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 multiple models as part of this submission. The CNN + GRU models excels in memory footprint, while the pre-trained VGG16 + GRU model excels in accuracy.

### **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 Pre-trained VGG16 + GRU model

**File Ipynb:** Gesture\_Recognition\_VGG16.ipynb

**Model:**  pretrained\_vgg16\_gru/ model-00042-0.04627-0.99397-0.06957-0.99000.h5 (Final 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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MODEL** | **ATTEMPT** | **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) * BatchNormalization * Kernal=(5x5) * Opt=Adadelta(1e-2) * *normalize\_type*=0 * 40 epochs+Reduce Lr | 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 |
| **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 |
| **Pre-trained**  **InceptionResNetV2**  **+ LSTM** | 1st | Used InceptionResNetV2 with pre-trained ImageNet weights.  Removed the last layer of InceptionResNetV2 was flattened before feeding it into a LSTM. |  | Model failed to converge during training. |  |
| **Pre-trained**  **EffiecientNetV2 + LSTM** | 1st | Used EfficientNetV2B2 with pre-trained ImageNet weights.   Removed the last layer of EfficientNetV2B2 was flattened before feeding it into a LSTM. |  | Model failed to converge during training. |  |
| **Pre-trained**  **VGG16 + LSTM** | 1st | Used VGG16 with pre-trained ImageNet weights.  The last layer of VGG16 was removed and its output was flattened before feeding it into a LSTM.  Used 128 units in the GRU followed by a 64-unit dense layer and single dropout layer.   * Target\_size = (120,120,3) * Batch-size=6, Epochs=50 * *scale\_zero\_centered = 1* * Adam(1e-3)+Reduce Lr |  | Promising results but there was some overfitting present. |  |
| 2nd | Added batch normalization layers and another dropout layer. |  | Model convergence was significantly slower during training. |  |
| 3rd | Used 32 units in the dense layer. |  | Model convergence was very slow during training. |  |
| 4th | Used a single dropout layer but increased the dropout percentage. |  | Model convergence was very slow during training. |  |
| 5th | Removed the batch normalization layers. | Best Model (after 19 epochs of training):   * Training loss: 0.0238 * Training accuracy: 0.9955 * Validation loss: 0.6031 * Validation accuracy: 0.84 | Model convergence was good, but overfitting was present. | Total: 17,144,325  Trainable: 2,429,637  Non-trainable: 14,714,688 |
| **Pre-trained VGG16**  **+ GRU** | 1st | Used VGG16 with pre-trained ImageNet weights, but replaced the LSTM with a GRU.  The last layer of VGG16 was removed and its output was flattened before feeding it into a GRU.  Used 128 units in the GRU followed by a 32-unit dense layer with single dropout layer.   * Target\_size = (120,120,3) * Batch-size=6 * scale\_zero\_centered = 1 * Adam(1e-3)+Reduce Lr | Training accuracy: > 0.85 | Very promising model. Training the GRU is significantly easier than the LSTM.  Model converges very quickly during training, and there are no signs of overfitting. |  |
| 2nd | Increased GRU size to 192 units. | Training accuracy: > 0.85 | Improved results. |  |
| 3rd | Increased GRU size to 256 units. | Training accuracy: > 0.85 | Improved results. |  |
| 4th | Increased GRU size to 384 units. |  | The validation results diverged from the training results.  Validation accuracy reduced and validation loss increased. This is indicative of overfitting. |  |
| 5th | Reverted back to 256 units in the GRU. Added batch normalization layers. | Validation accuracy: 0.92 | The batch normalization layers significantly improved the model stability between training epochs and improved the convergence. |  |
| 6th | Changed the dense layer size from 32 to 64 units. |  | Degraded results. |  |
| 7th | Reverted back to 256 units in the GRU followed by a 32-unit dense layer. Added another dropout layer. | Validation accuracy: 0.96 | Very good agreement between training and validation performance. No apparent signs of overfitting or underfitting. |  |
| 8th | Final Model.  Fine-tuned the dropout values. Trained for 50 epochs.   * Target\_size = (120,120,3) * Batch-size=6, Epochs=50 * scale\_zero\_centered = 1 * Adam(1e-3)+Reduce Lr | Best model (after 42 epochs of training):   * Training loss: 0.0463 * Training accuracy: 0.9940 * Validation loss: 0.0696 * Validation accuracy: 0.99 | Good results, but with a relatively large model.  The model is consistently able to achieve an accuracy score of around 0.97 on the validation dataset across different training epochs and when retraining with different batch sizes. The best run achieved 0.99 accuracy on the validation dataset. | Total: 18,461,317  Trainable: 3,746,053  Non-trainable: 14,715,264 |