Transfer Learning of Temporal Information for Driver Action Classification

Joseph Lemley
National University of Ireland Galway
College of Engineering and Informatics
Galway, Ireland
j.lemley2@nuigalway.ie

Shabab Bazrafkan
National University of Ireland Galway
College of Engineering and Informatics
Galway, Ireland
S.Bazrafkan1@nuigalway.ie

Peter Corcoran
National University of Ireland Galway
College of Engineering and Informatics
Galway, Ireland
peter.corcoran@nuigalway.ie

ABSTRACT

Correct classification of image data can depend on features learned in multiple sequential frames. We focus on the problem of learning action from video data with an emphasis on driver behavior monitoring. An insufficient quantity of high quality labeled data is a major problem in machine learning research. This is especially true when deep neural networks are used. Although some sufficiently large, general purpose image databases exist for action recognition, most of these are limited to single frames. This kind of data requires that the action recognition task is applied regardless of the temporal information (information from previous and next frames of a video sequence). In this paper, we show that temporal information is useful for accurate classification of video and that the temporal information in lower layers of a convolutional neural network can successfully be transferred from one network to another to greatly improve performance on the driver behavior monitoring task.

KEYWORDS

Deep Learning, Transfer Learning, Action Recognition

1 INTRODUCTION

In recent years, deep learning has become ubiquitous for image classification, with some results exceeding human accuracy for certain tasks. With few exceptions, these impressive results have been limited to a set of tasks for which a single picture provides all the information required to easily distinguish between classes and essentially ignore any time element for the recognition task.

As we approach the limits of frame-based methods, there is a desire to further improve deep learning algorithms by utilizing temporal information, which is information between multiple frames taken sequentially to give a more complete idea of what is happening. Using a single frame, it is trivial to train a classifier to determine if a person is holding a glass, but difficult or impossible to train a classifier to understand if the glass is being picked up or put down. Even distinguishing jogging from walking can be difficult without a time component.

For this more refined level of visual understanding, we need machine learning models that can learn temporal information and information about frame content at the same time. We also need quality labeled data of sufficient size to prevent overfitting. Much work has been done recently to address these issues (described in the related work section below), but there is still a lack of quality data for use in many practical applications. For example, there are no publicly available databases for driver monitoring that are of sufficient size to train a deep neural network from scratch.

MAICS'2017, Fort Wayne Indiana
© 2017 Copyright retained by the authors. .

The largest database for driver behavior monitoring that we could find is the "Distracted Driver Dataset", provided as part of a Kaggle challenge in mid-2016. Although this database is intended for single frame classification, it is possible to identify the original frame sequences from which movies can be created. These movies can then be used for learning a limited amount of temporal information.

In this paper, we address two questions:

"Can temporal information be used to improve accurate classification of driver actions?" and "Can low-level information about temporal information from an unrelated problem be successfully used to better understand driver actions in videos?"

In the next session, we provide brief background information on Recurrent Neural networks and 3D CNN's for the interested reader. In section three, we present related work. In sections 4, 5, and 6, we present the methods, experiments, and results.

2 BACKGROUND

In this section, we provide a brief overview of the key neural network architectures used in this work, including recurrent neural networks, long short term neural networks, convolutional neural networks and 3D convolutional neural networks. We also briefly describe transfer learning. Readers with a knowledge of these topics are encouraged to skip to the Related Work section below.

2.1 RNN (Recurrent Neural Network)

Recurrent neural networks(RNN) [19] are a highly flexible network architecture frequently used to model time series data, such as a sequence of frames from a video. RNNs remember the input at a previous stage and make a decision for the present input based on the sequence of the previous data. The hidden state of an RNN is calculated based on the state of the network at each sequence. The hidden state can be considered the memory. RNNs have had great success in speech and video recognition and are sometimes combined with convolutional layers [11].

