Activity and Emotion Detection for “Smart Homes”

Inimfon Akpabio

April 15, 2021

Abstract

Advances in IoT (Internet of Things) and artificial intelligence have facilitated the deployment of smart home devices that enable hospitals and health workers monitor the state and well-being of patients, particularly elderly people, who often live alone. Such technologies can be crucial in tracking the health of a patient and ultimately, providing emergency care to such individuals if needed. In this project we propose a model to detect “Blowing nose” in a smart home setting. Our model starts with a DenseNetV2 network used for feature extraction and then feeds into Bidirectional LSTM + Fully-connected layers to perform the final prediction. The feature extractor uses pre-trained image-net weights. For each video, we sample only 1 out of every 3 frames. The rationale is that an activity like reading is a perpetual motion, hence, this should be sufficient. A mixture of videos from both the Kinetics700 and HMDB51 datasets were used to train the overall classifier. The model currently has an accuracy of 80.9% which rivals the state of the art.

1. Topic

The selected action for this project is “Blowing nose”.

2. Motivation

Human activity recognition aims to detect common or pre-defined human activities either from sub-sampled video clips or full-length videos. Typically, in the case of full-length videos, the goal is to temporally localize the action. This field of research is particularly attractive due to the potential applications in health care and personalized care for elderly citizens. With the population of elderly citizens projected to rise to 22% by 2050 [1], smart monitoring systems could help such individuals to lead healthy and yet independent lifestyles [2]. There have already been multiple attempts to monitor human activities using signals from wearable devices [3]. While these methods can be useful, they are inevitably limited due to the one-dimensional nature of this approach. Hence, it makes sense to utilize visual data as a means for detection. The benefit of the visual approach is that it affords more features for better detection and can be easily collected using video capture devices. A lot of research work has been done in the field Action/emotion detection in videos in recent years. Advances in computer-vision AI and the advent of competitions worldwide [4] have significantly increased interest in video based approaches for action detection. In this project we will build a model that relies solely on visual data to detect the action: “Blowing nose”.

3. Related Works

While this model design is not directly based on any particular paper, there are multiple research endeavors that have adopted similar designs to tackle video and speech related problems. For instance, [5] uses VGG-16 layers to extract features from a video clip and then feeds this data to an LSTM network to generate captions for the video.

4. Proposed model

Fig 1 shows the complete model design. The model starts with a DenseNet121 network [6] which has been pre-trained with ImageNet weights, hence, making it ideal for feature extraction. At the end of the DenseNet121 network, we perform a global maxpool which basically takes the maximum value from each filter. This helps to reduce the dimensionality of features and promotes spatial hierarchy. Next, using time-distribution, the features are fed into a Bidirectional LSTM network with 64 units. Finally the output of the BLSTM network is fed into a fully connected network that performs prediction. The fully connected network consists of two layers with 1024 and two units respectively. The second layer applies Softmax activation on the 2 units to perform binary-classification.

As a pre-processing step, video clip frames must be resized to (224, 224) and scaled to [0, 1]. The input to the model is an array of 20 image frames sub-sampled from a 2 second clip and has a dimension of (20, 224, 224, 3). It outputs an array of size 2 representing the prediction probabilities for “Reading” and “not Reading”.

5. Dataset

Feature Extractor:

The feature extractor was pre-trained on ImageNet [8]. Most of the proposed solutions to activity recognition often employ pre-trained models due to their robustness and superior ability to extract features.

Full model:

Finding a video dataset to train the full model proved difficult. The Kinetics700 dataset is a large datasets that contains a myriad of human-performed actions and has been widely explored for video recognition tasks in recent years [9]. Due to its massive size, it could take days to download the entire dataset even with a stable internet connection. Hence, it made sense to download only the “Reading” class and combine this with the HMDB51 dataset which similarly contains videos of human-performed actions [10]. For training, the videos were separated into two classes: “Blowing nose” and “miscellaneous” which represents the collection of all “not blowing nose” videos. Both classes currently have sizes of 388 and 382. As a preprocessing step, the videos were sample-centered to improve convergence during training. It is a common practice to centralize and normalize data before training neural networks. No data Augmentation was applied due to the relatively large size of the dataset.

