# **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 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.

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.

# **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 CNNs are 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 introduced the Inception module to improve both accuracy and computational efficiency [11]. The state-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 CNNs. 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 in 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.

# **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.

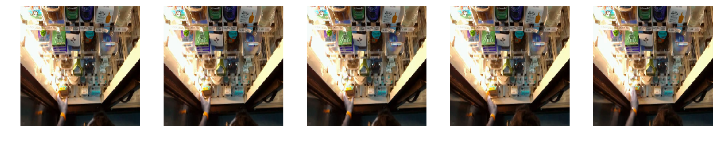
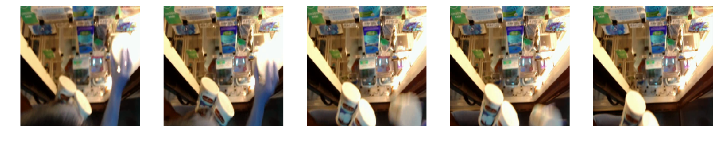
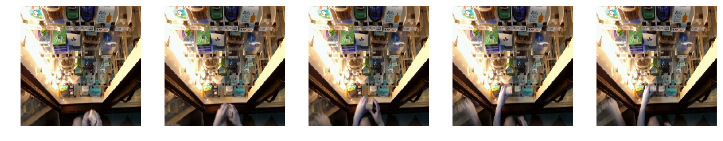


Figure 2: Data samples (From top to bottom: add 1, remove 1, and add 0)

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 will influence 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 models 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 a new fully connected layer for our model. This will not only allow us to leverage the good architecture of image classification model, but also allow us to initialize our model more efficiently with pretrained old weights in the earlier layers and only train the last fully connected layer.

We apply transfer learning to SqueezeNet model pretrained on ImageNet. Compared to other neural network models which also achieve high accuracy on ImageNet such as AlexNet, 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 our limited amount of resources.

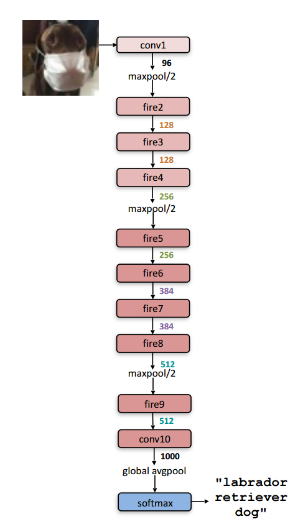


Figure 3: SqueezeNet model [14]

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 convolutional filter for squeezing and another concatenated 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 layer for our own video classification problem.

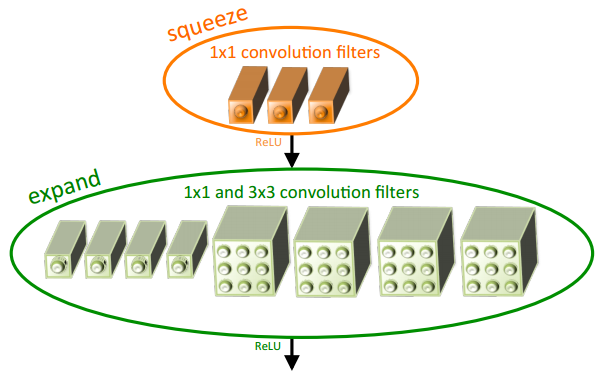


Figure 4: SqueezeNet fire module model [14]

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 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 [20], 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 . During training, the frames are randomly cropped to size , and subtracted by the mean images from the original training. The final split dataset consists of 70% for training, 15% for validation and 15% for test.

# **3.4 C3D**

C3D directly incorporates temporal information by adding a third dimension to the two-dimensional frame data. Both convolution and pooling use 3D filters. D. Tran et al. has shown that the best filter size for 3D convolution is [16]. They also propose a model architecture as shown in Figure 5, 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 the 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 [20].

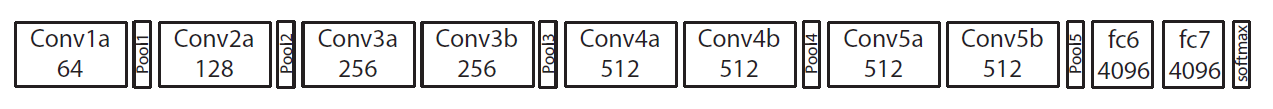


Figure 5: C3D video classification architecture [16]

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 on our dataset.

