Activity and Emotion Detection for “Smart Homes”

Inimfon Akpabio

April 29, 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 to 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 multiple models as well as a baseline model to detect various human actions in a smart home setting. The baseline model will serve as a comparison of results. Our models start with a deep convolutional network (convent) used for feature extraction and then feed into a Long Short Term Memory (LSTM) network + Fully-connected layers to perform the final prediction. Each model defers in the type of convent employed, the nature of the LSTM network (regular or bidirectional), and preprocessing steps applied. The convent/feature-extractor uses pre-trained image-net weights. For each video, we sample only 10 frames per second except for the baseline model which samples only 5 frames. A mixture of videos from both the Kinetics700 and HMDB51 datasets were used to train the overall classifier. We train this model for different classes of human actions and achieve positive test accuracies as high as 90% which completely outclasses the baseline. Finally, due to the inefficiency of our method, we propose a more efficient approach in section 8 of this document.

1. Topic

The selected action for submission 10 is: “Arrange flowers or water plants etc”.

Topics for the previous submissions:

“Brushing Teeth”, “Wash_dye_comb_cut_or_blow_dry_etc_for_hair”, “Shaking Hands”, “Reading”, “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 of 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 multiple models that rely solely on the video modality to detect each of the actions listed in section 1.

3. Related Works

While this model design is not directly based on any particular paper, multiple research endeavors 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.

In [6], image data is combined with wearable sensor data to detect 16 different techniques of brushing teeth. In this approach, a single stream convent trained to recognize activities in images is used to extract features. The second to last dense layer of this convnet is then merged with the output of an LSTM network which processes the signals from the wearable device. Three different models based on CIFAR-10, Inception-V3, and VGG19 were developed. The VGG-19 was able to achieve an impressive classification accuracy of 85.4%. It must be noted, however, that these models were trained to perform intra-class prediction according to the bass brushing techniques.

4. Proposed model

Here we discuss all developed models and state the topics they were applied to.

4.1 Baseline model - “Brushing Teeth”

The initial model was developed to handle the action of “brushing teeth”. This model’s structure was similar to that of VGG16 [7] but without any pre-trained weights. See fig. 1 for Keras based representation. For this model, we first trained the network from scratch on positive and negative images of people brushing their teeth. The images were obtained from a combination of the GettyImages website and the Stanford40 action dataset [8]. Initially, the images were simply divided into “brushing teeth” and “not brushing teeth” and fed directly to the model for retraining. However, the model exhibited very poor performance, yielding only about 52% accuracy on train and validation sets. To alleviate this, the classes were expanded to eight. The eight classes were comprised of “brushing teeth” and 7 “not brushing teeth” classes. This simple, yet effective adjustment increased both train and validation performance by at least +30%. Data Augmentation was also applied to prevent overfitting.

At the end of the VGG16-esque network, we perform a global max pool and feed the features into a GRU network with 64 units. GRU has been shown to be comparable to LSTM in performance and yet more efficient since they lack a memory cell [9]. Finally, the output of the GRU 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. We chose to use 2 Softmax units due to the poor results obtained from a single fully-connected layer unit with a sigmoid activation function. We observed that categorical cross-entropy loss significantly improves the training performance on videos. This design decision was also implemented in future models. The model takes as input a (5, 224, 224, 3) vector which represents 5 frames each resized to 224x224 with RGB channels. It outputs an array of size 2 representing the prediction probabilities for "brushing teeth" and "not brushing teeth".

4.2 DenseNet-LSTM - "Wash_dye_comb_cut_or_blow_dry_etc_for_hair", "Reading"

Fig 2 shows the complete DenseNet-LSTM model design. Note that this structure is also adopted in subsequent models. This design was applied to both "Wash_dye_comb_cut_or_blow_dry_etc_for_hair" and "Reading" and brought about huge performance increases. The model starts with a DenseNet121 network [10] which has been pre-trained with ImageNet weights, hence, making it ideal for feature extraction.