6. Model Training and Performance

During training, the weights of the feature extractor are frozen up to the "conv5_block3_2_relu" layer. Subsequent layers are fine-tuned for better performance. Hyper-parameters like batch-size, epochs and learning rate were selected by testing pre-selected values in fixed ranges and choosing the values that yielded the best convergence. Fig. 2 shows a snippet of the learning rate optimization. We use the Adam optimizer with a learning rate of 0.0005 and about 14 epochs to train the model. Even before this many epochs, the model begins to exhibit heavy overfitting, hence, we employ early stopping to prevent further training. We also attempted to increase its generalization power by adding L2 regularization and dropout of 50% to the fully connected layers. On the validation set, the model achieves an accuracy of about 91% and an overall accuracy of 80.9% on YouTube videos. The breakdown is about 82.1% on positive (contains desired action) YouTube videos and 79.7% on negative YouTube videos.

The dip in performance on positive YouTube video clips is a bit surprising. However, it is worth mentioning that they are mostly concentrated in clips from the same video. Perhaps expanding the dataset to incorporate more varieties would prevent this.

In addition to the aforementioned method, we also experimented with a slight modification to our experiments. In this modification, we apply subtractive sampling for the video frames. This involves taking the difference between consecutive frames. Surprisingly, this approach yielded very high accuracy on the training and validation data (> 90% accuracy), both of which contained videos of length longer than 2 seconds. On the other hand, it yielded very poor performance accuracy on the 2 second YouTube test videos (~ 20% accuracy). This most certainly due to the brevity of the test videos. Hence, we observed that this method could pose great potential when applied on longer videos that contain more contextual information.

7. YouTube

On the curated YouTube test set, the performance on positive videos is 80.9%. This is probably due to the ability of DenseNet to extract salient features for classification. The FPR and FNR are both 20.33% and 17.9% respectively. Hence there is still room for improvement.

As of right now, the videos are manually segmented into 2 second clips. We first use youtube-dl to download the entire video and employ the MoviePy library in python to manually segment 2 second subclips from the original video. It should be noted that other common approaches to this have been known to be inefficient and heavy on computing resources. A possible improvement would be to develop models that are capable of temporal localization. Hence, if we feed in an entire video, we expect the model to output time segments that contain the action.

8. Improve Accuracy

A possible improvement might be to explore ensemble methods to boost prediction accuracy.

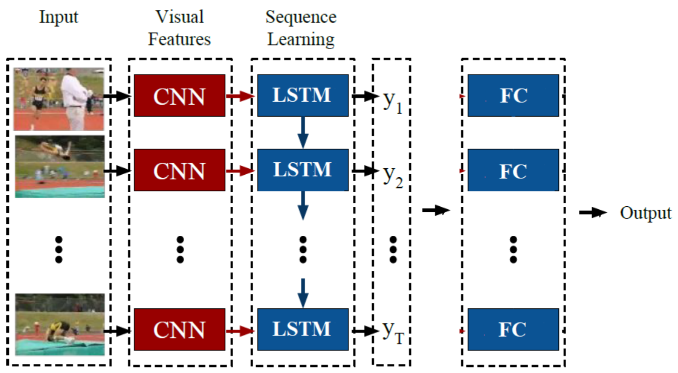
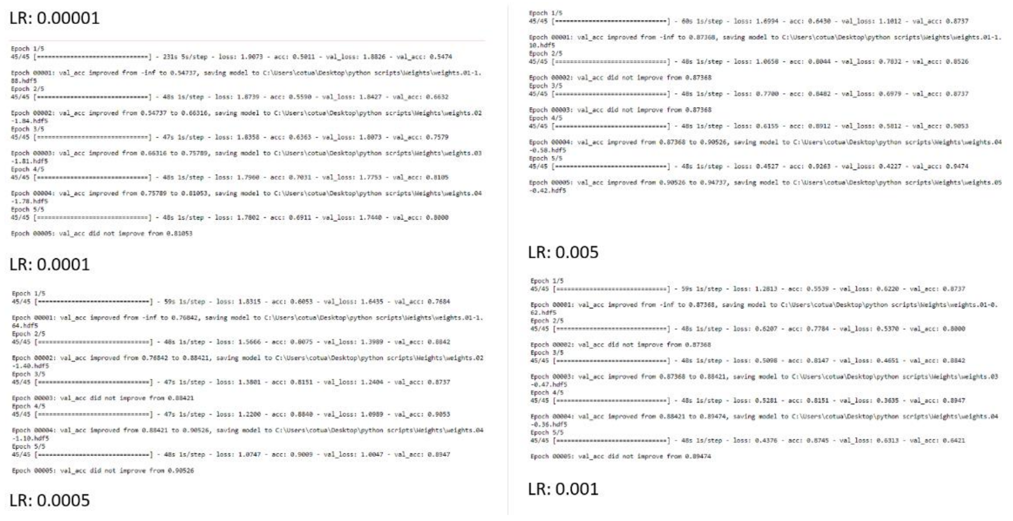