# **Experiment**

# **4.1 Late fusion experiment and results**

Since 1-frame, 3-frame and 5-frame late fusion models have different number of parameters for the last two layers as well as best regularization parameter, it’s not useful to compare their loss histories. Instead, we will compare the training accuracy. An increasing training accuracy will suggest a working model.

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 6 shows that the training accuracies slightly increase during the first 10 epochs. Then, it increases significantly during the next 10 epochs. Therefore, training of all layers is necessary because the parameter for convolutional layers was pretrained on ImageNet, which is very different from our dataset. We need to extract features that are specific to our dataset.

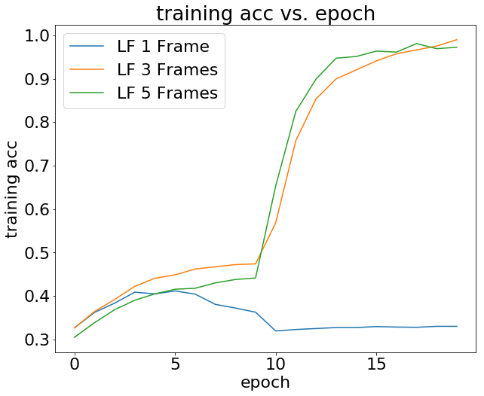


Figure 6: Training accuracy vs. epochs

Table 1: Late fusion accuracy

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 Frame | 3 Frames | 5 Frames |
| Training | 0.33 | 0.99 | 0.97 |
| Validation | 0.28 | 0.78 | 0.94 |
| Test | 0.26 | 0.92 | 0.98 |

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 corresponding test accuracy. There is slightly overfit for all three models. However, the more frames we have, the less our model will overfit and the higher the accuracy.

The confusion matrix for different late fusion models are shown in Figure 7, 8 and 9 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 between 0 remove and 1 add as well as 0 add and 1 remove. 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 time to train with more frames due to more parameters. Therefore, we need to select the best frame rate for video classification by late fusion based on desired accuracy and computational resource.

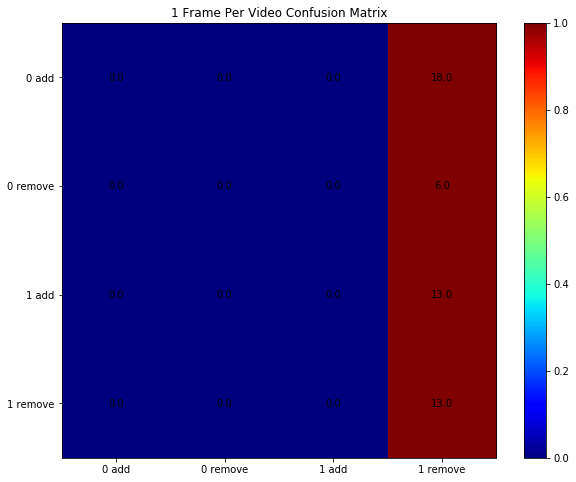


Figure 7: Confusion matrix for 1-frame late fusion

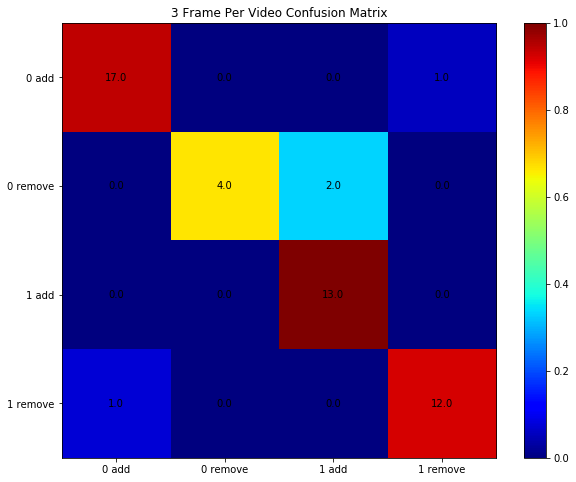


Figure 8: Confusion matrix for 3-frame late fusion

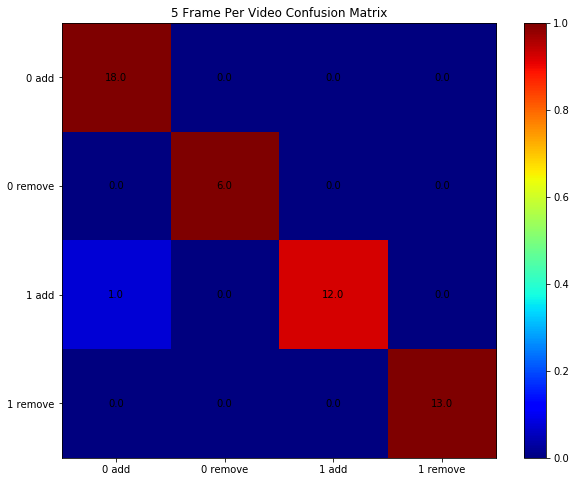


Figure 9: Confusion matrix for 5-frame late fusion

# **4.2 C3D experiment and results**

We first load the pretrained C3D model up to the second fully connected layer. In the first 100 iterations, we only train the output layer on our own dataset. 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 11. 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.

