0.63	0.7	3*3*3	0.25-.25-.	Conv2D + LSTM	Resize+Normalization+Scaling
12	50	40*40	32-64-256(LSTM)	0.1	15	0.5	0.53	0.63	0.7	3*3*3	0.15-.15-.	Conv2D + GRU	Resize+Normalization+Scaling
	Transfer Learning + LSTM												
13	20	40*40	Resnet50 + LSTM	0.1	15	0.5	0.87	0.87	0.96		0-.25	Resnet50 + LSTM	Resize+Normalization+Scaling
14	20	40*40	Resnet50 + LSTM	0.1	15	0.5	0.85	0.85	0.95		0-.15	Resnet50 + LSTM	Resize+Normalization+Scaling
15	50	40*40	Resnet50 + LSTM	0.1	30	0.5	0.7	0.7	0.9		0.25-0.25	Resnet50 + LSTM	Resize+Normalization+Scaling
16	20	40*40	Resnet50 + GRU	0.1	15	0.5	0.7	0.71	0.9		0-.50	Resnet50 + GRU	Resize+Normalization+Scaling
17	20	40*40	Resnet50 + GRU	0.1	15	0.5	0.79	0.82	0.96		0-.25	Resnet50 + GRU	Resize+Normalization+Scaling
18	20	40*40	VGG16 + LSTM	0.1	15	0.5	0.2	0.1	0.8		0-.25	VGGNet + LSTM	Resize+Normalization+Scaling
	Transfer Learning + LSTM + Resnet Preprocessing												
19	20	40*40	Resnet50 + LSTM	0.1	15	0.5	0.67	0.8	0.95		0-.25	Resnet50 + LSTM	PreProcess_V1
20	20	50*50	Resnet50 + LSTM	0.1	15	0.5	0.8	0.85	0.96		0-.25	Resnet50 + LSTM	PreProcess_V2
21	20	40*40	Resnet50 + LSTM	0.1	15	0.5	0.8	0.82	0.96		0-.25	Resnet50 + LSTM	PreProcess_V2
22	20	80*80	Resnet50 + LSTM	0.1	15	0.5	0.84	0.88	0.97		0-.25	Resnet50 + LSTM	PreProcess_V2
23	20	90*90	Resnet50 + LSTM	0.1	15	0.5	0.9	0.93	0.97		0-.25	Resnet50 + LSTM	PreProcess_V2

Details of our Approach

We experimented with multiple approaches to come up with a model that achieved an accuracy of 93% on the validation dataset. The different models that we tried were 3D CNN, 2D CNN + LSTM, Transfer Learning + LSTM with our custom pre-processing and Transfer Learning + LSTM with ResNet Preprocessing. For Transfer learning, we experimented with VGG16 and Resnet 50.

We also experimented with multiple model hyper-parameters like the Learning Rate, Drop out Regularization %, Learning Rate, Learning Rate Decay, Number of Layers and Filter Size (Number of trainable parameters). We also experimented with the image size and image preprocessing. We had to run these experiments under the hardware constraints of GPU time, training time and GPU memory.

Model Experiments

3D CNN

We started with a 3D Convolutional Neural Network considering the simplicity and its applications for video streams. We built a 3D CNN with multiple layers and multiple image sizes. We observed that adding more layers not improving the accuracy significantly but impacting the execution time. A 3D CNN with three layers with the filter sizes being 32-64-128 (Experiment 1) was not giving better accuracy than a model with lesser parameters 32-64 (Experiment 2). Where the model with 32-64 training ran is faster (approximately 40 times reduction in the training time per epoch – from 5 minutes per epoch to 55 seconds per epoch). Counterintuitively, we got a highest validation accuracy of 68% with two layers (Experiment Number 2). Since an accuracy of 68% is not good enough for a customer facing application, we did not select this model and moved on to a 2D CNN + LSTM Model.

