Generalization of Deep Learning Models on First-person Videos

Yu-Cheng Lin, Wei-Chih Chen and Chia-Chih Ho

I Overview: As the first-person cameras like GoPro become popular for some sports hobbyists and geeks, there would be more first-person videos in the near future. Most of the video online, such as youtube, are shot in third person video. And, the research of video action recognition on first-person videos is still in the beginning. We want to see if the deep learning models can also apply on the recognition of the first-person videos and whether they perform well as for the images and third-person videos.

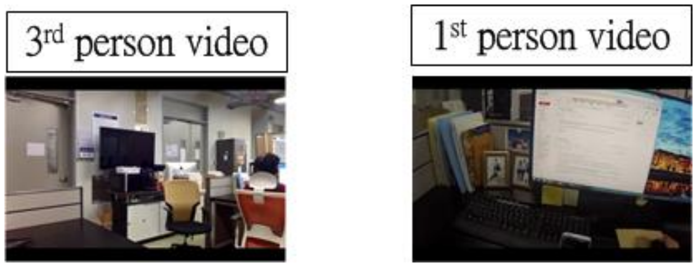


Figure 1. Left is a frame of 3rd person video and right is of 1st person video.

In our experiments, we trained 1st person and 3rd person videos on classifiers and try to predict labels on test videos which shown in Figure 2 and Figure 3. In this experiment, we used three neural networks to be our classifier and try to find their accuracy rate on test data. At the end of this project, we want to know whether CNN are good model for video classification and how to improve classification accuracy rate for networks.

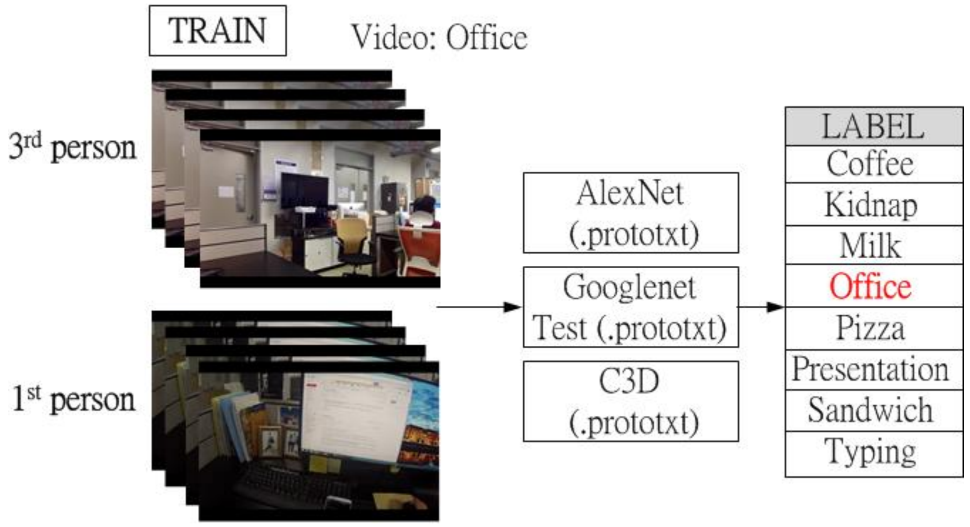
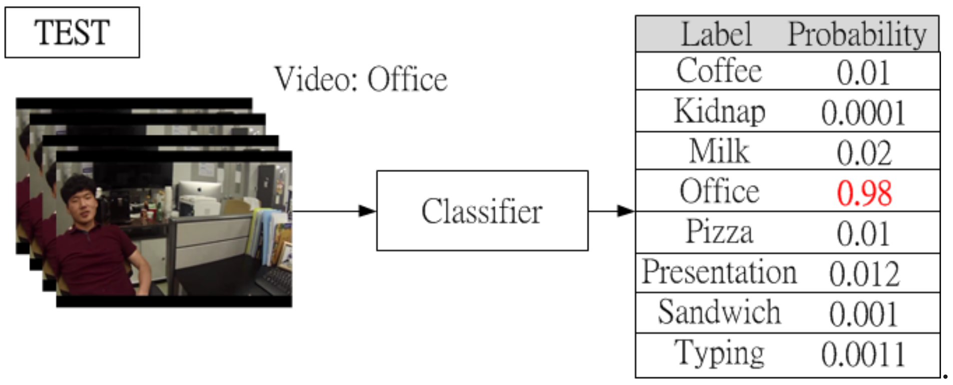