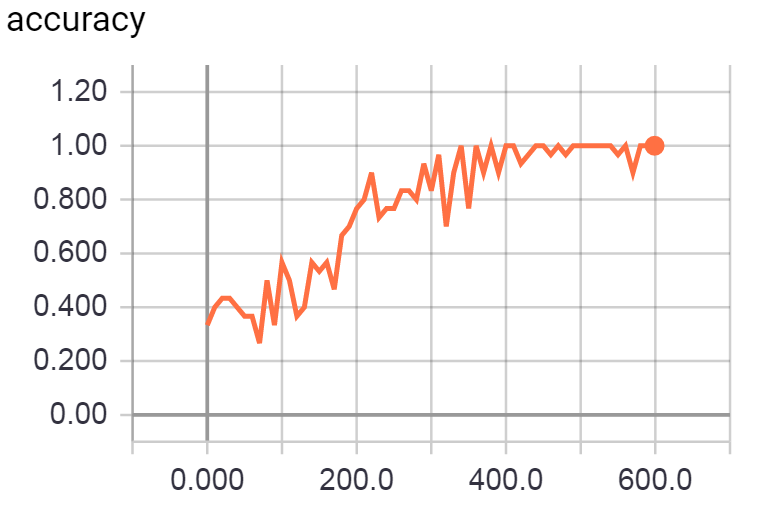


Figure 10: Training accuracy vs. iterations

The confusion matrix shown in Figure 12 demonstrates a 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 and removing 1 item. This is mainly because the model is mainly tracking the hand as shown in the saliency maps in Figure 12. It will be a lot more difficult to detect whether there is an object held in the hand. Therefore, sometimes the model could misclassify whether there is an item involved in an action or not.

Table 2: C3D accuracy

|  |  |
| --- | --- |
|  | 5 Frames |
| Training | 1.0 |
| Validation | 1.0 |
| Test | 0.98 |

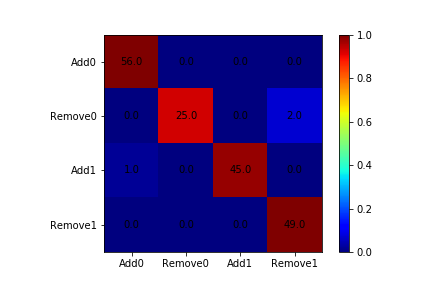


Figure 11: Confusion matrix for 5-frame C3D

The saliency maps shown in Figure 12 demonstrates 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 middle bottom portion of the frames lights up in the saliency map. The last point to notice is that the middle frames have relatively smaller salient region for the model, suggesting that the information from the middle frames is not as useful as those from the start and end frames. We could potentially use fewer frames with C3D for this problem. However, this is only true if the sampled frames really represent the content of the video.

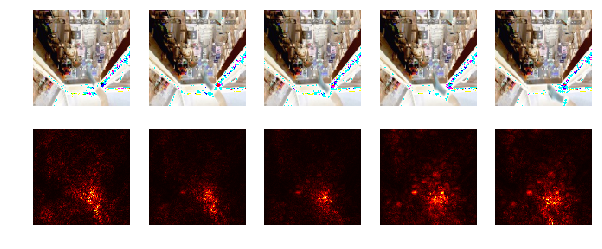
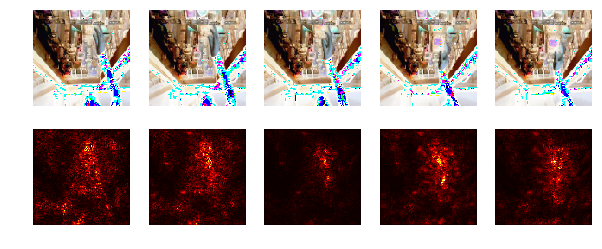


Figure 12: Saliency map examples (Top: customer reaching for a top shelf. Bottom: customer reaching for a bottom shelf)

# **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 and allow the customers to shop more easily.

We could explore 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.

[20]