Densely connected convolution networks are special types of convolution networks typically used for image classification tasks. They can curb the problem of input information disappearing as you descend deeper into the layers of a deep neural network. It does this using "deep" connections. This involves connecting each layer to all of its previous layers in a network while preserving feature size to allow for an unhindered flow of information between layers. This architecture typically comprises dense blocks, each of which contains densely connected composite functions. Between these blocks are transitional layers that carry out convolution and pooling operations.

At the end of the DenseNet121 network, we perform a global max pool which 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 an LSTM network with 64 units. Finally, the output of the LSTM 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]. They are also centered to improve training convergence. 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 detecting the action and not detecting the action.

4.3 DenseNet-BLSTM - "Shaking Hands", "Blowing nose"

This model is very similar to the previous model except for the use of a bidirectional LSTM. The advantage of this model is the ability to handle sequences in both forward and reverse directions, unlike the regular

LSTM which only learns in the forward direction. The drawback of this model is a 2-fold increase in the number of parameters of the LSTM network. Hence, in situations where the performance improvement over the previous model is minimal, we opt for the previous model per Occam's razor. The input and output sizes remain the same as in the previous model.

4.4 Inception-BLSTM - "Arrange flowers or water plants etc"

Inception-BLSTM simply replaces the DenseNet network with an InceptionV3 [11] network. This model was ultimately employed for the final topic and yielded a +4% improvement in testing accuracy.

The first Inception architecture was presented as Inception-v1/GoogleNet and finished second in the ImageNet [12] competition. Inception-v3 comes with improvements to the initial architecture such as Batch normalization, factorizing convolutions, efficient grid size reductions, and label smoothing. These improvements ultimately made the InceptionV3 network more efficient and yet more accurate. Batch normalization plays an important role in stabilizing the learning process by smoothing the optimization landscape in addition to reducing internal covariate shift thus making it easier to converge. Factoring convolutions involves breaking larger convolution filter operations into smaller ones thereby using fewer parameters. For instance, while a single 5x5 filter would use 25 parameters, 2 3x3 filters would require 18 parameters. Hence we achieve parameter reduction without sacrificing efficiency. A similar technique is also used in VGGNet. A deeper discussion of these improvements is detailed in the original paper. In summary, InceptionV3 offers high computational efficiency and accuracy with less complexity than VGGNet.

It must be noted that while InceptionV3 has less complexity than VGGNet, it still has over 3 times more parameters than DenseNet. Consequentially, training this model proved to be more tasking on our compute resources. The input and output sizes are modeled to match that of DenseNet-BLSTM.

4.5 Experimental models:

These models were trained and tested but were not deployed due to certain limitations, poor performance, or overfitting concerns.

4.5.1 Subtractive Net - "Blowing nose", "Arrange flowers or water plants etc"

This approach was originally inspired by [13]. We have taken this idea a step further by combining it with our previously developed models and achieved interesting results. The key feature of this approach is the preprocessing step applied before training. Whereas in previous models we simply sample N frames to represent an entire video, Subtractive Net samples (N + 1) frames and computes the abs diff between frames. Although this is a popular technique in computer vision, it has proved to be useful for tracking motion between frames in a video.

While this model yielded even better training/validation accuracy than the aforementioned models, we noticed that it only worked well on videos that are about 10 seconds in length or more. Hence, this proved to be a huge problem when the size of test videos was suddenly dropped to 2 seconds. Despite having a

high train/test accuracy, the test accuracy on 2 second YouTube clips drops to about 20%. For this reason, we decided not to use this model for direct action recognition.

4.5.2 Flow Net

Similar to the previous model, Optical Flow involves a preprocessing step of applying the FarneBack flow algorithm to input videos before passing them to the model. We used a window size of 15 and experimented with flow levels of 2, 5, 7, and 9. Other parameters such as iterations were kept at default. On visualizing the optical flow for different videos we immediately noticed that this method can easily get hampered when a frame/image is cluttered with too many background objects that are not necessarily important to the action in question (See fig 3). Additionally, the complexity of the algorithm made it extremely difficult to train. To alleviate this we tried using pre-trained weights as well as training from scratch. The former approach was hampered based on the fact that there are currently no comprehensive pre-trained Flow networks for general human activities. Most of the available pre-trained networks such as the UCL ground truth Optical flow dataset and the MPI Sintel dataset are based on animated objects and hence, do not apply very well to actual human beings. Nevertheless, we believe that poor performance can be improved given more time and better computing resources. We wish to re-visit this approach in the future.