Fig 1: Model Diagram

Source: Adapted from [7]



LR: 0.00001

```
Epoch 1/5
45/45 [==============================] - 231s 5s/step - loss: 1.9073 - acc: 0.5011 - val_loss: 1.8826 - val_acc: 0.5474

Epoch 00001: val_acc improved from -inf to 0.54737, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.01-1.
88.hdf5
Epoch 2/5
45/45 [==============================] - 48s 1s/step - loss: 1.8739 - acc: 0.5590 - val_loss: 1.8427 - val_acc: 0.6632

Epoch 00002: val_acc improved from 0.54737 to 0.66316, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.02
-1.84.hdf5
Epoch 3/5
45/45 [==============================] - 47s 1s/step - loss: 1.8358 - acc: 0.6363 - val_loss: 1.8073 - val_acc: 0.7579

Epoch 00003: val_acc improved from 0.66316 to 0.75789, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.03
-1.81.hdf5
Epoch 4/5
45/45 [==============================] - 48s 1s/step - loss: 1.7960 - acc: 0.7031 - val_loss: 1.7753 - val_acc: 0.8105

Epoch 00004: val_acc improved from 0.75789 to 0.81053, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.04
-1.78.hdf5
Epoch 5/5
45/45 [==============================] - 48s 1s/step - loss: 1.7802 - acc: 0.6911 - val_loss: 1.7440 - val_acc: 0.8000

Epoch 00005: val_acc did not improve from 0.81053
```

LR: 0.0001

```
Epoch 1/5
45/45 [==============================] - 59s 1s/step - loss: 1.8315 - acc: 0.6053 - val_loss: 1.6435 - val_acc: 0.7684

Epoch 00001: val_acc improved from -inf to 0.76842, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.01-1.
64.hdf5
Epoch 2/5
45/45 [==============================] - 48s 1s/step - loss: 1.5666 - acc: 0.8075 - val_loss: 1.3989 - val_acc: 0.8842

Epoch 00002: val_acc improved from 0.76842 to 0.88421, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.02
-1.40.hdf5
Epoch 3/5
45/45 [==============================] - 47s 1s/step - loss: 1.3801 - acc: 0.8151 - val_loss: 1.2404 - val_acc: 0.8737

Epoch 00003: val_acc did not improve from 0.88421
Epoch 4/5
45/45 [==============================] - 47s 1s/step - loss: 1.2200 - acc: 0.8840 - val_loss: 1.0989 - val_acc: 0.9053

Epoch 00004: val_acc improved from 0.88421 to 0.90526, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.04
-1.10.hdf5
Epoch 5/5
45/45 [==============================] - 48s 1s/step - loss: 1.0747 - acc: 0.9009 - val_loss: 1.0047 - val_acc: 0.8947

Epoch 00005: val_acc did not improve from 0.90526
```

