# **Item Removal Detection in Retail Environments with Neural Networks**

|  |  |
| --- | --- |
| Lingjie Kong  Stanford University  Department of Mechanical Engineering  ljkong@stanford.edu | Xingchen Fan  Stanford University  Department of Mechanical Engineering  xcfan@stanford.edu |

**Abstract**

*Inspired by the recent success of deep neural networks in image classification, localization and segmentation, we propose a deep neural network application for 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.*

# **Introduction**

Neural network models have been successfully applied to recognize human actions from images and 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 related 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.

With most research focusing on image classification, we are exploring how to use transfer learning combined with the image classification model and applying it to video classification problem.

The input to our deep neural network model 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 : Raw video frame example

We will then 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 dataset of videos is small, we will use pretrained models available online and implement transfer learning to avoid overfitting. Different approaches will be investigated and compared in terms of classification accuracy as well as computational efficiency. We started with an image-based late fusion approach and gradually increase the complexity of our models. We eventually explore video-based approach such as C3D, so we can incorporate more temporal information into our model.

# **Background**

So far, 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 CNNs are extremely power in extracting useful image features for specific tasks [8].

Since 2012, CNN has been used to achieve high accuracy in most image competition such AlexNet on ImageNet competition and so on [9]. Based on the model of image net, other network has been invented to fully extend the ability of CNN. In 2013, ZFNet was created to fully enhance the accuracy on ImageNet competition [10]. In 2014, GoogleNet was created by google which also introduce the Inception module to for computational efficiency [11]. So far, the state-of-art is ResNet which use network layers to fit a residual mapping instead of directly truing to fit a desired underlying mapping [12]. Other networks such as VGGNet and SqueezeNet which use less parameters and enable deeper neural networks [13] [14].

There are so many active researches on image classification. However, currently there is no single video classification benchmark dataset. Firstly, compared to images, videos are significantly more difficult to annotate. It takes large amount of time to collect a large enough dataset to train CNNs. Secondly, videos contain more information compared to images: in addition to spatial and appearance information in each isolated frame, videos also contain temporal information. Therefore, solving a video classification problem is not only technically more challenging, but also more time consuming for training and parameter tuning.

Fortunately, several approaches have been developed to solve video classification problem.

The first possible method is late fusion which is to use pretrained image classification models to extract features from each frame and assemble image information with 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.

The second possible approach is C3D. C3D stands for deep three-dimensional convolutional networks (3D ConvNet) [16]. Compared to image-based CNNs which apply a series of 2D convolutions, C3D simply stacks each frame of video together into a 3D tensor and apply a 3D convolutional filter in all intermediate hidden layers and some fully connected layers at the end.

The third possible approach is two-stream. Two-stream convolutional networks explicitly incorporate temporal relationships into the network [17]. Unlike C3D which feeds stacked images extracted from the video, two-stream convolutional networks run 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 and fed into fully connected layers for classification.

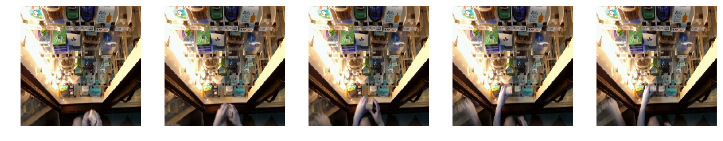
The last possible approach is LRCN. Inspired by RNNs which are widely used in natural language processing (NLP) [18], long-term recurrent convolutional networks are also developed for visual recognition and description [19]. LRCNs use long short-term memory (LSTM) structure. At each time step, it feeds in both the hidden information from last time step as well as a new frame from the video. Eventually, it will use the RNN structure to combine the hidden as well as input information for each frame to predict the final class.

# **Approach**

With so many approach free for us to select, we focus our project mainly on two approaches by using late fusion as well as C3D. We implemented transfer learning based on available image and video classification on Tensorflow. We will discuss our dataset, preprocessing, late fusion, and C3D in details.

# **3.1 Dataset and Preprocessing for Late Fusion**

We work with Jake Lussier from Sebastian Thrun’s lab to collect around 450 videos of people’s retail shopping behavior. We label each video with start frame and end frame as well as what item has been taken, what item has been removed, and null action. For simplicity, we ignore 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 sample of frames which are clipped from the video are as below.



