Gesture Recognition Project

16 November 2022

Students:

- Phạm Thái <phamthai.ats@gmail.com>
- Gert Agenbag <gert.agenbag@gmail.com>

Generator

A generator was configured to feed training and validation data into the model during training.

Using all the images in the sequences gave the best results during. However, this comes at the cost of larger memory requirements and smaller batch sizes during training. One could skip over every second image in the sequence to reduce the memory footprint by setting:

```
img_idx = [*range(1, 30)]
```

The image dimensions used were 120x120 pixels. Images that are 160x120 were cropped to 120x120, and images that were 360x360 were resized using a Lanczos filter. Resizing images by an integer factor minimizes the introduction of resizing artifacts.

The generator was also used to rescale the image pixel value range from 0..255 to 0..1 or -1..1, or so on, depending on which model was used.
Detail: *(Key = normalize_type* or *scale_zero_centered)*

Id 0: Devide pixel values / 255.
Id 1: Scale zero center (img-127)/127.
Id 2: Scale with distribution percentile
Id 3: Feature out of pretrain VGG16

Model Building

Several approaches were tried and a number of models were built during this project. We considered the tradeoff between model size and model performance, and therefore present two models as part of this submission. The small model excels in memory footprint, while the large model excels in performance.

Result of Models

We achieved acceptable results with the small CNN + LSTM model, but the model did exhibit signs of overfitting (1st), Underfitting (2nd + 3rd)
File Ipynb: Gesture_Recognition_CNN_LSTM.ipynb
Model: cnn_lstm/model-00014-0.26809-0.91101-0.64664-0.76000.h5 (1st Model)

We achieved acceptable results with the small CNN + GRU model, but the model did exhibit signs of overfitting (1st + 3rd), Underfitting (2nd)
File Ipynb: Gesture_Recognition_CNN_GRU.ipynb
Model: cnn_gru/ model-00017-0.26209-0.91855-0.71141-0.76000.h5 (3rd Model)

We achieved acceptable results with the small CNN + Conv3D model, but the model has not reached convergence (2nd), Underfitting (1st + 3rd)
File Ipynb: Gesture_Recognition_CNN3D.ipynb
Model: cnn3d/ model-00017-0.28145-0.88839-0.68774-0.76000.h5 (2rd Model)

We achieved acceptable results with the Pretrain VGG16 + GRU model, but the model has good convergence sign, high Accuracy on train set and val set (1st)
File Ipynb: Gesture_Recognition_VGG16.ipynb
Model: *pretrained_vgg16_gru/ model-00046-0.03527-0.99397-0.15629-0.96000.h5 (1st Model)*

Decision Log

The table below summarizes our findings and observations during model building.

The table is not exhaustive, but it highlights the most important learnings and decision points. We've sometimes trained numerous variants of a model while only making minor adjustments. Some adjustments that had relatively little impact, and therefore were not captured here.

Models	Times	EXPERIMENT	RESULT	DECISION + EXPLANATION	PARAMETERS
CNN+LSTM	1st	• Tartget_size = (128,128,3) • CNN2D(8,16,16,16)+LSTM(128)+Dropout(0.5 • Opt=Adam(1e-3) • *normalize_type* =1 • 40 epochs + Reduce Lr	Acc_train : 0.91 Acc_val : 0.76 (Epochs 14)	*Overfitting* *Try with Adadelta (1.)* *Optimization + Image normalize method*	105,493
	2nd	• Tartget_size = (128,128,3) • CNN2D(8,32,32,32)+LSTM(128)+Dropout(0.5) • Adadelta(1.) • *normalize_type* =2 • 40epochs + Reduce Lr	Acc_train :0.19 Acc_val : 0.24 (Epochs 10)	*Underfitting* *Try with Adadelta (1e-2)* *Optimization+ Image normalize method*	124,997
	3rd	• Tartget_size = (128,128,3) • CNN2D(8,32,32,32) • LSTM(128) • Dropout(0.3)	Acc_train :0.472 Acc_val : 0.51 (Epochs 20)	Still Underfitting Adadelta not Good Ideas to Optimzation Find another Image normalization method like mean values	125,125

		• BatchNormalization • Kernal=(5x5) • Opt=Adadelta(1e-2) • *normalize_type*=0 • 40 epochs+Reduce Lr			
CNN3D	1st	• Target_size = (128,128,3) • Conv3D(8,16,32) • BatchNormalization() • Dense (256, 128) • Dropout (0.5, 0.5) • Adam(1e-3) • *normalize_type*=1 • 20 Epochs + Reduce Lr • Batch_size = 16	Acc_train :0.309 Acc_val : 0.42 (Epochs 14)	Underfitting Change the image normalization method to Improve accuracy. Make deeper depth of Conv3D	6,343,461
	2nd	• Target_size = (128,128,3) • Conv3D(8,8,32,32) • BatchNormalization() • Dense (128) • Dropout (0.5) • Adam(1e-3) • *normalize_type*=2 • 20 Epochs + Reduce Lr • Batch_size = 16	Acc_train :0.888 Acc_val : 0.76 (Epochs 17)	Model has not converged, Cannot confirm overfitting + underfitting. It is possible to increase the epochs coefficient	300,253
	3rd	• Target_size = (128,128,3) • Conv3D(16,16,32) • BatchNormalization() • Dense (128, 128) • Dropout (0.5, 0.5) • Adam(1e-3) • *normalize_type*=2 • 20 Epochs + Reduce Lr • Batch_size = 16	Acc_train :0.23 Acc_val : 0.34 (Epochs 11)	Model Underfitting Model has high Parameters number but the result is not good. Reduced depth, increased hidden notes from the 2nd model does not bring good quality.	3,185,365
CNN + GRU	1st	• Tartget_size = (128,128,3) • CNN2D(8,16,16,16) • GRU(128)+Dropout(0.5 • Opt=Adam(1e-3) • *normalize_type* =1 • 30 epochs + Reduce Lr • Batch_size=16	Acc_train : 0.895 Acc_val : 0.72 (Epochs 26)	Overfitting	81,173
	2nd	• Tartget_size = (128,128,3) • CNN2D(8,32,32,32) • GRU(128)+Dropout(0.5) • Adam(1e-3) • *normalize_type* =3 • 30 epochs + Reduce Lr • Batch_size=32	Acc_train : 0.253 Acc_val : 0.38 (Epochs 27)	Underfitting Modify kernel matric, Reduce Dropout=0.3, Adadelta(1e-2)	104,773

	3rd	• Tartget_size = (128,128,3) • CNN2D(8,32,32,32) • GRU(128)+Dropout(0.3) • BatchNormalization • Kernal=(5x5) • Opt=Adadelta(1e-2) • *normalize_type*=0 • 30 epochs+Reduce Lr • Batch_size=16	Acc_train : 0.918 Acc_val : 0.76 (Epochs 17)	Model from Underfitting to Overfitting	104,901
Pretrain VGG16	1st	• Target_size = (120,120,3) • Pretrain_Vgg16 + TimeDistributed • GRU(256)+Dense(32)+Dropout(0.25, 0.25) • Batch-size=6, Epochs=50 • *scale_zero_centered = 1* • Adam(1e-3)+Reduce Lr	Acc_train : 0.97 Acc_val : 0.93 (Epochs 46)	Convergence Sign. Values are excellent. Using pre-train gives better model quality.	3,746,053

Demo and Testing

- We have created a notebook file to load models and predict actions via : load_test_vgg16.ipynb
- We use the best model is the Pretrain-VGG16. See load_test_vgg16.ipynb for more details.