4.5.3 Ensemble

Combining multiple networks to perform prediction is a popular and widely used technique in the machine learning community. After training DenseNet-BLSTM and Inception-BLSTM, we decided to train a network on top of them that could take their output probabilities and correctly predict whether or not the action was detected. We noticed that by varying the number of neurons in a single hidden layer, and even the number of hidden layers we were able to get the accuracy of the ensemble model to somewhere between 97% and 100%. The concern with this approach is overfitting. Despite its impressive performance, it is likely that this model will not generalize very well to new/unseen data points.

5. Dataset

Feature Extractor:

The feature extractor for each of the models except for the baseline model was pre-trained on ImageNet. Most of the proposed solutions to activity recognition often employ pre-trained models due to their robustness and superior ability to extract features. The baseline model's feature extractor, on the other hand, was trained on images obtained from a combination of the GettyImages website and the Stanford40 action dataset.

Full model:

Finding a video dataset to train the full model proved difficult. The Kinetics700 dataset [14] is a large dataset that contains about 650,000 human-performed actions spread across 700 categories. Each video is roughly about 10 seconds long. The Kinetics700 dataset has been widely explored for video recognition tasks in recent years. Due to its massive size, it could take days to download the entire dataset even with a stable internet connection. We decided to combine this with the HMDB51 dataset [15] which similarly contains videos of human-performed actions spread across 51 classes with each class containing at least 100 clips. Video lengths range from 2 seconds to 10 seconds but mostly 10 seconds. For training, the videos were separated into two classes: "Action name" and "miscellaneous" which represents the collection of all videos that contain the action and videos that don't contain the action. Table 1 shows the sizes of the training set for each topic.

Action name	**Positive Sample Count**	**Negative Sample Count**
"Brushing Teeth"	255	363
"Wash_dye_comb_cut_or_blow_dry_etc_for_hair"	538	412
"Shaking Hands"	512	512
"Reading"	786	750
"Blowing nose"	388	382
"Arrange flowers or water plants etc"	484	484

Table 1: Training dataset positive and negative sample size for each topic.

As a preprocessing step, the videos were sample-centered and standardized to improve convergence during training. It is a common practice to centralize and normalize data before training neural networks. No data Augmentation was applied to avoid corrupting important landmarks and features that could be used for prediction.

6. Model Training and Performance

The models were trained on a windows 10 operating system coupled with an NVidia 1070 graphics card equipped with 1920 CUDA cores. Other specifications include a clock speed of 1683 MHz and 8 GB of RAM.

It is worth stating that these results are not necessarily a true indicator of model performance since the YouTube clips had to be manually hand-picked. Since the goal of this project was to detect as many moments as possible, YouTube videos with more moments were naturally favored. This inevitably increases the accuracy, however, in the real world this might not always be the case.

6.1 General Hyper-parameter Tuning

Optimization was performed for parameters such as batch size, epochs, and learning rate. These were selected by testing pre-selected values in fixed uniform ranges and choosing the values that yielded the best convergence. The chosen batch size for each topic is shown in table 2. Fig. 4 shows a snippet of the learning rate optimization process for 5 out of the 6 candidate values. We observed that when paired with the Adam optimizer as opposed to SGD, values in the range 0.0005 to 0.001 yielded the best results. A dropout percentage value of 50% was found to be ideal probably due to how much overfitting occurs with such large models. Additionally, we fixed the maximum number of epochs to 30 (except for the baseline model) and employ both model check-pointing as well as early stopping to curb overfitting. On average, most models stopped after about 10-12 epochs.

