Item Removal Detection in Retail Environments with Neural Networks

Lingjie Kong[1]
Stanford University
Department of Mechanical Engineering
ljkong@stanford.edu

Xingchen Fan[1]
Stanford University
Department of Mechanical Engineering
xcfan@stanford.edu

Jake Lussier[2]
Stanford University
Department of Computer Science
Jake.t.lussier@gmail.com

Abstract

Inspired by the recent success of deep neural networks in image classification, localization and segmentation, we propose a deep neural network application for video item removal detection in retail environments. In contrast to Amazon Go which accomplishes similar task with both weight measurement and computer vision, we focus on using only computer vision with deep learning to enable customers to explore and shop more efficiently. The input into our network is a stream of video while the output is a prediction of whether item is added to or removed from the shelves. Specifically, we will implement two video classification algorithms: late fusion and 3D convolutional network (C3D). We will evaluate the effectiveness of both algorithms, compare their performance, and explore techniques to enhance the prediction accuracy.

1. Introduction

Neural network models have been successfully applied to recognize human actions from images to videos. This paper explores how deep neural networks with computer vision can be used for action recognition in a very specific setting, namely item removal detection in retail environments. The most relevant technology in the market today is Amazon Go, where computer vision is combined with weight measurements from scales embedded in shelves to detect item removal in grocery stores. Our approach differs from Amazon Go such that we only use visual information to classify item removal and addition based on video information without sensor fusion.

The input to our deep neural network is video of people interacting with items on shelves in front of a vending machine. The camera is mounted at the top of the machine and triggered to record video only when the door is open. An example of our raw video frame is shown in Figure 1.

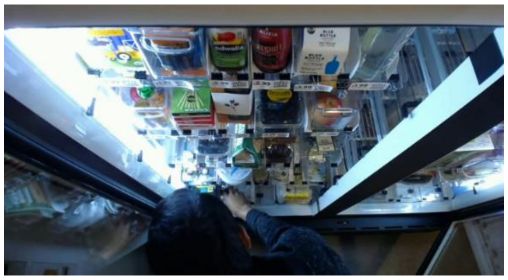
Figure 1: Raw video frame example

We will use several different deep neural network architectures to classify whether items have been removed or added to the shelf within the time frame of the video. Since our own video dataset is small, we will use pretrained models available online and implement transfer learning to avoid overfitting. We will first explore how to apply transfer learning to a state-of-the-art image classification model, SqueezeNet, for our video classification problem. We will then use video classification models like C3D with transfer learning to identify customer actions in the videos. Different approaches will be investigated and compared in terms of classification accuracy as well as computational efficiency.

2. Background

Convolutional Neural Network (CNN) [1] has outperformed most other algorithms in understanding image contents and shows the state-of-the-art performance in image classification, localization, segmentation and detection [2] [3] [4] [5] [6] [7]. The main reason is that CNN is extremely powerful in extracting useful image features for specific tasks [8].

CNN has been used to achieve high accuracy in most image classification competitions, like AlexNet which won the ImageNet challenge in 2012 [9]. Based on the architecture of AlexNet, other deep neural networks have been invented to fully extend the capability of CNN. In 2013, ZFNet was created to improve the accuracy on ImageNet challenge [10]. In 2014, GoogleNet created by Google which introduced the Inception module to improve both accuracy and computational efficiency [11]. The state-

[1]Both authors contributed equally to this work.

[2]Jake Lussier is a non-CS231N contributor who helped collect data.

of-art ResNet which uses network layers to fit a residual mapping instead of direct tuning won the ImageNet challenge in 2015 [12]. Other networks such as VGGNet and SqueezeNet use fewer parameters and allow neural networks to grow deeper [13] [14].

In contrast to so many active researches on image classification, currently there is no single video classification benchmark dataset. Firstly, compared to images, videos are significantly more difficult to annotate. It takes longer to collect a large enough dataset to train CNN. Secondly, videos contain more information compared to images: in addition to spatial information in individual frames, videos also contain temporal information across frames. Therefore, solving a video classification problem is not only technically more challenging, but also more time consuming for training and parameter tuning. Several approaches have been developed to solve video classification problems.