Figure 2. Training three different networks by 1st and 3rd person videos



Label	Probability
Coffee	0.01
Kidnap	0.0001
Milk	0.02
Office	0.98
Pizza	0.01
Presentation	0.012
Sandwich	0.001
Typing	0.0011

Figure 3. Testing model and results for classification.

In the following paragraphs, we describe three networks in part two and talk about the dataset for this experiment in part three. In part four, we show how to implement this experiment by Caffe[1] and analysis the result on part five. And conclusion is in part six.

II Network Introduction: Convolutional neural networks (CNNs) did great improvement on visual recognition since AlexNet [2]. In addition to applying CNNs for image recognition, detection, segmentation and retrieval, some papers use CNNs to combine image information across videos [3][4]. In here, we briefly discuss three networks used in this project such as AlexNet, GoogleNet [5] and C3D [6].

2.1 AlexNet: ImageNet Classification with Deep Convolutional Neural Networks

This paper described how to build neural network architecture in details to do image recognition job including using ReLu nonlinearity for speedup, training this model by two GTX580 GPUs, setting local response normalization to decrease test error rate and overlapping pooling. They also reduce overfitting by data augmentation and dropout machine learning method on this network. Overall, this paper proposed a 8-layers convolutional neural networks which contains five convolutional and three fully-connected layers. This model achieved a winning top-5 test error in ILSVRC-2012 competition. It was first group to use CNN in ILSVRC-2012 competition and made a tremendous improvement on error rate that from previous stat-of-art 26.2% to 16.4%.

2.2 GoogleNet: Deeper convolutions

While deep learning with more concretely convolutional networks, the quality of image recognition and object detection can be improved. The biggest gains are from the synergy of deep architectures and classical computer vision, like the R-CNN algorithm. The efficiency of deeper-convolution algorithms keeps a reasonable computational budget and could be put to real world use, even on large datasets. For deep neural network architecture, it implements a new level of organization in the form of the “Inception module” as a logical culmination and increased network depth.

Contrary to the fixed 2-layer deep model, all filters in the Inception model are learned. In the case of the GoogLeNet model, Inception layers are repeated many times, which leads to a 22-layer deep model. Network-in-Network is also applied to convolutional layers and viewed as additional one by one convolutional layers followed typically by the rectified linear activation. The biggest advantage is that it can be easily integrated in the current CNN pipelines. Also, there are dual purposes: the first purpose is for dimension reduction modules to remove computational limitation of the size of networks and the other is to increase not only the depth, but also the width without significant performance penalty.

In this method, it decomposes the overall detection problem into two sub-problems:

- Low-level cues such as color and superpixel consistency for potential object proposals
- CNN classifiers to identify object categories at those locations

To improve the performance of deep neural networks: one simple way is to increase their size. However, there are two major drawbacks: the higher probability for overfitting, which makes the creation of high quality training sets tricky and expensive, and the dramatically increased use of computational resources. To solve these problems, this method moves from fully connected to sparsely connected architectures, even inside the convolutions. It states that if the probability distribution of the data-set is representable by a large, very sparse deep neural network, then the optimal network topology can be constructed layer by layer by analyzing the correlation statistics of the activations of the last layer and clustering neurons with highly correlated outputs.

Since the overhead of lookups and cache misses is still dominant, it is very inefficient to calculate non-uniform sparse data structures. Furthermore, the gap can be widened by the use of steadily improving, highly tuned, numerical libraries. Non-uniform sparse models also require more sophisticated engineering and computing infrastructure.