Action name	Batch Size
“Brushing Teeth”	32
“Wash_dye_comb_cut_or_blow_dry_etc_for_hair”	19
“Shaking Hands”	32
“Reading”	16
“Blowing nose”	22
“Arrange flowers or water plants etc”	22

Table 2: Selected Batch Size from Hyper-parameter optimization for each topic

6.2 Model specific training and performance

6.2.1 Baseline model - “Brushing Teeth”

About 80 epochs were used to train the baseline model. Initially, the model began to exhibit heavy overfitting. This unwanted phenomenon was curbed by adding L2 regularization and a dropout of 20% to the fully connected layers. Additionally the training videos were also trimmed for this topic. On the training set, the model achieves a reasonable accuracy of about 87% and about 80% on the validation set. The performance on the “Brushing Teeth” YouTube test set is about 70%. While this performance was decent for a freshly trained model, we felt there was still plenty of room for improvement.

6.2.2 DenseNet-LSTM - “Wash_dye_comb_cut_or_blow_dry_etc_for_hair”, “Reading”

“Wash_dye_comb_cut_or_blow_dry_etc_for_hair”

For the topic “Wash_dye_comb_cut_or_blow_dry_etc_for_hair”, we set the number of epochs to 30 as explained in 6.1 but experienced early stopping after about 12 epochs. 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. The training accuracy is as high as 99% while on the validation set, the model achieves an accuracy of about 93% and an overall accuracy of 91% on YouTube videos. The breakdown is about

94.8% on positive (contains desired action) YouTube videos and 87.25% on negative YouTube videos. Quite noticeably, the model seems to experience dips in performance when dealing with hairstyles that are uncommon in the dataset. For instance, the accuracy of test clips containing wavy, African-American hair is less than 90%. This is obviously due to the lack of diversity in our training data. This also alludes to the common issue of lack of representation in machine learning. If such systems are to be deployed in real life, they should be robust enough to handle different demographics.

"Reading"

For "Reading", due to heavy overfitting, we applied L2 regularization and set the dropout at 50%. As a result, we were able to achieve a validation accuracy of 95 % but on the YouTube test set, the accuracy drops to 85.8 %. The breakdown is about 88.7% on positive (contains desired action) YouTube videos and 83% on negative YouTube videos. While this performance is still impressive and exceeds that of the baseline model, it indicates that the model suffers slightly from overfitting and does not generalize perfectly on foreign data points. Another issue with this topic was the dataset. The Kinetics dataset for "Reading" contains clips of people just holding up book cover or just zooming into one of the pages. A lot of these videos may not necessarily count as reading depending on who is judging. As a result we had to employ more leniency to account for this flaw.

6.2.3 DenseNet-BLSTM - "Shaking Hands", "Blowing nose"

"Shaking Hands"

Similar to DenseNetLSTM, we apply fine-tuning, regularization, and dropout to curb overfitting. For the topic of "Shaking Hands", DenseNet-BLSTM outperforms DenseNet-LSTM during training with a training accuracy of 99.6% at 14 epochs. On the validation set, the model achieves an accuracy of about 90.6% and an overall accuracy of 81.6% on YouTube videos. The breakdown is about 90.43% on positive (contains desired action) YouTube videos and 72.5% on negative YouTube videos. Despite the poor performance on negative videos, its ability to learn unconventional handshakes such as daps, high-5, fist bumps, etc. is noteworthy. This is probably attributable to the robustness of BLSTMs.

"Blowing nose"

This proved to be the most difficult out of all the topics attempted so far despite achieving a high training accuracy of 99 % at 10 epochs. On the validation set, the model achieved an accuracy of about 88 % 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. This performance is considerably decent but still leaves a lot of room for improvement.