LR: 0.0005

```
Epoch 1/5
45/45 [==============================] - 60s 1s/step - loss: 1.6994 - acc: 0.6430 - val_loss: 1.1012 - val_acc: 0.8737

Epoch 00001: val_acc improved from -inf to 0.87368, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.01-1.
10.hdf5
Epoch 2/5
45/45 [==============================] - 48s 1s/step - loss: 1.0658 - acc: 0.8044 - val_loss: 0.7832 - val_acc: 0.8526

Epoch 00002: val_acc did not improve from 0.87368
Epoch 3/5
45/45 [==============================] - 48s 1s/step - loss: 0.7700 - acc: 0.8482 - val_loss: 0.6979 - val_acc: 0.8737

Epoch 00003: val_acc did not improve from 0.87368
Epoch 4/5
45/45 [==============================] - 48s 1s/step - loss: 0.6155 - acc: 0.8912 - val_loss: 0.5812 - val_acc: 0.9053

Epoch 00004: val_acc improved from 0.87368 to 0.90526, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.04
-0.58.hdf5
Epoch 5/5
45/45 [==============================] - 48s 1s/step - loss: 0.4527 - acc: 0.9263 - val_loss: 0.4227 - val_acc: 0.9474

Epoch 00005: val_acc improved from 0.90526 to 0.94737, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.05
-0.42.hdf5
```

LR: 0.005

```
Epoch 1/5
45/45 [==============================] - 59s 1s/step - loss: 1.2813 - acc: 0.5539 - val_loss: 0.6220 - val_acc: 0.8737

Epoch 00001: val_acc improved from -inf to 0.87368, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.01-0.
62.hdf5
Epoch 2/5
45/45 [==============================] - 48s 1s/step - loss: 0.6207 - acc: 0.7784 - val_loss: 0.5370 - val_acc: 0.8000

Epoch 00002: val_acc did not improve from 0.87368
Epoch 3/5
45/45 [==============================] - 48s 1s/step - loss: 0.5098 - acc: 0.8147 - val_loss: 0.4651 - val_acc: 0.8842

Epoch 00003: val_acc improved from 0.87368 to 0.88421, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.03
-0.47.hdf5
Epoch 4/5
45/45 [==============================] - 48s 1s/step - loss: 0.5281 - acc: 0.8151 - val_loss: 0.3635 - val_acc: 0.8947

Epoch 00004: val_acc improved from 0.88421 to 0.89474, saving model to C:\Users\cotua\Desktop\python scripts\Weights\weights.04
-0.36.hdf5
Epoch 5/5
45/45 [==============================] - 48s 1s/step - loss: 0.4376 - acc: 0.8745 - val_loss: 0.6313 - val_acc: 0.6421

Epoch 00005: val_acc did not improve from 0.89474
```

LR: 0.001

Fig 2: Learning Rate Optimization

Hyper-parameter optimization for Adam optimizer learning rate. The value of 0.0005 was ultimately selected based on validation accuracy.

References

[1] ChildStats.gov. "Share of Old Age Population (65 Years and Older) in The Total U.S. Population from 1950 to 2050." Statista, Statista Inc., 23 Sep 2020, https://www.statista.com/statistics/457822/share-of-old-age-population-in-the-total-us-population/.

[2] K. Davis et al., "Activity recognition based on inertial sensors for Ambient Assisted Living," 2016 19th International Conference on Information Fusion (FUSION), Heidelberg, Germany, 2016, pp. 371-378.

[3] Huang, Hua. (2016). Toothbrushing Monitoring using Wrist Watch. 202-215. 10.1145/2994551.2994563.

[4] Dhall, Abhinav & Goecke, Roland & Gedeon, Tom & Sebe, Nicu. (2016). Emotion recognition in the wild. Journal on Multimodal User Interfaces. 10. 10.1007/s12193-016-0213-z.

[5] Y. Bin, Y. Yang, F. Shen, N. Xie, H. T. Shen and X. Li, "Describing Video With Attention-Based Bidirectional LSTM," in IEEE Transactions on Cybernetics, vol. 49, no. 7, pp. 2631-2641, July 2019, doi: 10.1109/TCYB.2018.2831447.

[6] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. In IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[7] Donahue, J.; Hendricks, L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; Darrell, T. Long-term recurrent convolutional networks for visual recognition and description. IEEE Trans. Pattern Anal. Mach. Intell. 2017, 39, 677–691

[8] L. Fei-Fei and O. Russakovsky, Analysis of Large-Scale Visual Recognition, Bay Area Vision Meeting, October, 2013

[9] J. Carreira, E. Noland, C. Hillier, and A. Zisserman. A short note on the kinetics-700 human action dataset. arXiv preprint arXiv:1907.06987, 2019.

[10] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In 2011 International Conference on Computer Vision, pages 2556–2563, 2011.