Neural Network Project - Gesture Recognition

Problem Statement:

Develop an algorithm for a smart TV that can recognise five different gestures performed by the user which will help control the TV without using a remote. The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command.

Understanding the Dataset

The training data is categorized into one five classes. Each video (typically 2-3 seconds long) is divided into a sequence of 30 frames (images). These videos have been recorded by various people performing one of the five gestures in front of a webcam - similar to what the smart TV will use.

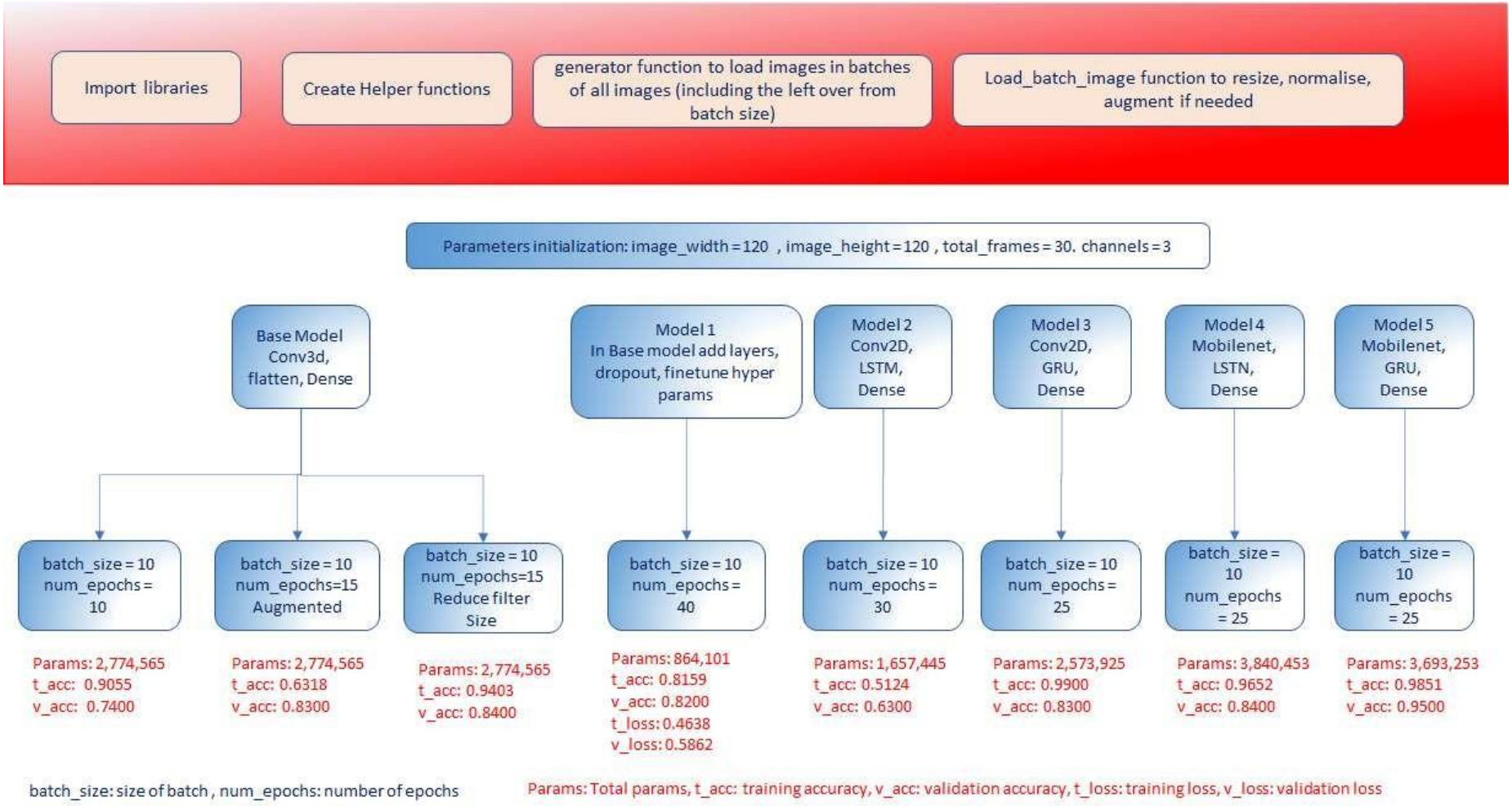
Objective

For analysing videos using neural networks, two types of architectures are used commonly. One is the standard CNN + RNN architecture in which the images of a video are passed through a CNN which extracts a feature vector for each image, and then pass the sequence of these feature vectors through an RNN. The objective of this assignment is to train different models on the 'train' folder to predict the action performed in each sequence or video and which performs well on the 'val' folder as well. The final test folder for evaluation is withheld - final model's performance will be tested on the 'test' set.

Approach:

As part of this case study, multiple options were attempted, and the steps taken for case study has been captured in the code walkthrough flow diagram

Code Walkthrough:



Experiments carried out:

- As seen in the above diagram, eight models were tried with different params and architecture:
- Image width , height, total number of frames and channels were set to be 120,120,30, 3 respectively.
- Augmentation was carried out as part of calling load_batch_image function and was carried out as an option based on user input.
- Augmentation performed on base model and was found to be not so effective and hence not used in other trials.
- Architecture used in each models captured in the above diagram.
- In all the models , adam optimiser was used.
- As it is multi target classification , softmax was used in the last dense function.

Testing the GPU utility:

The batch size was increased incrementally to test for GPU utility. We had hit the limit of memory resources when we increased the batch_size to 512. We got the below error.