2D CNN + LSTM

From the above experiment we found out that the optimal learning rate is 0.1 and the optimal Learning Rate Decay is 0.5 for model convergence in fewer epoch and minimal amount of time. We experimented with dropout regularization % and achieved better accuracy with 25% dropout over a 50% dropout. With these hyperparameters we experimented with three (Experiment 10, 12 and 12) 2D CNN + LSTM models and achieved a validation accuracy of 63%. This was lesser than the 3D CNN and hence we moved on to the Transfer Learning + LSTM model.

The rationale behind this is that neural networks require a large amount of training data. However, we had only 620 videos (datapoints) for the training. This datasize is not optimal for training a neural network. Since Resnet50 and VGG have been trained with over 14 million images and have over 50 layers using transfer learning was the optimal choice considering these pretrained models on large datasets.

Transfer Learning + LSTM

From the above two experiments we concluded on the optimal hyperparameters of Learning Rate = 0.1, Learning Rate Decay of 0.5 and Drop out % of 25%. We then experimented with VGG and Resnet50 as the two options for Transfer Learning. We tried using VGG first, however we got a validation accuracy of only 10% (Experiment 18). For Resnet we tried freezing the first 10, 70 and 140 layers however we got optimal performance by training all the layers (Experiment 13). We also experimented with Resnet 152, however we got some errors running it on the GPU. We also tried to add a couple of more layers to Resnet50, however it did not improve the model performance. We also experimented with dropout % here considering a high variance in the model. We observed that 50% dropout in dense layer sometime deteriorated the accuracy and the ideal dropout found from 15 to 25%. We observed that Resnet50 + LSTM performed better (87% validation accuracy) than Resnet 50 + GRU.

Hence, we zeroed in on Resnet50 + LSTM and a candidate for further improvement in accuracy and did the following experiments.

Preprocessing

We experimented with two preprocessing methods. We built our own custom preprocessing method and we also used the inbuilt resnet pre-processing function provided by keras.

Custom Preprocess Method where we do the following:

1. Crop the 120x160 image to 120x120,
2. Resize the image to the target size,
3. Subtract the mean of the imagenet mean.

The keras pre-processing function performed better in terms of the model accuracy.

Tuning Image Size

We also had additional constraints on the GPU memory and time taken to train the model. In order to speed up the model training, we initially did our experiments with 40x40 pixels image and once we identified an optimal model, we trained the optimal model with an image size of 90x90 pixels. Training the model with a 90x90 pixels gave an increase of 13% improvement from 80% to 93%. We anticipate that an image of 120x120 would give a better accuracy, however the Nimble Box GPU gave an out of memory error when we tried to set the image size to 120x120 pixels. The graph of Validation accuracy vs Image Size is provided in Figure 1.

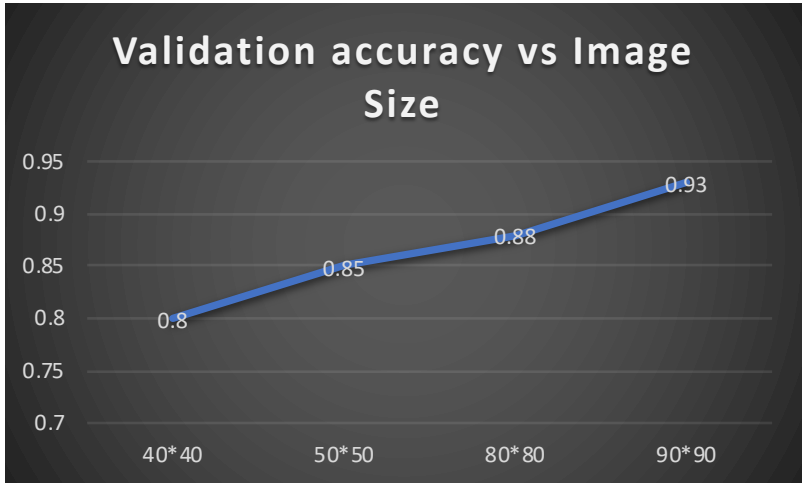

Figure 1. Validation Accuracy (Y axis) vs Image Size in pixels (X Axis)