One method is to use pretrained image classification models to extract features from each frame and assemble image information through various fusion strategies like late fusion and slow fusion [15]. That paper also combines a low-resolution context stream and a high-resolution fovea stream to increase computational efficiency without sacrifice in accuracy.

Another approach is three-dimensional convolutional networks (3D ConvNet, or C3D) [16]. Compared to image-based CNNs which apply a series of 2D convolutions, C3D simply stacks videos frames together into a 3D tensor and apply 3D convolutional filter in hidden layers and several fully connected layers in the end.

The third approach is two-stream convolutional network, which explicitly incorporates temporal information into the network [17]. Unlike C3D which operates on stacked frames extracted from the video, two-stream convolutional network still runs a conventional 2D convolutional network to extract spatial information and a separate optical flow-based network to extract temporal information. The results from both networks will be fused together at the end and fed into fully connected layers for classification.

Another video classification architecture is long-term recurrent convolutional network (LRCN). Inspired by recurrent neural networks (RNNs) that are widely used in natural language processing (NLP) [18], LRCNs are developed for visual recognition and description [19]. They use long short-term memory (LSTM) structure. At each time step, it feeds in both the hidden state from the last time step as well as a new frame from the video and uses the RNN architecture to predict the class.

3. Approach

In this project, we focus on two video classification approaches: late fusion and C3D. We implement transfer learning based on available image and video classification TensorFlow models. We will discuss our dataset, preprocessing, late fusion and C3D in details.

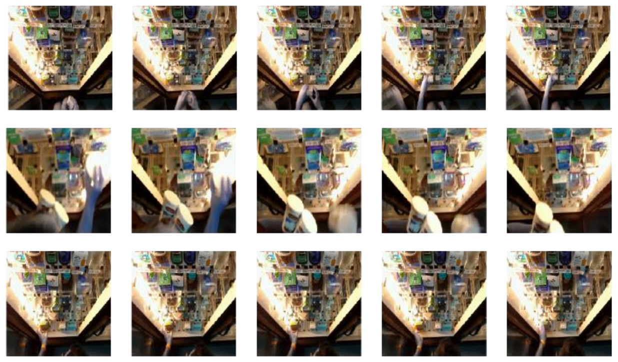
Figure 2: Data samples (From top to bottom: add 1, remove 1 and add 0)

We worked with Jake Lussier from The Thrun Lab to collect around 450 videos of people's shopping behavior in front shelves. We label each video with start frame and end frame as well as whether item has been taken or removed or null action. For simplicity, we neglect the item type and only has four labels to classify as below: add 0 item, remove 0 item, add 1 item, and remove 1 item. Three frame samples corresponding to three classes are shown in Figure 2.

3.1 Data preprocessing for late fusion

Each video will have an average of 10 to 20 frames to capture a single shopping action. In order to analyze how the number of frames influences training, we decided to randomly sample 1, 3 and 5 frames. In addition, we randomly sample five sets of frames for each video in order to augment our dataset. Eventually, we have 2250 video in total.

We randomly selected 50 sets of frames for testing, 50 for validation, and the rest for training. However, since the test and validation datasets might contain frames sampled from the same video as the training set, we removed all the training frames that are from the same video as the test or validation dataset.

Eventually, we have around 1780 for training, 50 for test and 50 for validation with either 1, 3 or 5 frames. We will use these data for our late fusion model.

3.2 Late fusion

Late Fusion was first introduced in [15] for large-scale video classification using convolutional neural networks. One advantage of late fusion is that it can use any image classification model for transfer learning. In order to do so, we pass each frame individually as an image. Rather than using the last layer from the old model for image classification, we concatenate outputs from each frame and train several new fully connected layers for our dataset. This will not only allow us to leverage the good architecture of image classification model, but also help initialize our model more efficiently with pretrained weights in the earlier layers and only train the last fully connected layer.