6.2.4 Inception-BLSTM - “Arrange flowers or water plants etc”

With the Inception-BLSTM we decided not to apply any fine-tuning due to memory constraints on our computer. This decision is also validated by the high accuracy achieved on train, validation, and test sets. The training and validation accuracies are 99.9 % and 92.7 % respectively which is much higher than the baseline model. Similarly, the accuracy on positive YouTube video is about 94 % which completely trumps the baseline model. As we mentioned it the beginning of this model, these results should not be taken at face value as they greatly depend on the test set. The test videos for this entire project have been chosen to increase the total number of identifiable clips. On the other hand, they are good for comparing the efficiency of one model with another.

7. YouTube

7.1 Moment detection

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 confirmed 2-second subclips from the original video. It must be noted that this approach is not very efficient and in section 8 we will present a new approach to alleviating this.

7.2 View Moments in iLab

The number of found moments for each topic is listed in table 3. To view the clips for a particular action and uploaded them for this project, follow these steps:

1. Click on the Search widget at the top of the iLab website.
2. On the left panel, look for the entry corresponding to the action and click on “see clips”.
3. After step 2, there should be statistics and 3 pie charts in the center panel for the action that was selected.
4. Go to the “Observers” pie chart and click on the desired username. The username format for this project is “csce636spring2021-nini16-{Submission number}”
5. Now return to the main page and click on the Videos widget at the right of the screen.

Action name	Number of found moments
“Brushing Teeth”	319
“Wash_dye_comb_cut_or_blow_dry_etc_for_hair”	403
“Shaking Hands”	333
“Reading”	1476
“Blowing nose”	224
“Arrange flowers or water plants etc”	255

Table 3: Number of found moments for each action.

7.3 Performance/Accuracy

For the method described in 7.1, it is quite easy to calculate the false-positive ratio and the false-negative ratio. The formulas for these are shown below. Positive samples are clips that are human confirmed to contain the action. Negative samples are clips that are human confirmed to not contain the action. See table 4 for a list of these metrics for each action.

$$FPR = \frac{number\ of\ negative\ samples\ were\ action\ is\ detected}{number\ of\ negative\ samples}$$

$$FNR = \frac{number\ of\ positive\ sample\ were\ action\ is\ not\ detected}{number\ of\ positive\ samples}$$

Action name	FPR (%)	FNR (%)
"Brushing Teeth"	40	19
"Wash_dye_comb_cut_or_blow_dry_etc_for_hair"	12.75	5.2
"Shaking Hands"	27.5	9.57
"Reading"	17	11.3
"Blowing nose"	20.33	17
"Arrange flowers or water plants etc"	10	6

Table 4: FPR and FNR for each topic.

7.4 Efficiency

Note that while these values look satisfactory, the method for collecting clips is not very efficient. Our method involves dividing a video into 2 second clips and then sampling 20 equally spaced frames to pass to the model. Additionally we use human help to separate the clips into positive and negative samples before calculating the FPR and FNR. This method is quite inefficient. In section 8 we will present a more efficient method.

Quantifying the efficiency of this method is quite difficult. As an example, we will use the topic "Reading" without a 30% positivity constraint. On average, about ~30 - 40 moments were found per 3-minute video. This equates to about ~600 - 800 moments per hour. However, note that this does not accurately evaluate the efficiency of this approach because actions like "Reading" tend to produce more moments. Perhaps another way to quantify the efficiency is to see how many frames need to be sampled per hour. In the case of our model, this amounts to:

$$Number\ of\ sampled\ frames\ in\ an\ hour = (3600\ s) * (10\ frames) = 36000\ frames$$

While this is achievable with a decent amount of computing resources, it is not satisfactory and can be improved.

8. Improve Accuracy

Rather than segmenting the entire video into 2-second clips, we suggest segmenting the video into 10-second clips and extracting 50 frames from each sub-clip. The key is to train a separate model that is capable of masking out human motion from these 10-second candidate clips. This can be achieved in multiple ways. An easy solution might be to employ a pre-processing module to crop out human beings from the background. Another plausible approach would be segmentation. In [16], the authors employ different variations of sparse Optical flow to automatically segment an input image and track motion. By segmenting out human activity, we can achieve better performance on activity recognition tasks. Given better training performance and fine-tuning, possible candidates for this include our Subtractive Net and Flow Net which were introduced in section 4.5 earlier. On the candidate clips, we can then further segment these into 2-second clips and run them through our Inception-BLSTM to obtain predictions.