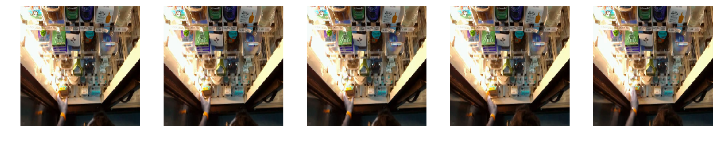
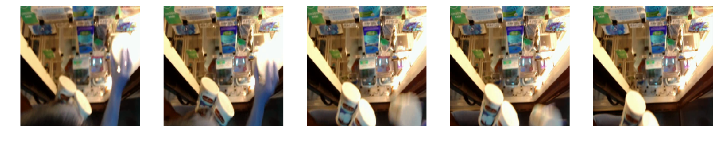


Figure Data Sample (from top to bottom are add 1, remove 1, and add 0)

Overall, each video will have around 10-20 frames information to summarize people’s shopping actions. Under the condition that we want to analyze how will more frames help increase the accuracy, we decide to random sample 1 frames, 3 frames, and 5 frames for five times per video. Eventually, we have 2250 video in total.

We random sample 50 samples for frames for test, 50 frames for validation, and the rest for train. However, under the condition that the test and validation dataset might be sampled from the same video for the train, we remove all the train dataset which are from the same video of the test and validation.

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

# **3.2 Late Fusion**

Late Fusion was first introduced in this paper [15] for largely scale the video classification by using convolutional neural networks. One advantage of late fusion is that it can image classification model for transfer learning. In order to do so, we can pass each frame individually as an image. Rather than using the last layer from the old model for image classification, we concatenate all parameters from each frame and train a new fully connected layer for our model. This will not only allow us the inherit the good architecture for image model, but also allow us to warm start our model by using the old weight from image model for all previous layers and only train the last fully connected layer. Let’s take a detail look on how we set up the model and train it.

We use transfer learning based on SqueezeNet model trained on ImageNet. Compared to other neural network model which also achieve high accurate on ImageNet benchmark such as AlexNet, SqueezeNet only use less than 0.5MB model size to achieve the same accuracy. Therefore, it is more convenient for us to train and test on our local machine as well as on Google Cloud based on our limited amount of resource.

The SqueezeNet model is as below.

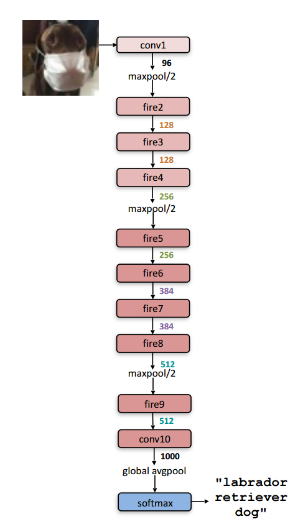


Figure SqueezeNet Model [14]

Each sublayer of SqueezeNet is built by fire module building block as below. Each file module is a convolution filters for squeezing, and another concatenated convolutional filters for expending. It will be passed through activation function and into next sublayer. Because SqueezeNet is designed for image net, it will have 1000 predicted classes. As for our case, we move the last layer and add a fully connected layer for our application.

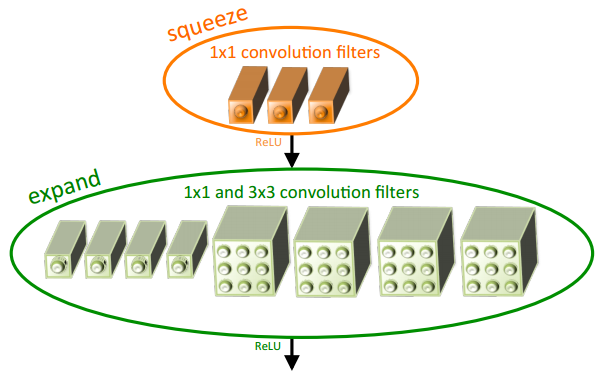


Figure SqueezeNet Fire Module Model [14]

Train the last fusion based on the image model is kind of trick specially to reshape the data to the correct shape, so we can feed it into the image model. For most image classification model, the input image batch is like in which is batch size, is image height, is image width, and is channels. However, the input data for the video is like in which is input video batch and is input frames per video.