We apply transfer learning to SqueezeNet [20] model pretrained on ImageNet. Compared to other neural network models such as AlexNet which also achieves high accuracy on ImageNet, SqueezeNet uses less than 0.5 MB of model size to achieve the same accuracy. Therefore, it is more convenient for us to train and test with limited amount of computational resource.

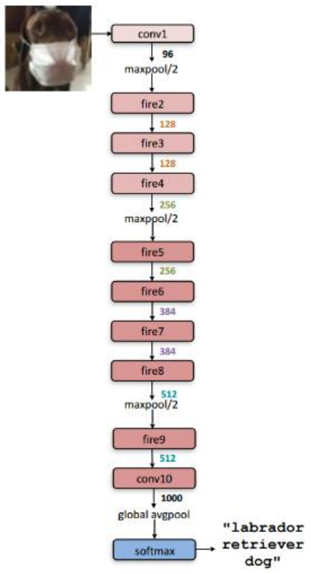

Figure 3: SqueezeNet model [14]

The architecture of SqueezeNet is shown in Figure 3. Each layer of SqueezeNet is built by fire module as shown in Figure 4. Each fire module consists of a set of 1×1 convolutional filters for squeezing and another set of concatenated $1 \times 1, 3 \times 3$ convolutional filters for expanding. It is followed by activation function before connecting to the next sublayer. Since SqueezeNet is designed for ImageNet, it will have 1000 predicted classes. In our case, we replace the last layer with our own fully connected layers for our own video classification problem.

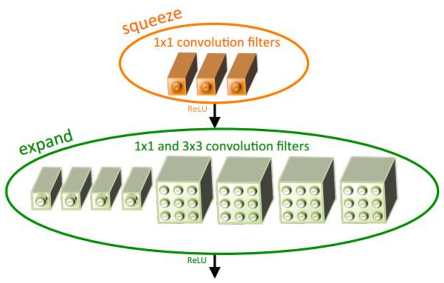

Figure 4: SqueezeNet fire module model [14]

We need to reshape our data smartly in order to use image classification models like SqueezeNet for late fusion. Most image classification models take an input image batch with size $N \times H \times W \times C$ where N is batch size, H is image height, W is image width, and C is channels. However, the input data for video is $N \times F \times H \times W \times C$ where an extra dimension F is the number of frames per video. We first reshape the video batch from $N \times F \times H \times W \times C$ to $(N * F) \times H \times W \times C$ so we can efficiently apply SqueezeNet to every frame in the whole batch. Then the original output layer is replaced with two fully connected layers: the first one downsamples from input size to 100, and the second one downsamples from 100 to 4 labels for prediction. Before the last two layers, the data is reshaped from $(N * F) \times H' \times W' \times C'$ back to $N \times (F * H' * W' * C')$, so all frames for a video clip are concatenated together.

We will train the model with 100 random combinations of hyperparameters and pick one with the highest validation accuracy. Then, we will fine tune the selected hyperparameters to obtain test accuracy on the test set.

Furthermore, we will use late fusion with 1 frame per video to achieve an accuracy of at least 25% which is the same as random guess. We suspect that 1-frame-per-video does not contain enough information to summarize the video content. Later on, we will experiment with 3-frame-per-video and 5-frame-per-video models and hope to improve the accuracy since more information can be extracted across several frames.

3.3 Data preprocessing for C3D

Similar preprocessing procedures are applied before C3D training: every event video is randomly sampled five times, each with a unique set of 5 frames. Since we used a pretrained C3D model on GitHub [21], we had to process and organize the data the same way the model was originally trained. The sample frames are then center cropped and resized to 128×128. During training, the frames are randomly cropped to size 112×112, and subtracted by the mean images from the original training. The final split dataset consists of about 70% for training, 15% for validation and 15% for test.

3.4 C3D

C3D incorporates temporal information by adding a third dimension to the two-dimensional frame data. Frames from a single video clip are directly concatenated to create a $F \times H \times W \times C$ tensor. Both convolution and pooling layers use 3D filters. As shown in Figure 5, 3D convolution is very similar to 2D convolution except that the filter has to slide along the time dimension in addition to the two spatial dimensions.