The idea is inspired by the observation that over longer periods, background information may become irrelevant towards action recognition. Conversely, it carries huge importance for shorter videos. We noticed this behavior, especially when dealing with Subtractive Net. Unfortunately, we were unable to successfully implement this idea with Subtractive-Net and Flow Net due to poor accuracy (~20 %). Nevertheless, we are confident that with a properly trained model, this technique can be effective.

9. GitHub links

The link to the Github Repo is https://github.com/nini16/VideoActionEmotionRecognition

The tutorial videos describing how to run both notebooks are contained in the google drive:

https://drive.google.com/drive/folders/16AT5Sm-o4sx_PE3Dgls_wai2Py3kqgQt?usp=sharing

Note that within the Google drive, the folder for each submission is labelled as “v{submission number}” (i.e. v2, v3, v5, v6, v8, v10)

Please note that the notebooks consume a heavy amount of RAM and as such may crash on Google-Colab. A separate computer was required to run the train and test notebooks.

Figures

```
momentum = .9
model = keras.Sequential()
model.add(Conv2D(64, (3,3), input_shape=shape,
    padding='same', activation='relu', name='conv1'))
model.add(Conv2D(64, (3,3), padding='same', activation='relu', name='conv2'))
model.add(BatchNormalization(momentum=momentum))

model.add(MaxPool2D())

model.add(Conv2D(128, (3,3), padding='same', activation='relu', name='conv3'))
model.add(Conv2D(128, (3,3), padding='same', activation='relu', name='conv4'))
model.add(BatchNormalization(momentum=momentum))

model.add(MaxPool2D())

model.add(Conv2D(256, (3,3), padding='same', activation='relu', name='conv5'))
model.add(Conv2D(256, (3,3), padding='same', activation='relu', name='conv6'))
model.add(Conv2D(256, (3,3), padding='same', activation='relu', name='conv7'))
model.add(BatchNormalization(momentum=momentum))

model.add(MaxPool2D())

model.add(Conv2D(512, (3,3), padding='same', activation='relu', name='conv8'))
model.add(Conv2D(512, (3,3), padding='same', activation='relu', name='conv9'))
model.add(Conv2D(512, (3,3), padding='same', activation='relu', name='conv10'))
model.add(BatchNormalization(momentum=momentum))

model.add(MaxPool2D())

model.add(Conv2D(512, (3,3), padding='same', activation='relu', name='conv11'))
model.add(Conv2D(512, (3,3), padding='same', activation='relu', name='conv12'))
model.add(Conv2D(512, (3,3), padding='same', activation='relu', name='conv13'))
model.add(BatchNormalization(momentum=momentum))
```

Fig 1: Keras based representation of feature extractor for baseline model.

Notice that this model includes Batch Normalization after the convolutional layers

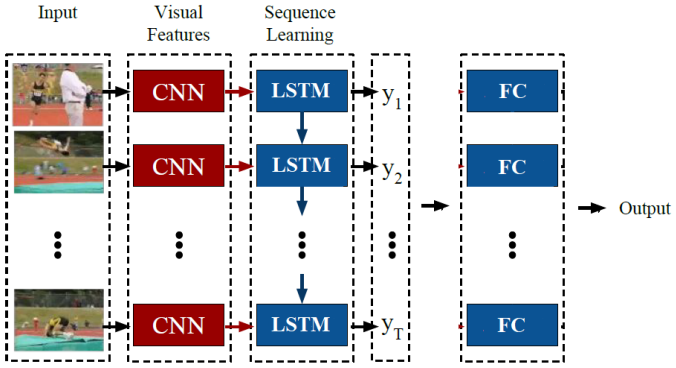

Fig 2: Model Diagram

Source: Adapted from [16]