First, we reshape the video batch with shape to , therefore, we should be able to pass them into our SqueezeNet Model. Now, the original last layer has been remove. Instead, we add a two more fully connected layer first down sample from necessary size to 100, and one more to down sample from 1000 to 4 labels for prediction. Before entering the last two layers, our data should look like , we apply our late fusion by reshape the data to the shape of . By doing so, all frames information for each video will be concatenated together for the last two layers for prediction.

Second, during the training, we first run 10 epochs by only training the last two layers while keep all the previous convolutional layers the same as the original SqueezeNet. Then, we train another 10 epochs which fine tune the all layers. The batch size is 6. We set up random search to run for 100 times and pick the combination of hyperparameters which have the highest validation accuracy. Then, we will fine tune the selected combination of hyperparameters and use the test set to check the accuracy.

Third, we first use 1 frames per video to check our model and hope to have an accuracy around 25% which is just random guess. We believe that 1 frames per video do not have sufficient amount of information to summarize the video content. Later on, we try 3 frames per video and 5 frames per video and hope these models will significant enhance the accuracy based on the temporal and spatial information summarize from one frame to another.

**3.3 Preprocessing for C3D**

**3.4 C3D**

# **Experiment**

**4.1 Late Fusion Experiment and Result**

Under the condition that the 3 different models with 1 frames, 3 frames, and 5 frames has different number of parameters for the last two layers as well as best regularization parameter. Comparing the training loss might be meaningless. Instead, the compared the training accuracy to see whether our model has increasing accuracy during each epoch during the training. It is obviously the training accuracy is going up which proves our model is correct.

To recap how we set the training, for the first 10 epochs, we only train on the last two fully connected layers. Then, for the next 10 epochs, we train for all layers. From the figure below, we can see that the training accuracy slightly increase for the first 10 epochs. Then, it increases significantly for the next 10 epochs. Therefore, training for all layers is necessary because the parameter for convolutional layers is trained based on ImageNet dataset. As for our dataset, we need to retrain then to capture necessary features that we need for our dataset.

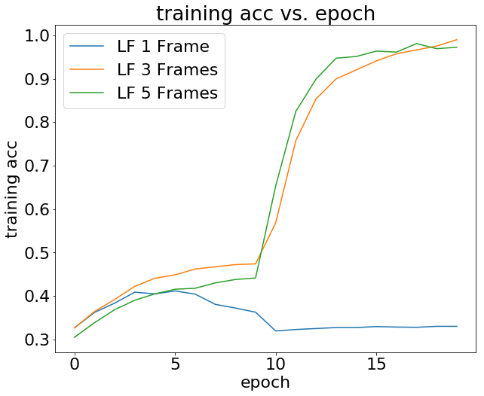


Figure Training Accuracy vs. Epochs

We also want to have a look at our training, validating, and testing accuracy to see whether our training overfit the data.

Table Train, Validation, Test Accuracy

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 Frames | 3 Frame | 5 Frames |
| Train | 0.33 | 0.99 | 0.972 |
| Validation | 0.28 | 0.78 | 0.94 |
| Test | 0.26 | 0.92 | 0.98 |

Training accuracy is the last accuracy for training. Validation accuracy is the highest one during random search for hyperparameters. Test accuracy is the corresponding one to the highest validation accuracy.

As one can see, there is slightly overfit for all three cases. However, the more frames we have, the less overfit our model will be and the more accurate our model is.

We also plot confusion matrix for 1 frames, 3 frames, and 5 frames to compare which label will be more liked to be confused with another.

Obvious, 1 frame does not have enough information for video prediction, it predicts all cases to be one label. 3 frames give a better result while get confused between 0 remove and 1 add as well as 0 add and 1 remove. 5 frames give the highest accuracy of 98%. This again proves our initial assumption is correct that the more frames we use for video classification, the more accurate it will be. However, it usually takes longer time to train due to more parameters. Therefore, we need to optimize based on the expected accuracy as well as hardware resource to decide the best frames rate for video classification by using late fusion.

Figure Training Accuracy for Late Fusion

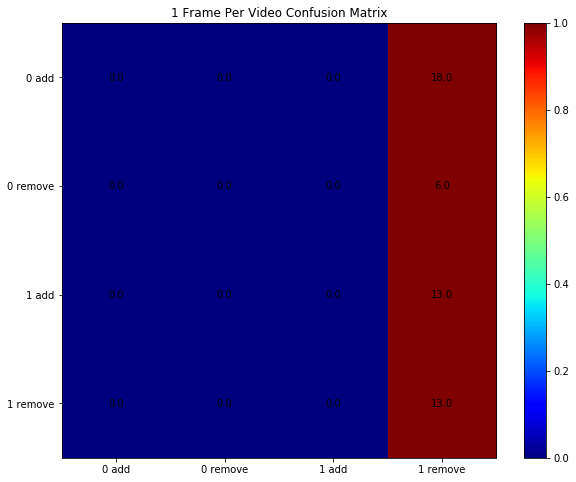


Figure Confusion Matrix for 1 Frame per Video

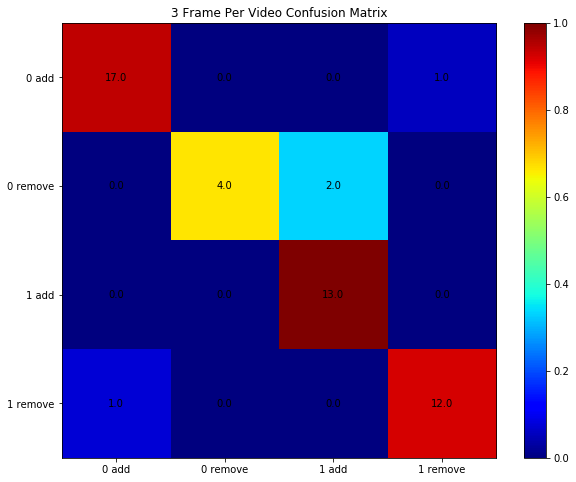


Figure Confusion Matrix for 3 Frame per Video

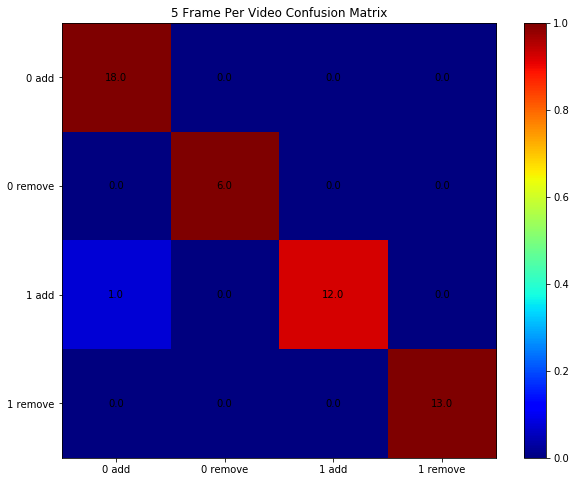


Figure Confusion Matrix for 5 Frame per Video

**4.2 C3D Experiment and Result**

# **Conclusion**

We also plan to implement two-stream networks with pretrained models. We will form two stream convolutional networks for spatial stream from single frame and temporal stream from multi frame optical flow. Eventually, two frame results will be either concatenated together through fully connected layer and apply stochastic gradient decent to minimize the Softmax loss.

If time permits, we also wish to implement long-term recurrent convolutional networks. Each RNN layer will use an LSTM structure to prevent gradient vanishing. Hidden layer parameters will be from pre-trained model and only the fully connected layer parameters will be tuned on our dataset.

Furthermore, we will explore whether certain handcrafted features or data augmentation procedures could improve classification. Ideally, convolutional layers will learn to extract features from frames at various scales automatically throughout training. However, some preprocessing of data could speed up training and reduce the complexity of the network. Common techniques to implement include principal component analysis (PCA) and histogram of oriented gradients (HOG).

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.