2.2 LSTM (Long Short Term Memory)

LSTMs [5] are a type of RNN that are designed with both long and short term memory which permits the modeling of more complex time series. LSTMs include at least 3 gates which control the way information flows.

2.3 CNN(Convolutional Neural Networks)

Convolutional Neural networks (CNNs) were popularized by lecun et al. [9], who used them successfully for handwritten digit classification. These networks are inspired by the organization of the visual cortex and allow spatial information to be more efficiently learned. Convolutional Neural Networks can be used on input of any number of dimensions but due to their success in pictures, are most popularly implemented for 2D input plus color channels. Other popular types of CNN's include 1D CNNs which are commonly used for time series and 3D CNNs which can be used for volumetric data or time series data where the third dimension represents either spatial frames or temporal frames [11].

2.4 Transfer Learning

Transfer learning is the process of transferring knowledge that has already been learned by one neural network into another one. This is often accomplished by copying the learned weights and biases from one or more layers of a fully trained network to a different network. Transfer learning can be used to overcome overfitting issues and to speed up the training process for a related task.

3 RELATED WORK

In 2008, before the recent wave of deep learning, Klaser et al. [8] used 3D HOG (Histogram of Oriented Gradients) descriptors and showed that 3D gradients were able to assist in understanding action from videos. More modern approaches to action classification from videos are primarily based on 3D convolutional neural networks (CNN) or recurrent neural networks (RNN) with some convolutional component or a combination of the two. 3D CNNs have been shown to capture short-term temporal information.

For example, [18] makes a compelling case for the use of 3D CNNs for understanding video data. Their method, which they named C3D, compared favorably to other published results on 5 of 6 generic action datasets used. They also showed that their network learns information about both motion and appearance, first learning appearance and then motion. One problem with this design is that only relatively short action sequences (16 frames) can be learned.

Addressing this problem, Lei, et al. [10] present an interesting approach to action recognition, they combine a 3D convolutional neural network with a hidden Markov model (HMM) and show that their method compares favorably with other methods. This paper adds support to the idea that 3D convolutions are important to understanding short-term temporal information about movement and the need for another mechanic (RNN, HMM, etc) to understand long-term context (more than a few frames)

Keeping on the theme of combining 3D CNN with a network that is able to learn long-term information, Molchanov et al. [12] combined a 3D convolutional neural network with a recurrent neural network to classify short video clips of hand gestures, reporting excellent results.

Donahue et al. introduced a new architecture called LRCN (Long-term Recurrent Convolutional Networks) that combines RNNs with CNNs [2]. Specifically, they show how to train and optimize a long-term RNN model that can account for temporal information in video data. They evaluate their method on the UCF - 101, a database that contains over 12,000 videos with 101 human action classes [16] .

Another new architecture that combines LSTM with convolutions is introduced in "A Machine Learning Approach for Precipitation Nowcasting" [20]. Although the focus of their paper is forecasting precipitation, their method is generally applicable to the task of gathering long and short term time information from video sequences.

In [21], convolutional temporal pooling is used with a long short term memory (LSTM) network to produce state of the art results. In the same year, [2] used a convolutional LSTM to narrate videos.

Because convolutions on 3D data are time intensive, there have been attempts to combine 2D networks while retaining temporal information. For example, in [3], Feichtenhofer et al. train separate CNNs for motion and appearance and report very good results.

Addressing the problem of insufficient data to train a neural network, [7] introduced a large, automatically generated, database gathered from YouTube clips called the sport 1M dataset. They showed that a transfer learning approach is effective at gaining accuracy on UCF 101 when a network is first trained on sports 1M.

Object proposal networks, specifically fast/faster R-CNNs (region-based convolutional neural networks) have been used successfully as in [14],and [22], for action recognition, and seem to work especially well in cases where there is not enough data to train a full network. Hoang, et al. used this method for detecting cell phone usage and hands on steering wheel detection in [4].