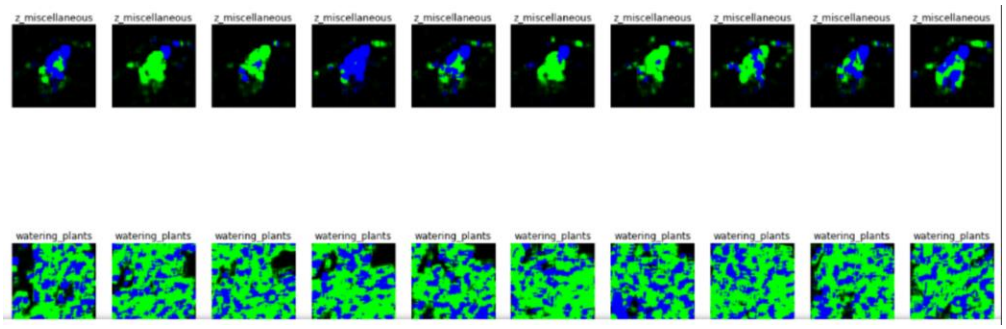

Fig 3: FarneBack Optical Flow

Visualization of Optical Flow algorithm on watering plants. The original frame is cluttered with plants in the background making this approach less effective.

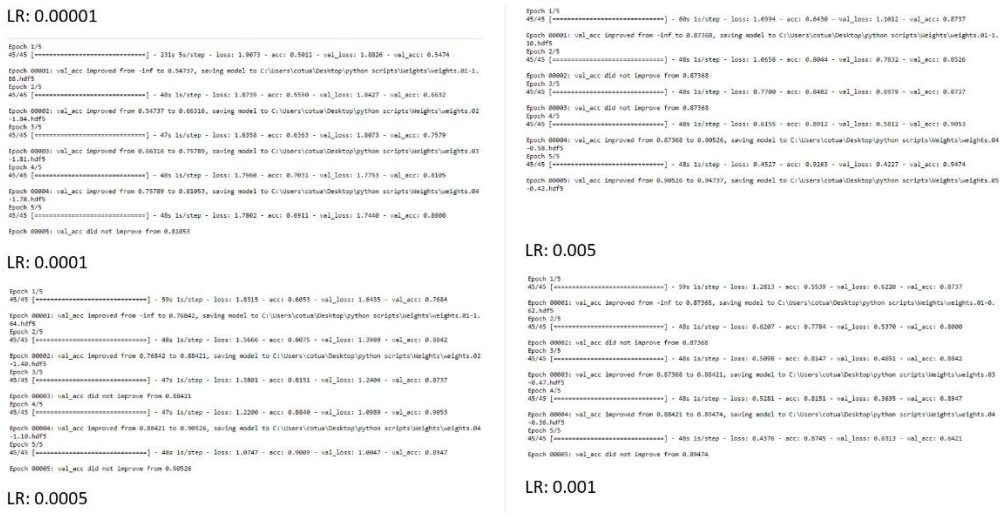

Fig 4: Learning Rate Optimization

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] M. Jiang et al., "Teeth-Brushing Recognition Based on Deep Learning," 2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), Taichung, Taiwan, 2018, pp. 1-2, doi: 10.1109/ICCE-China.2018.8448684.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[8] B. Yao, X. Jiang, A. Khosla, A.L. Lin, L.J. Guibas, and L. Fei-Fei. Human Action Recognition by Learning Bases of Action Attributes and Parts. Internation Conference on Computer Vision (ICCV), Barcelona, Spain. November 6-13, 2011.

[9] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, Dec. 2014, [online] Available: http://arxiv.org/abs/1412.3555.

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

[11] Santurkar S, Tsipras D, Ilyas A, How does batch normalization help optimization?[C]//Advances in neural information processing systems. 2018: 2483-2493.

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

[13] He, Z., Jin, T., Basu, A., Soraghan, J., Di Caterina, G., Petropoulakis, L.: Human emotion recognition in video using subtraction pre-processing. In: Proceedings of the 2019 11th International Conference on Machine Learning and Computing, pp. 374–379 (2019)

[14] 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.

[15] 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.

[16] 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