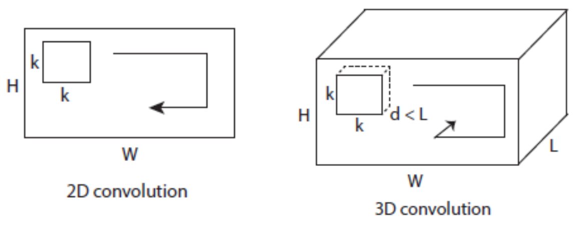

Figure 5: 3D convolution compared to 2D convolution [16]

Tran et al. has shown that the best filter size for 3D convolution is $3 \times 3 \times 3$ [16]. They also propose a model architecture as shown in Figure 6, which achieves more than 80% accuracy for action recognition on UCF-101 dataset and outperforms many other algorithms in various video-based tasks. The specific TensorFlow C3D model used in this project was pretrained on Sports-1M dataset and fine-tuned on UCF-101 dataset, and achieves a top-1 accuracy of 72.6% on UCF-101 [21].

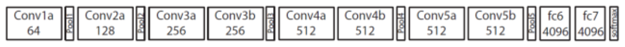

Figure 6: C3D video classification architecture [16]

The C3D model pretrained on UCF-101 has a fully-connected output layer with 101 classes. In our case, we only have four classes. Therefore, we modified the output layer and fine-tuned it on our dataset.

4. Experiment

4.1 Late fusion experiment and results

Since 1-frame, 3-frame and 5-frame late fusion models have different numbers of parameters in the last two layers and different regularization strengths, it's not useful to compare their loss history. Instead, we will compare their training accuracy. An increasing training accuracy will suggest a working model.

We used a minibatch of size 6. For the first 10 epochs, we only train the last two fully connected layers. Then, for the next 10 epochs, we train all layers. Figure 7 shows that the training accuracies slightly increase during the first 10 epochs. Then, they all increase significantly during the next 10 epochs. Therefore, training of all layers is necessary because the SqueezeNet was pretrained on ImageNet, which is very different from our dataset. We need to extract features that are specific to our dataset.

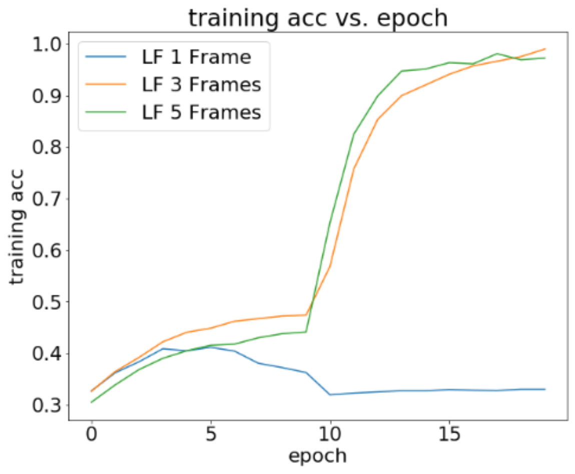

Figure 7: Training accuracy vs. epochs

Table 1 summarizes the performance of different late fusion models. Training accuracy is the accuracy achieved in the last epoch during training. Validation accuracy is the highest one achieved during hyperparameter tuning. We then use the same hyperparameter to get the testing accuracy. There is slight overfit for all three models. However, the more frames we have, the less our model will overfit and the higher the accuracy.

Table 1: Late fusion accuracies

	1 Frame	3 Frames	5 Frames
Training	0.33	0.99	0.97
Validation	0.28	0.80	0.94
Test	0.26	0.88	0.98

The confusion matrices for different late fusion models are shown in Figure 8, 9 and 10 to compare which classes are more likely to be confused with each other by the model. 1-frame late fusion does not have enough information for video prediction and predicts all cases to be one label. 3-frame model yields a better result while get confused across classes. 5-frame model achieves the highest accuracy of 98%. This demonstrates the benefits of having more frames for our video classification problem. However, it takes longer to train with more frames due to more parameters. Therefore, we need to select the best frame rate for video classification based on desired accuracy and computational resource.

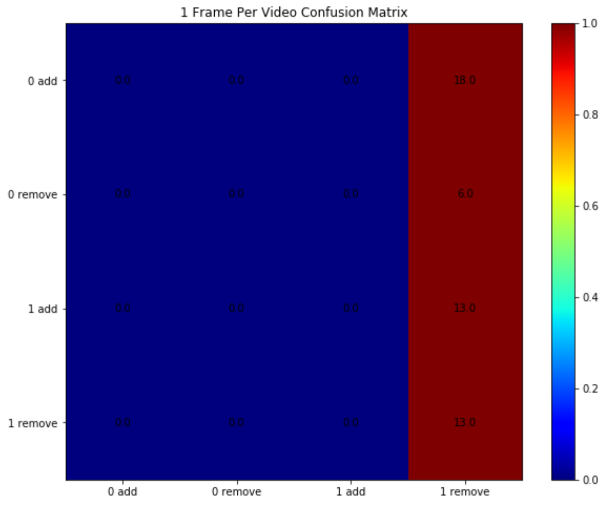

Figure 8: Confusion matrix for 1-frame late fusion

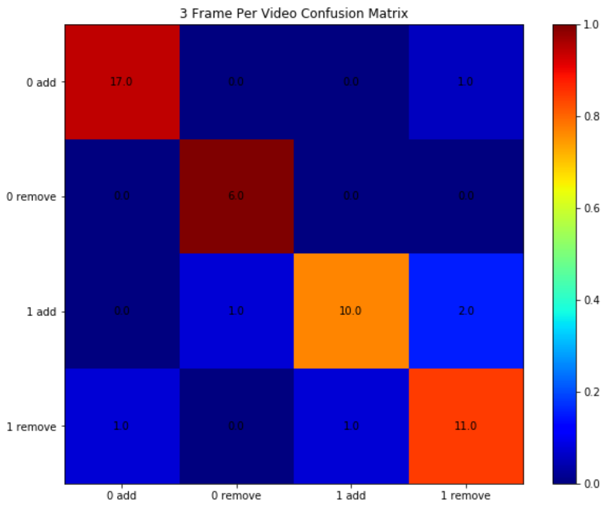

Figure 9: Confusion matrix for 3-frame late fusion

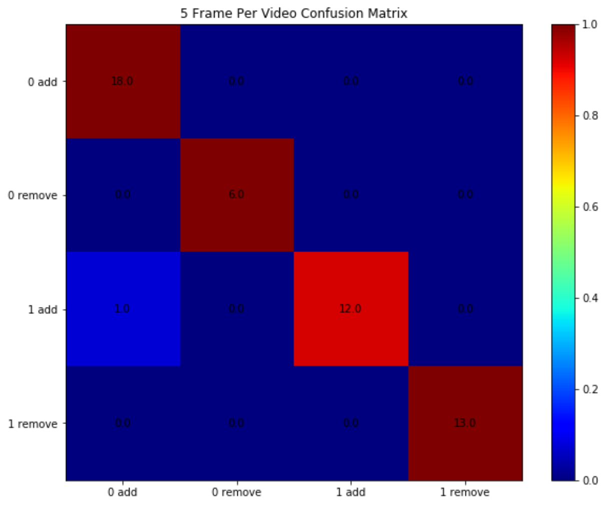

Figure 10: Confusion matrix for 5-frame late fusion

In order to further understand what the model is looking for, we evaluate some saliency maps as shown in Figure 11. Even though the accuracy increases with the number of frames, the saliency map does not highlight the regions around the hand or the arm as we would expect from a working model. Instead, the whole image is used to make predictions. One interesting observation is that the saliency maps become brighter and brighter, suggesting that the model does capture some reasonable temporal information for classification: the later frames are more important for prediction compared to the earlier ones.

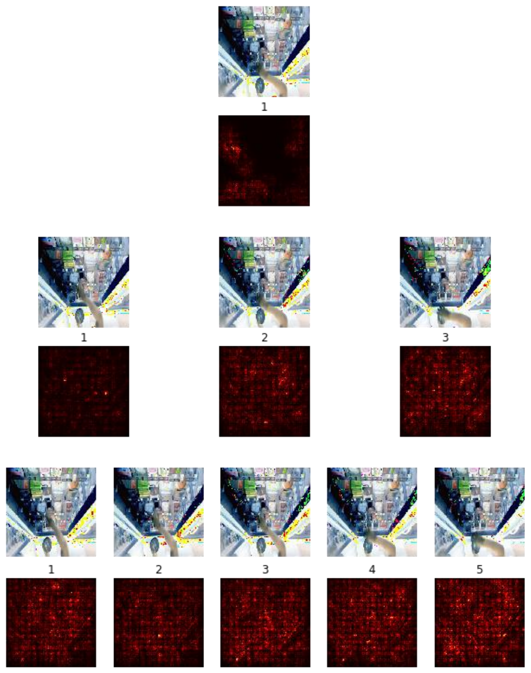

Figure 11: Saliency map examples for late fusion (Top to bottom: 1-frame, 3-frame and 5-frame)

4.2 C3D experiment and results

We first load the pretrained C3D model up to the second fully connected layer. Similar to the approach in late fusion, we first train the output layer on our own dataset for 100 iterations. In the next 500 iterations, we train the full model in order to allow the model to extract features specific to our dataset. We used a minibatch of 30 video clips for each iteration, a training rate of 0.01 for training the output layer, and another training rate of 0.003 for training the entire model. A regularization strength of 0.0005 is applied to every parameter to avoid overfitting.

The history of training accuracy of an example run is shown in Figure 12. Similar to the situation of late fusion, the training accuracy barely increases when only the output layer is trained but increases significantly when the entire model is trained. Due to our relatively small dataset, the training accuracy can easily reach 100%. Table 2 shows that both validation and test accuracies match the training accuracy so there is not much overfitting.

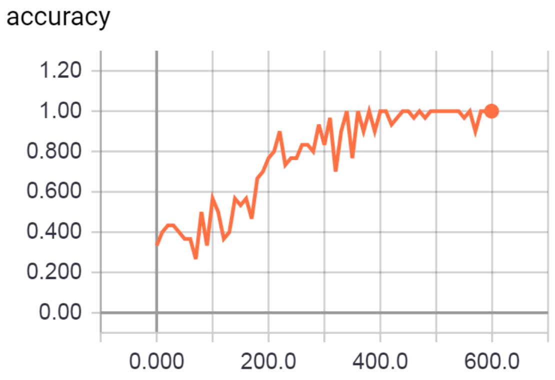

Figure 12: Training accuracy vs. iterations

The confusion matrix shown in Figure 13 demonstrates the good performance of the C3D model. The three mistakes the model makes also make sense: it confuses adding nothing with adding one item, and removing nothing with removing one item. This is because the model is mainly tracking the hand as shown in the saliency maps in Figure 14. It will be a little more challenging to detect whether an object is held in the hand. Therefore, sometimes the model could misclassify whether there is an item involved in an action or not.

Table 2: C3D accuracies

	5 Frames
Training	1.0
Validation	1.0
Test	0.98

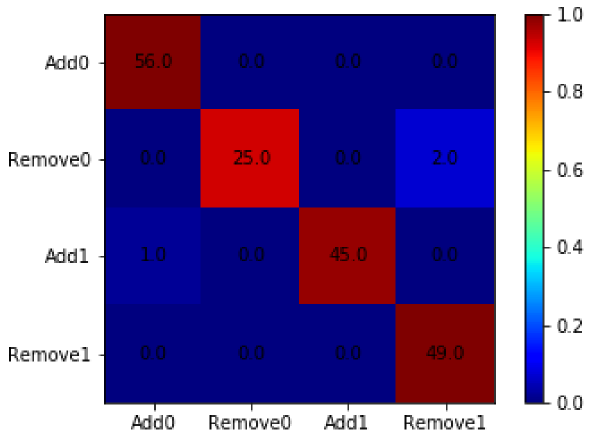

Figure 13: Confusion matrix for 5-frame C3D

The saliency maps shown in Figure 14 show what the model is looking for during prediction. The hand and arm are always the focus of the model. In the top example, the hand is moving away from a top shelf, so the classification result is more sensitive to the pixels near where the hand is. In the bottom example, the hand is moving away from a bottom shelf, so the bottom portion of the frames lights up in the saliency map. The last thing to point out is that the middle frames have relatively smaller salient region, suggesting that the information from the middle frames is not as useful as those from the start and end frames. Therefore, we could potentially use fewer frames with C3D for this problem. However, this is only true if the sampled frames are representative of the video content.

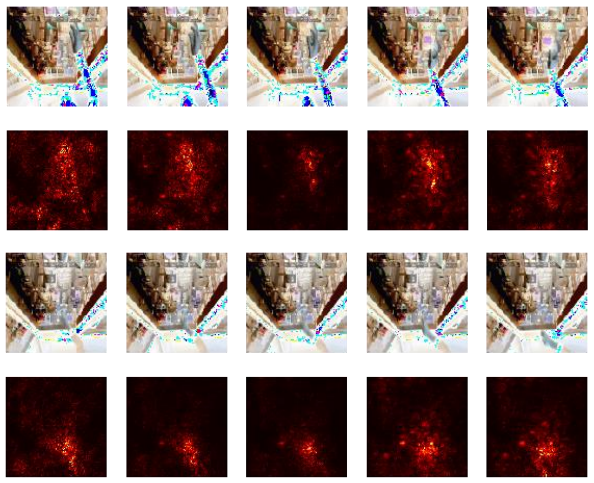

Figure 14: Saliency map examples for C3D (Top: customer reaching for a top shelf. Bottom: customer reaching for a bottom shelf)

5. Conclusion

We have successfully implemented two video classification methods, late fusion and C3D, to our item removal detection problem. Both of them have shown great performance and potential for being implemented in real world.

It seems our models can easily solve the problem presented in this project. Hence in the future, in addition to detect whether the customer has added or removed an item, we could increase complexity and try to classify how many items are involved and what the items are, which will in the end enable the stores to organize inventory more efficiently and allow the customers to shop more easily.

In addition to late fusion and C3D, we would like to experiment with more video classification architectures like two-stream network and long-term recurrent convolutional network. Furthermore, we will explore whether certain handcrafted features or data preprocessing techniques like principal component analysis (PCA) and histogram of oriented gradients (HOG) could improve classification in terms of both accuracy and efficiency. All of these methods and techniques could be useful when the classification problem becomes more complicated as suggested above and if we intend to implement the algorithms in real time.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[2] A. Krizhevsky, I. Sutskever, and H. Geoffrey E., "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst. 25*, pp. 1–9, 2012.

[3] C. Farabet, C. Couprie, L. Najman, and Y. Lecun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, 2013.

[4] D. Ciresan, A. Giusti, L. Gambardella, and J. Schmidhuber, "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images," *Nips*, pp. 1–9, 2012.

[5] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," *arXiv Prepr. arXiv*, p. 1312.6229, 2013.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[7] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 512–519.

[8] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks arXiv:1311.2901v3 [cs.CV] 28 Nov 2013," *Comput. Vision–ECCV 2014*, vol. 8689, pp. 818–833, 2014.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2012.

[10] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8689 LNCS, no. PART 1, pp. 818–833.

[11] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. 07–12–June, pp. 1–9.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[13] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *Int. Conf. Learn. Represent.*, pp. 1–14, 2015.

[14] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," pp. 1–13, 2016.

[15] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[16] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2016, vol. 11–18–Dece, pp. 4489–4497.

[17] K. Simonyan and A. Zisserman, "Two-Stream Convolutional Networks for Action Recognition in Videos," *arXiv Prepr. arXiv1406.2199*, pp. 1–11, 2014.

[18] A. Karpathy and F. F. Li, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. 07–12–June, pp. 3128–3137.

[19] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. 07–12–June, pp. 2625–2634.

[20] S. CS231N, "http://cs231n.github.io/assignments2017/assignment3/." .

[21] H. Xin, "https://github.com/hx173149/C3D-tensorflow." .