Just as there are attempts to classify behavior without exploiting temporal information, there are also ways of addressing the problem of driver behavior monitoring that do not depend on spatial information but instead use only temporal information. For example, [6] compared LSTM, RNN, logistic regression, and deep neural networks, for detecting driver confusion. Rather than using images, that paper uses multimodal sensor data to assess driver confusion. They found that LSTM outperforms the other models, likely because some long-term information is important for this task. Similarly, [13] uses human motion, as given by cell phone sensors, as a biometric, and an RNN was utilized to process sensor signals.

4 METHODS

Experiments were conducted on NVIDIA Titan X GPU's running a pascal architecture with python 2.7, using the Theano [17] and Keras [1].

4.1 Data Preparation

4.1.1 Driver Monitoring Dataset. The Distracted Driver Dataset was provided as part of a Kaggle Competition in 2016. The dataset was created by filming actors on a closed driving course engaging in various distracted and undistracted behaviors. It should be noted that these images were obtained in a controlled environment the car was not actually being driven. It was being pulled by a truck instead. The objective of the competition was to correctly classify still images into 10 categories.

The training set of the distracted driver database contains frames of 26 subjects displaying several of the following behaviors/actions:

(1) c0: safe driving

(2) c1: texting - right
(3) c2: talking on the phone - right
(4) c3: texting - left
(5) c4: talking on the phone - left
(6) c5: operating the radio
(7) c6: drinking
(8) c7: reaching behind
(9) c8: hair and makeup
(10) c9: talking to passenger

Although all the images in the supplied training set are still images, it is possible to reconstruct the original "movies" based on their order in the CSV file supplied with ground truth annotations.

While classifying frames for driver monitoring is an interesting problem, we wanted to see if we could learn anything from the temporal information in the movies. Instead of using individual frames as required for the competition, we created short movie clips, as we describe later. It is not possible to create such movies from the supplied testing set because sequence information was intentionally left out of it by the competition organizers.

First, we arranged the dataset into distinct subject, action segments ordered by time. The generated movies varied between 38 and 135 frames in length (average is about 101) for a total of 100 clips. Because the 3D convolutional neural network we were planning to experiment with requires a fixed frame size, we further processed the videos into fixed sized frames using the sliding time window method. For each of these segments, we create a series of 5, 10, 16, and 30 frame clips. Each clip is made using a sliding window starting with the first (subject, action) with a step equal to approximately 70% of the clip length to allow sufficient frame overlap. Appropriate action labels are applied to each clip.

In all experiments, 20 drivers are used for the training set and the remaining 6 for the validation set. Due to the small size of this database, we did not feel that further dividing the data into a smaller test set would be reasonable.

In our experiments, we used 4 variations of these:

(1) Grayscale video: All videos of the 26 drivers were reduced to 60×80 Grayscale images.
(2) Color video: All videos of the 26 drivers were reduced to 112×112 full color.
(3) Grayscale frames: All frames of the 26 drivers were reduced to 60×80 Grayscale images.
(4) Color Frames: All frames of the 26 drivers were reduced $112 \times' 112$ Grayscale images.

Due to copyright restrictions, we are unable to include examples of these images in this paper, but the interested reader may view them by visiting the relevant competition at:

https://www.kaggle.com/c/state-farm-distracted-driver-detection

4.2 Augmentation

In experiments where augmentation was applied, the ImageDataGenerator class within Keras was used. This class is used to dynamically create augmented images during training given a set of parameters. Since the standard implementation of ImageDataGenerator only supports 2D data, we subclassed it to properly apply the transformations to video data. This modification involved ensuring that the same transformation was applied to every frame of a clip instead of treating each frame as an individual image with a potentially different transformation.

5 EXPERIMENTS

In this section, we describe 10 experiments on the distracted driver database. The experiments were designed to allow comparison between networks that use temporal information (LSTM, 3D CNN, etc) and networks that ignore it (2D CNN). The last experiments are designed to measure the improvement that is achieved by transfer learning.

5.1 2D experiments

We train an implementation of VGG-16 [15] from scratch on the distracted driver dataset in Keras with a learning rate of 0.001. Our loss function was categorical cross entropy and our output class predictions were obtained from a final softmax layer.

We then repeated our experiments on our grayscale dataset with rotation, translation, and feature normalization. By feature normalization, we mean that the inputs are divided by the standard deviation of the dataset.

5.2 Very Small CNNs and LSTMs

In this set of experiments, we utilized several small (one or two layer) configurations of CNN+LSTM, 3DCNN +LSTM, and 3DCNN.

For these experiments, the LSTM used is the one described by [20], implemented in Keras as ConvLSTM2D.

Three networks were evaluated:

Network 1: 3DCNN followed by Softmax (Num_Classes)

One 3D convolution with 16 filters and a kernel size of 3,3,3 followed by a softmax layer.

Network 2: LSTM followed by 3DCNN → Softmax (Num_Classes)

One LSTM with time length 16, and a 3x3x3 convolution kernel followed by one 3D convolutional layer with 16 filters and a kernel size of 3,3,3 ending with a softmax layer.

Network 3: 3DCNN followed by LSTM → Softmax (Num_Classes)

One 3D convolutional layer with 16 filters and a kernel size of 3,3,3 followed by one LSTM with time length 16, and a 3x3x3 convolution ending with a softmax layer.

All three experiments used a learning rate of 0.001 with nestrov momentum as 0.95. Categorical crossentropy was our loss function and stochastic gradient descent as the training method.

These experiments were then repeated with the grayscale video database.

5.3 Experiments with small 3DCNN architectures

Since the distracted driver dataset has a strong tendency to overfit due to the limited number of subjects, we designed a network and training strategy that would delay overfitting for as long as possible and prevent fast convergence. This involved using a high learning rate and large batch size, resulting in a training method that would allow the network to jump over minima. We also observed that the overfitting came from very low level information, so we restricted the first layer to only 8 features. The network architecture was designed as follows:

(1) input layer with 15 frames, a width and height of 112.

(2) 3D Convolutional Layer(RELU). 8 filters with size: 3, 3, 3
(3) 3D Max pooling operator with size: (2,2,2)
(4) 3D Convolutional Layer(RELU). 16 filters with size: 3, 3, 3
(5) 3D Max pooling operator with size: (2,2,2)
(6) 3D Convolutional Layer(RELU). 32 filters with size: 3, 3, 3
(7) 3D Max pooling operator with size: (2,2,2)
(8) 3D Convolutional Layer(RELU). 64 filters with size: 3, 3, 3
(9) 3D Convolutional Layer(RELU). 128 filters with size: 3, 3, 3
(10) 3D Convolutional Layer(RELU). 256 filters with size: 3, 3, 3
(11) Fully connected Layer (RELU) with 4096 units.
(12) Dropout with a probability of 0.5
(13) Fully connected Layer (RELU) with 4096 units.
(14) Dropout with a probability of 0.5
(15) Softmax with 10 units (one for each of the 10 action classes in the Distracted Driver Dataset)

A learning rate of 0.001 with nestrov momentum as 0.95 and categorical crossentropy as the loss function was used. Stochastic gradient descent was used as the optimizer.

Variations with additional dropout and Batch Normalization layers were also evaluated, but only increased the overfitting and decreased validation accuracy.

These experiments were repeated with rotation, translation, and featurewise normalization.

5.4 C3D trained from scratch

In this experiment, we trained a C3D [18] from scratch on the distracted driver dataset. Since C3D was designed to use 16 frame color video clips of size 112 X 112 we used the color video dataset described previously. As with previous experiments, a learning rate of 0.001 with nestrov momentum as 0.95 was used with categorical crosentropy and gradient descent.

5.5 Transfer Learning with C3D

Since other approaches to reducing the overfitting problem were of limited success, we tried a transfer learning approach. The idea is to use pretrained weights from an existing network, trained for a more generic action recognition task, and then to tune them with the Distracted Driver training set.

We used a pretrained 3CD on the sports 1-M dataset, included the architecture, and trained weights for The Sports-1M dataset in a GitHub repository as a 3D CNN. They reported very good results at capturing temporal information using their 3D CNN, so it seemed like a natural place to start.

In this experiment, we investigate the use of transfer learning to overcome the overfitting problem identified in the previous cases. In the previous experiments, the first layers were identified as being the primary source of overfitting, so we wanted to try two transfer learning approaches.

The first transfer learning approach we tried was to train the C3D network with a very low learning rate of 0.0001 without freezing any layers with these pretrained weights.

We then used an alternate transfer learning approach where we trained only on the final softmax layer, freezing the learning rate of the first layers.

Since we previously identified the first layers as being the greatest source of overfitting, we repeated this experiment freezing only the first five layers, and then repeated it again with only the first two layers frozen.

6 RESULTS

In this section, we report the results of the experiments in the previous section. The experiments with the best results are listed in table 1. Since overfitting is found to be the primary cause of validation error in most experiments, we also show details about the loss and accuracy in tables 2 and 3 before and after we reach 100% on the training set.

6.1 2D experiments

The best 2D accuracy we obtain without augmentation is 30% using a modified VGG 16. [15]

Interestingly, randomly augmenting the 2D dataset with shifts of up to 10% in the horizontal or vertical direction is enough to increase the accuracy from 30% to 46.3%. Adding rotation (5 or 15 degrees) lowered accuracy and allowing shifts of greater than 10% also decreased accuracy.

6.2 Very Small CNNs and LSTMs

In this set of experiments, we used 3D convolutional LSTM of 3 different configurations that have been widely cited in the literature as being effective for this task.

Despite widely different architectures, these networks all quickly resulted in an overfit network with 100% accuracy on the training set and roughly 10% accuracy on the validation set, often within the first 10 epochs.

When trained on reduced size grayscale data the results were similar but slightly better with the best accuracy on the validation set at around 16%.

The goal of this set of experiments was to identify the network architecture which would be most promising for measuring the impact of temporal information for the driver monitoring task, but due to the overfitting issue, we can not reach any conclusions from these 3 experiments.

6.3 Experiments with small 3DCNN architectures

Here we report the results of our experiment with small 3DCNN, which was designed to reduce overfitting by forcing the network to concentrate on more general features and by limiting the number of low level features.

This network achieved an accuracy of 39.57% on the validation set which is better than the 2D VGG 16 inspired network, which had been shown to be effective at this task in the Distracted Driver Competition without manual augmentation, but not better than the 2D VGG when augmented data was provided.

Variations with additional dropout and Batch Normalization layers were also evaluated, but only increased the overfitting and decreased validation accuracy.

None of the augmentation strategies that were found to improve accuracy for the 2D network (translation, rotation, featurewise normalization) increased accuracy for this small 3DCNN architecture.

Table 1: Top performing approaches (>= 30%)

Approach	Accuracy
Transfer learning on 3D CNN. First 2 layers frozen	73.35%
Transfer learning on 3D CNN. First 5 layers frozen	60%
2D VGG 16 with augmentation	46%
3D CNN without augmentation	39.57%
2D VGG 16 without augmentation	30%

Table 2: The best result on the validation set before reaching 100% on the training set was:

Train loss*	Val loss	Train Accuracy	Val Accuracy
0.0359	0.9004	0.9984	0.7273

Table 3: The best results on the validation set after reaching 100% on the training set was

Train loss*	Val loss	Train Accuracy	Val Accuracy
0.0173	0.8563	1	0.7335

6.4 C3D trained from scratch

When we trained the C3D from scratch on the distracted driver dataset, the results were: 100% accuracy on the training set. 12% accuracy on the validation set.

6.5 Transfer Learning with C3D

The first experiment was to train the pretrained 3CD network at a very low learning rate using the driver monitoring dataset. This still resulted in overfitting and unsatisfactory results on the validation set (13%).

The next and most successful transfer learning approach we used was to train only on the last few layers, freezing the learning rate of the first layers. When the first 5 layers were frozen, we achieved 60% on the validation set, which is a significant improvement over the previous results. Further refinement (freezing just the first 2 layers) allowed us to achieve over 72% accuracy on the validation set.

*in all cases loss is the categorical cross-entropy calculated on an entire batch.

7 DISCUSSION

When training a CNN, Information flows in a cone-shaped manner (see figure 1). This is a property of the convolution and pooling operations, which merge and mix the information of several pixels (a window of pixels) into a single value. In the early layers of the convolutional network, each row of the layer corresponds to a small spatial portion of the input image. This means that in these early layers, the network is restricted to the details and low-level features of the input data.

This is also the case for temporal information where time is the depth dimension. In the early stages of the 3D network, the kernels are able to observe a very short part of the input time sequence data. i.e., the beginning layers of the network are processing a short time sequence from a small spatial portion of the input data.

This could correspond to small finger movement, or eye blinking for example. These movements are common among a wide variety of human activities like swimming or holding a cell phone. This is the main reason a transfer learning approach that involved freezing the first two layers gave significantly better results than other methods (See figure 2). In our presented approach, the network trained on the sports 1m dataset (a database of YouTube videos focused on sports) was tuned for the driver monitoring task, but the first two layers were frozen during the tuning procedure. The success of these features indicates that the early stage features learned from sports activity classification are generic enough to be used for other human activities.

These features represent the most detailed activities, which are the same for a wide variety of human behaviors. Figure 1 shows that by going deeper in network layers, each row of a convolutional layer is observing a wider and larger temporal and spatial region of the input data.

This suggests that the kernels in these later layers are dealing with coarser features of the input data, which includes specific movements or behaviors of the subject in the input movie stream. This explains why tuning later layers for our specific task converged to a reasonable approximation of driver activities. In figure 2, network A is the C3D network trained on the sports 1M dataset and the network B is the presented network for driver activity classification. As you can see, the parameters from the first two layers of the network A were transferred to network B and the rest of the network was tuned for our task.

This is our explanation of the results obtained by a transfer learning approach.

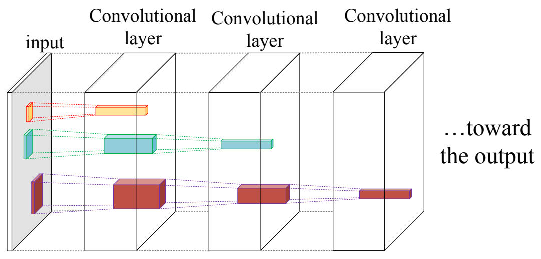

Figure 1: Illustration of cone shape of information flow in convolutional neural networks.

8 CONCLUSIONS

We have shown that the lower level filters learned by a 3D CNN can be used to greatly increase the accuracy on small datasets of drivers for the driver behavior classification task. It is not obvious that the first layers of a network trained for identifying actions in sports videos, such as basketball and swimming, could also be used to distinguish between distracted driver actions like left and right hand cell phone use or speaking with a passenger.

We have also shown that temporal information can be used to increase accuracy for the driver behavior monitoring task over a network that does not use such temporal information.

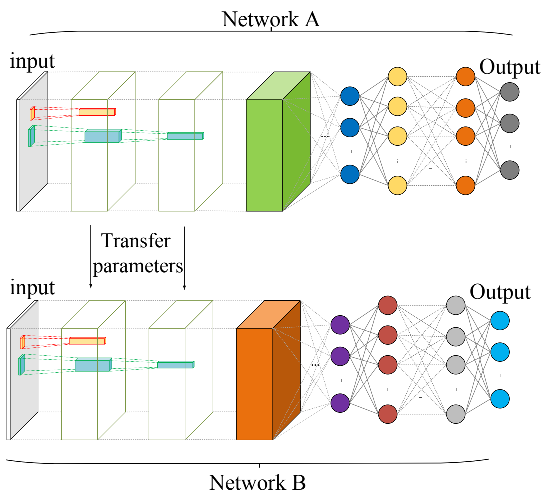

Figure 2: Illustration of transfer learning concept where the first layers in network A and network B are the same.

At a very low level, the action of Moving fingers and heads may not be substantially different between different action recognition problems for convolutional neural networks. In our experiments, using any more than the first two layers decreased accuracy.

Given the shortage of well-labeled task-specific datasets for action recognition, this is an encouraging result.

ACKNOWLEDGMENTS

This research is funded under the SFI Strategic Partnership Program by Science Foundation Ireland (SFI) and FotoNation Ltd. Project ID: 13/SPP/I2868 on Next Generation Imaging for Smartphone and Embedded Platforms. This work is also supported by an Irish Research Council Employment Based Programme Award. Project ID: EBPPG/2016/280.

We gratefully acknowledge the support of NVIDIA Corporation for the donation of a Titan X GPU used for this research.

REFERENCES

[1] François Chollet. 2015. Keras. https://github.com/fchollet/keras. (2015).
[2] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2625–2634.
[3] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. 2016. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 1933–1941.
[4] T Hoang Ngan Le, Yutong Zheng, Chenchen Zhu, Khoa Luu, and Marios Savvides. 2016. Multiple Scale Faster-RCNN Approach to Driver's Cell-Phone Usage and Hands on Steering Wheel Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.* 46–53.
[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[6] Chiori Hori, Shinji Watanabe, Takaaki Hori, Bret A Harsham, JohnR Hershey, Yusuke Koji, Yoichi Fujii, and Yuki Furumoto. 2016. Driver confusion status detection using recurrent neural networks. In *Multimedia and Expo (ICME), 2016 IEEE International Conference on.* IEEE, 1–6.
[7] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition.* 1725–1732.
[8] Alexander Klaser, Marcin Marsza lek, and Cordelia Schmid. 2008. A spatio-temporal descriptor based on 3d-gradients. In *BMVC 2008-19th British Machine Vision Conference.* British Machine Vision Association, 275–1.
[9] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1, 4 (1989), 541–551.
[10] Jun Lei, Guohui Li, Jun Zhang, Qiang Guo, and Dan Tu. 2016. Continuous action segmentation and recognition using hybrid convolutional neural network-hidden Markov model model. *IET Computer Vision* 10, 6 (2016), 537–544.
[11] Joe Lemley, Shabab Bazrafkan, and Peter Corcoran. 2017. Deep Learning for Consumer Devices and Services: Pushing the limits for machine learning, artificial intelligence, and computer vision. *IEEE Consumer Electronics Magazine* 6, 2 (2017), 48–56.
[12] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. 2016. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 4207–4215.
[13] Natalia Neverova, Christian Wolf, Griffin Lacey, Lex Fridman, Deepak Chandra, Brandon Barbello, and Graham Taylor. 2016. Learning human identity from motion patterns. *IEEE Access* 4 (2016), 1810–1820.
[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems.* 91–99.
[15] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[16] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* (2012).
[17] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688 (May 2016). http://arxiv.org/abs/1605.02688
[18] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision.* 4489–4497.
[19] DRGHR Williams and GE Hinton. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533–538.
[20] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems.* 802–810.
[21] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. 2015. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 4694–4702.
[22] Liliang Zhang, Liang Lin, Xiaodan Liang, and Kaiming He. 2016. Is Faster R-CNN Doing Well for Pedestrian Detection?. In *European Conference on Computer Vision.* Springer, 443–457.