For convolutions, it is implemented as collections of dense connections to the patches in the earlier layer, and random and sparse connection tables in the feature dimensions are used for better optimize parallel computing. This allows for utilizing efficient dense computation.

There is vast literature on sparse matrix computations, which suggests that clustering sparse matrices into relatively dense submatrices tends to give state of the art practical performance for sparse matrix multiplication. As for Inception architecture, after two iterations on the exact choice of topology, there are already modest gains. After further tuning of learning rate, hyperparameters and improved training methodology, it is more useful in the context of localization and object detection, at least locally optimal.

2.3 C3D: Generic Features for Video Analysis

We do not have a general way of representing videos and a universal descriptor for a video. This paper provides a solution for a general way of representing videos, which is very helpful for solving various large-scale video tasks in a homogeneous way. For a universal descriptor, there are three important properties to be useful at Internet scale.

First, this descriptor needs to be generic to represent different types of videos as well. Second, the desired descriptor needs to be compact for processing, storage, and retrieval. Last but not least, it should be efficient to applied to compute the features.

Pre-trained convolutional network(ConvNet) models are commonly used for extracting image features and the activations of the network's last few fully connected layers are shown to be useful for transfer learning tasks. However, when referring to the video domain, it is lacking such generic features due to two reasons. There is no large-scale supervised video dataset while it is not an efficient way to learn compact spatio-temporal features. This paper uses a large manually annotated dataset it builds to train a deep 3D ConvNet for learning video features. While it is a better model for spatio-temporal information, it applies better features for video analysis. Also, it gives better performance in action recognition.

In the paper, it trains the deepest 3D ConvNet model while it outperforms 2D ConvNet models by a good margin. The features have all the desirable properties: generic, compact, and efficient. Therefore, the author declares that they have come up with an approach for generic spatio-temporal feature learning based on the right choice of dataset design and an appropriate learning model using 3D ConvNet. It is a simple linear model, which can work as well as state-of-the-art performance. Also, the features are compact, discriminative and orders of magnitude so it is faster to compute compared with current best hand crafted features and current best deep learning model.

With availability of parallel machines (GPUs, CPU clusters) and large amounts of training data, convolutional neural networks provide breakthroughs on many AI problems. Despite the good performance, the method of 3D recognition is computationally intensive and becomes intractable on large-scale datasets. While the 3D ConvNets approach is designed for action classification which is very task-specific, this paper takes full video frames as inputs and does not rely on any preprocessing, making it easily applicable to larger scale and more generic video analysis tasks. The main methods are as follows:,

- Use 3D ConvNet as a method for feature learning
- Use the trained model as a feature extractor
- Apply a network which is much deeper.

Compared to 2D convolution's collapsing temporal information, the 3D convolution preserves it and passes it to the next layer. The model performs 3D convolutions and 3D pooling propagating temporal information throughout the network and learning temporal filters in all convolution layers.

2.4 Summary

In general, we implement three different architectures for recognition on first-person videos. The first two methods are originally for image classification. AlexNet is the first implement convolution neural network on ILSVRC 2012 competition. GoogleNet also achieves better improve accuracy rate of image classification based on AlexNet. It applies different size of filters in each layer that is good at capture local special structure to improve the performance. Therefore, in this project, we want to see whether it is still efficient for video recognition. After applying these two image classification methods, we want to compare their results with the C3D, which is designed for video recognition. The biggest difference is that it uses 16 frames as input than the first two. Therefore, network learns difference between frames which is better for spatial-temporal information.

III Data-Set: This dataset comprises 8 different labels for videos and each scene shot in three different views which have two still cameras in different positions and one GroPro. These videos in one scene are all time-synchronized. We got 191 video in total that shot in 63 different scenes and all of them are provided by a professor in Indiana University. In figure 4 shows how many videos shot in each category.

Label	Class	Video number
0	Coffee	18
1	Kidnap	56
2	Milk	3
3	Office	59
4	Pizza	15
5	Presentation	18
6	Sandwich	13
7	Typing	9
	Total	**191**

Figure 4. List all videos shot in different category for data set.

Since input of classifiers is an image, we use frames of video as input. But a video has more than four thousand frames. We sample 1/10 of total frames in a video for reducing training time. And within all of the dataset, we random sample three fourth of data as training set and one fourth of data as testing set.

IV How to implement by Caffe: In this section, we introduce how to train and test part on Caffe in practice.

4.1 Train

This experiment has implement on Linux by Caffe library. Caffe's Imagenet pre-train model adopts the same architecture as AlexNet. We also download C3D and GoogleNet library on github. However our network is not exactly the same as original networks since these networks were trained for 1000 categories. In our network, instead of using

1000 categories at the softmax layer, we set a new 10 categories softmax layer for our experiment. Our networks are shown in figure 5.

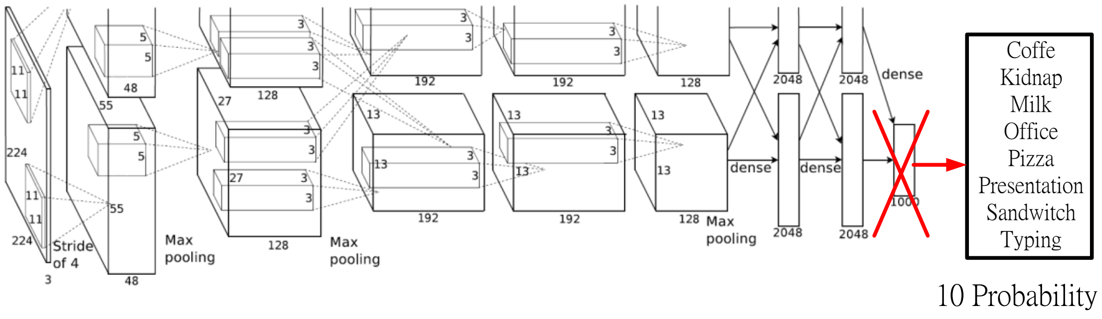


Figure 5. Difference between ImageNet and our classifier is on last softmax layer.

In Caffe, input format for training and testing are lmdb database format, so we used shell script to transfer frames which is jpg file to lmdb file. To implement network, there are three files to set that are solver, parameter weight and architecture file. Solver is a file to set for training environment, such as learning rate, step size, moment, etc. Parameter weight is a pre-train weight of previous network which filename extension is caffemodel. And, last, we can set architecture file to our desire network which filename extension is prototxt shown in figure 7. So if we want to retrain last softmax layer as 10 neurons, we set it up in this file. After setting these files, we can fine-tune our classifier based on previous network.

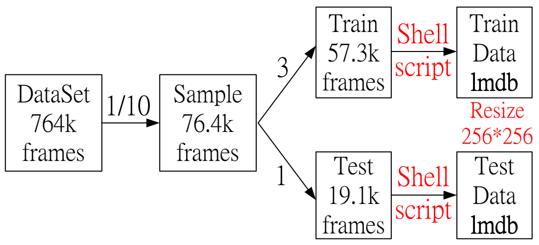


Figure 6. Block diagram to separate data set into train and test set.

We used stochastic gradient descent with batch size 32 to optimize. And all these three networks are trained on nvidia tesla K40c GPU. To take GoogleNet for example, fine-tune it took 36 hours for 170k iterations.

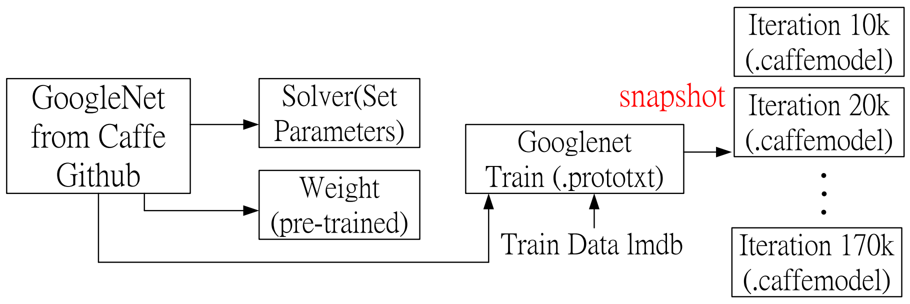

GoogleNet from Caffe Github
Solver(Set Parameters)
Weight (pre-trained)
Googlenet Train (.prototxt)
Train Data lmdb
snapshot
Iteration 10k (.caffemodel)
Iteration 20k (.caffemodel)
Iteration 170k (.caffemodel)

Figure 7. Block diagram to train classifier. This took GooglNet for example.

4.2 Test

For testing classifiers on Caffe, we need to set some parameter on architecture file for calculating final frame accuracy rate. And then based on the weight we trained; we tested data on classifiers and extracted features of last layer. Comparing label with highest probability of last layer, it's able to calculate frame accuracy rate. Since we try to classify the label of test video, we need to add the all the probability of frames in one video and get the video accuracy rate. As we mentioned before, the data processed in architecture is used lmdb database file. There needs a transformation to text file out of extraction feature process which is shown in Figure 8. Last, both database transformation and frame to video transform are built by python. We did this process on all of three architectures. Last, since we only have limited data set, we use both 1st person and 3rd person video for test verification.

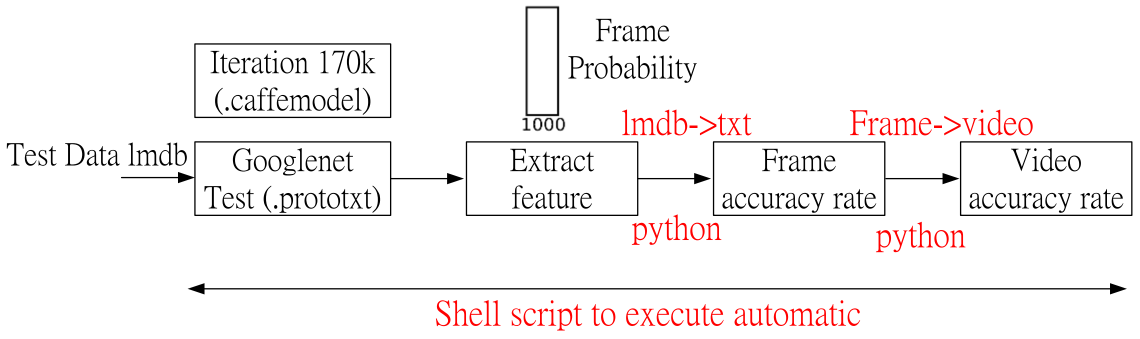


Figure 8. Block diagram to test classifier. This took GooglNet for example.

V Experiment result:

5.1 Results

In figure 9, it's shown the result of train GoogleNet on different iterations. Blue is frame error rate which is the accuracy rate of all frames of testing videos. Frame accuracy rate slightly increasing from 71% to 73% at iterations 10 thousands to 170 thousands which is shown at figure 10. Red is testing video accuracy rate.

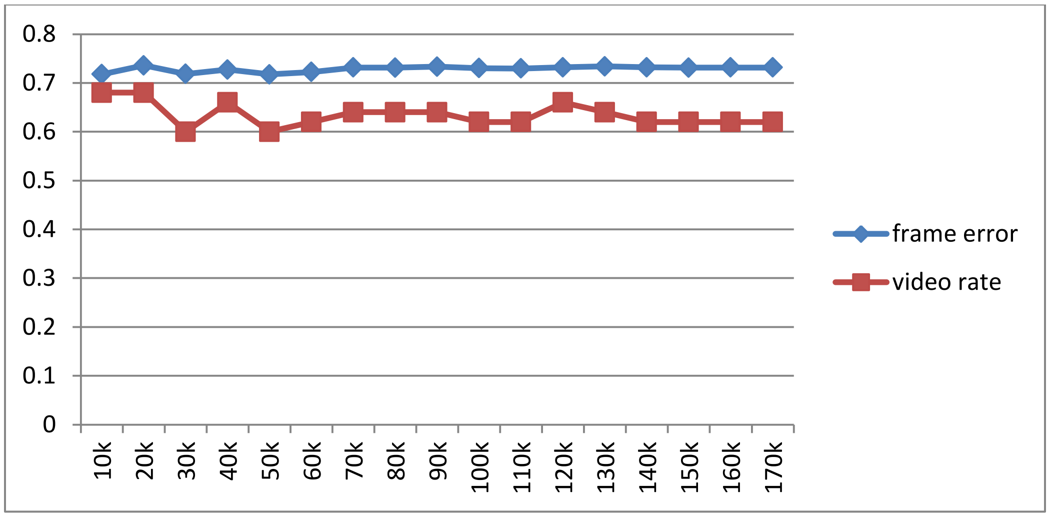


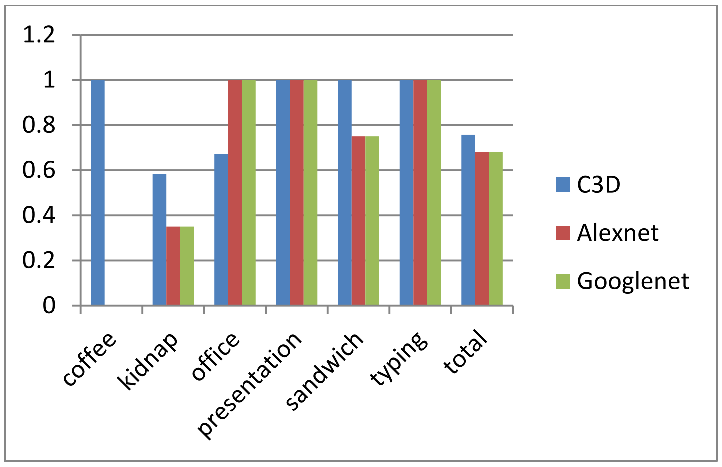
Figure 9. Frame and video accuracy rate of different iterations on GoogleNet.

This experiment is shown that increasing number of iterations truly increasing accuracy rate. However, it just gains a slightly different and may cause by overfitting. But the testing video accuracy rate is worse with more iteration. We will describe this in next section.

Iteration	Frame error	Video rate
10k	0.7178185	0.68
20k	0.735987	0.68
30k	0.71834608	0.6
40k	0.727380638	0.66
50k	0.717356898	0.6
60k	0.722302822	0.62
70k	0.73160116	0.64
80k	0.731139541	0.64
90k	0.73311791	0.64
100k	0.730150356	0.62
110k	0.729556845	0.62
120k	0.73199683	0.66
130k	0.733975204	0.64
140k	0.731864943	0.62
150k	0.731337378	0.62
160k	0.731667106	0.62
170k	0.73160116	0.62

Figure 10. Frame and video accuracy rate of different iterations on GoogleNet in numeric number.

In figure 11 right charts, we got the best testing result on C3D which is around 76% accuracy rate and 68% accuracy rate on both Alexnet and GoogleNet. To check accuracy rate in each category, it got lower accuracy rate on both coffee and kidnap video when comparing to other five classes shown in figure 11 left charts. The misclassify labels statistic is shown in figure 12 charts. Coffee videos are misclassified as kidnap and office on Alexnet and Googlenet; and kidnap videos are misclassified as office on all three classifiers. We will analyze these results in section 5.2.



Based on iterations 20K			
total	correct videos	video counts	accuracy
C3D	25	33	0.7575
Alexnet	34	50	0.68
Googlenet	34	50	0.68

Figure 11. Left is the testing video accuracy rate on three architectures by different class. Right is numeric data for the testing on three architectures.

label type	Networks	Misclassify label
Coffee	Alexnet	3 videos labeled as kidnap, 3 videos labeled as office
	Googlenet	2 videos labeled as kidnap, 4 videos labeled as office
Kidnap	C3D	all labeled as office
	Alexnet	all labeled as office
	Googlenet	all labeled as office
Office	C3D	all labelled as kidnap
Sandwich	Alexnet	all labeled as pizza
	Googlenet	all labeled as pizza

Figure 12. Labels of misclassified testing videos.

5.2 Analysis

Figure 12 is shown most of the misclassify videos are coffee, kidnap and office, so we try to analyze our data set to understand the possible reasons. We saw most of the test videos and chose three of them to represent in these three labels on figure 13. In figure 13, we use main content frames to stand for section of video and visualize it in time axis. To illustrate coffee video, first 5 seconds scene is video setup; following 25 second is a woman sit in front of an office desk; continuing 80 second is that woman goes to coffee machine to make a coffee; and the final 45 seconds is the woman goes back to her desk to drink coffee. This video total is 155 seconds long.

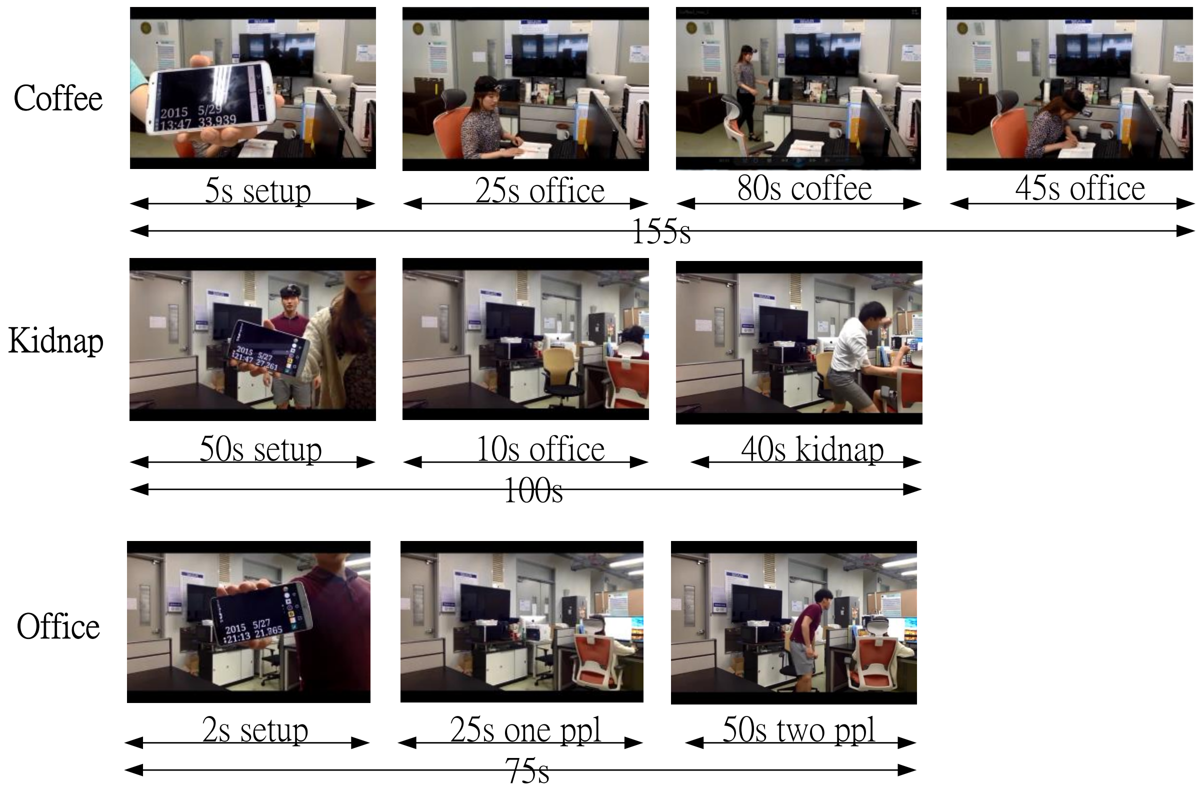


Figure 13. Videos compose what contents in three most misclassify video.

In Coffee video, there are almost 70 seconds is the same scene as office video which is almost half length of that video, since there is some possibility to misclassify coffee video as office. To compare kidnap video and office video, there are 40 seconds of kidnap characteristic for the fore one and 50 seconds of office characteristic for the following one. Both of scenes are a person sit in front of a desk and the other one walk to him. So only if the classifier learns person behavior, it has chance to predict correct. And this is also the reason why all kidnap video is classified as office.

As it explained in section two, GoogleNet is good at capturing local features that means it helps on frame accuracy rate. But the video we want to classify need to learn not only scene but also behavior of people. Therefore, higher frame accuracy rate can help distinguish significant feature of label but not behavior of human. Since C3D consider time features in the network, it is able to capture motion between different frames and to learn behavior of human. So GoogleNet and Alexnet cannot compete with C3D network.

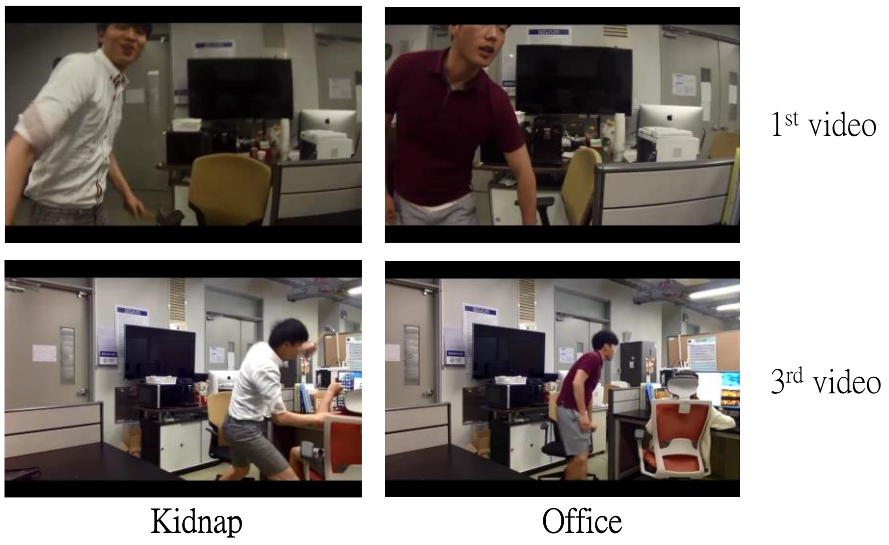


Figure 14. Classifiers learned limited information in 1st person video rather than 3rd person video.

VI Conclusion: At first, we did experiments on the popular deep CNN model, Caffe's Imagenet pre-train model, which adopts the same architecture as AlexNet. Based on this model, we fine-tune AlexNet by training both first-person and third-person point-of-view frames from videos. Then we use un-training dataset to evaluate classification accuracy rate of first-person and third-person point-of-view videos. Besides, to find a better network for recognizing first-person video, we also fine-tune C3D and GoogLeNet on Caffe. Results of three networks for C3D, Alexnet and GoogleNet are 76%, 68% and 68% separately. We can see that C3D gets the best performance among these three architectures for video recognition. This suggests that including motion information helps improve video accuracy rate.

In future work, we could eliminate not necessary frames in the video and only conserve significant frames. Classifiers can learn correct features of video. Besides, we try to use features in 3rd person video to help classifiers to learn features for 1st person video. In figure 14 left top pictures, classifiers learned limited information from the frames but it learns the person with white short with hammer from 3rd person video. So we can use Siamese network to learn features in 1st and 3rd person video and view it as the same features.

VII References:

[1] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. (2014, November). Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the ACM International Conference on Multimedia (pp. 675-678). ACM.
[2] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. 2012. Imagenet classification with deep convolutional neural networks. In NIPS.

[3] Ng, J. Y. H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., & Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. arXiv preprint arXiv:1503.08909.
[4] Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In Advances in Neural Information Processing Systems (pp. 568-576).
[5] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2014). Going deeper with convolutions. arXiv preprint arXiv:1409.4842.
[6] Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2014). C3D: generic features for video analysis. arXiv preprint arXiv:1412.0767.