- ResourceExhaustedError: OOM when allocating tensor with shape[512,30,30,120,120] and type float on /job:localhost/replica:0/task:0/device:GPU:0 by allocator GPU_0_bfc

Results and interpretation:

S No	Model	Architecture & Layers	batch Size	# epochs	Augmentation	Total Parameters	Training Accuracy	Validation Accuracy	% diff in accuracy	Observation
1	Base Model	Conv3d, flatten, Dense	10	10	No	27,74,565	0.906	0.74	18%	Good accuracy but Overfitting
2	Base Model	Conv3d, flatten, Dense	10	15	Yes	27,74,565	0.632	0.83	-31%	Augmentation is not improving accuracy
3	Base Model	Conv3d, flatten, Dense	10	15	No	27,74,565	0.94	0.84	11%	Reduced filter increases the dense and improves overfitting issue
4	Model 1	in Base model add layers, dropout, finetune hyper params	10	40	No	8,64,101	0.816	0.82	-1%	adding layers in base model, dropout and hyper param tuning produced a model with decent accuracy with no evidence of over fitting. Having the least params.
5	Model 2	Conv2D,LSTM,Dense	10	30	No	16,57,445	0.512	0.63	-23%	Accuracy dropped from previous experiment
6	Model 3	Conv2D,GRU,Dense	10	25	No	25,73,925	0.99	0.83	16%	Accuracy increased significantly and but the model is over fitting
7	Model 4	Mobilenet,LSTN,Dense	10	25	No	38,40,453	0.965	0.84	13%	Total number of parameters increased , Accuracy is good and the model is over fitting
8	Model 5	Mobilenet,GRU,Dense	10	25	No	36,93,253	0.985	0.95	4%	Total number of parameters increased , Accuracy is very good and the model is not over fitting

Conclusion:

As per the above table, Sl no 4 and 8 were found to be having no Over fitting issue. However, the SlNo 4, Model 1 was chosen as the number of parameters is 75% less than the Sl No 8. The details of the Model 1 were mentioned below

1. Four hidden layers with activation function as relu with filter size as 0,1,2,3 respectively and the last layer has 25% dropout, The first 2 layers had kernel size as (3,3,3) and the last 2 layers had (1.3,3). First 3 layers used batch Normalisation and the last one without batch normalisation. Padding with 'same' attribute was added to all the 4 layers.
2. flatten layer with 2 drop out of 50%
3. SoftMax dense function is used to classify.

- The loss and accuracy graph for the model is as mentioned below:
- The loss is getting reduced, and the accuracy increased in each epochs as seen below: