Information Modelling and Knowledge Bases XXXII
M. Tropmann-Frick et al. (Eds.)

doi:10.3233/FAIA200822

Recognizing Human-Object Interaction in Multi-Camera Environments

Marina TROPMANN-FRICK [a,1], Thien Phuc TRAN [b]
[a] *University of Applied Sciences Hamburg, Berliner Tor 7, 20099 Hamburg, Germany.*
marina.tropmann-frick@haw-hamburg.de
[b] *thienphuc.tran@haw-hamburg.de*

Abstract. This work introduces Multi-Fusion Network for human-object interaction detection with multiple cameras. We present a concept and implementation of the architecture for a beverage refrigerator with multiple cameras as proof-of-concept. We also introduce an effective approach for minimizing the required amount of training data for the network as well as reducing the risk of overfitting, especially when dealing with a small data set that is commonly recorded by a person or small organization.
The model achieved high test accuracy and comparable results in a real-world scenario at the Event Solutions in Hamburg 2019. Multi-Fusion Network is easy to scale due to shared learnable parameters. It is also lightweight, hence suitable to run on small devices with average computation capability. Furthermore, it can be used for smart home applications, gaming experiences, or mixed reality applications.

Keywords. Deep learning, computer vision, human-object interaction detection

1. Introduction

Nowadays, in the era of Internet-of-Things, computers or computational units can be used on many devices or *things*, also for personal usage. Computers cannot anticipate or recognize human activities without users giving them a hint or somehow telling them about their intent. Human-object interaction detection tries to make computers understand the visual world by learning how humans and objects in the real-world are interacting with each other. There is no common approach to tackle this problem yet, though almost all state-of-the-art methods are deep learning models trained in supervised manner.
There has been a big hype around deep learning in the last decade, especially in the area of computer vision. Many solutions in this area are shifting from statistical approaches to neural networks because of the enormous advantages of convolutional neural networks.
State-of-the-art methods for human-object-interaction detection such as HAKE [3] and InteractNet [4] are using convolutional neural networks with special architecture as well. Besides, HAKE and InteractNet integrate human action embedding and region proposal networks, see Figure 1. Such complex approaches require high computational capability.

[1]Corresponding Author: Marina Tropmann-Frick, University of Applied Sciences Hamburg, Berliner Tor 7, 20099 Hamburg, Germany; E-mail: marina.tropmann-frick@haw-hamburg.de.

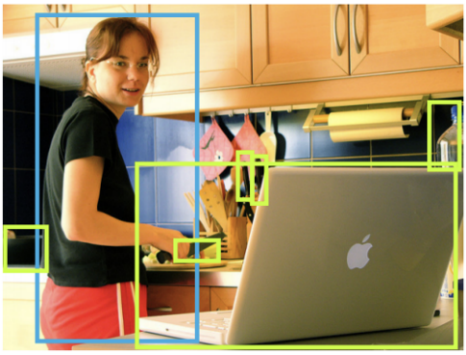


Figure 1. Region proposal network [4]

Therefore, they are mostly inappropriate for real-world use cases. For instance, when there can be only one person interacting with the system at once and only some human body parts are to be seen.
A monitoring system that uses a single camera could struggle to work reliably because of the dead zones, blind spots of the camera setup or occlusions that occur, when an object is hidden behind another object, see Figure 2. For that reason, many systems use multiple cameras to be able to cover the monitoring area fully.
However, an autonomous monitoring system using multiple cameras from different view angles and positions encounters other difficulties. Namely, additional computational burden and the necessity for merging observed information from all cameras into a final one stream. Development of such algorithms is highly difficult and time consuming. Furthermore, physical and mathematical approaches to solve this problem may strongly depend on cameras relative positions and hence require camera calibration or strict camera installation procedures, which may lead to further problems.

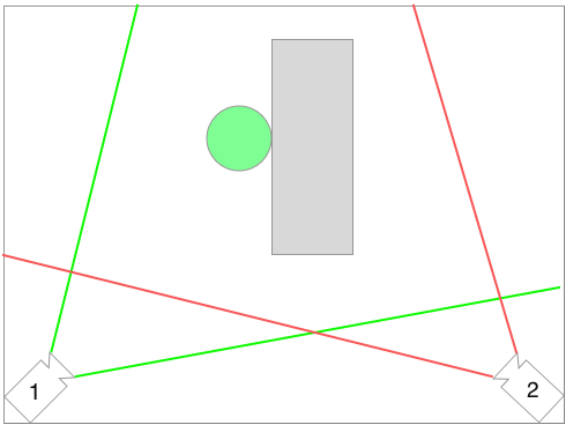


Figure 2. The occlusion problem: Only the camera 1 can see the object

Human perception enables us to understand actions regardless of view angles. Because we have been seeing and learning them since we were born. If a person cannot

tell which action is being performed, he or she would try to move around and to look at the action from different view angles. In consideration of this intuition, deep learning is well-suited for solving this kind of problems without requiring any calibration or having to follow any installation rule.
The main contributions of this work are:

1. We introduce an architecture for human-object interaction detection: Multi-Fusion Network, which is simpler than state-of-the-art methods but is better suited for simpler use cases: when there is only one single subject to be observed. More importantly, the architecture supports both, single and multi-camera environments.
2. We show the implementation process for a beverage refrigerator with multiple cameras as a proof-of-concept. The cameras in the refrigerator should recognize in real-time, what the user is doing with objects inside the fridge and which object is being interacted with.

2. Multi-Fusion network

Multi-Fusion network follows the divide-and-conquer principle. It firstly divides the task of recognizing human-object interaction into two smaller ones: action recognition and object classification. This approach helps to use the data set more effectively and makes the models intuitive and more simple to construct.
Action recognition with multiple cameras is broken down into movement recognition with single-camera subtasks, the recognized movements from all cameras are then aggregated, forming a final action. The same approach is applied for object classification.

2.1. Architecture

The architecture of the Multi-Fusion network consists of two sub-models, see Figure 3. The action classification model is responsible for predictions on each video, the object classification model is responsible for prediction on every single video frame instead.

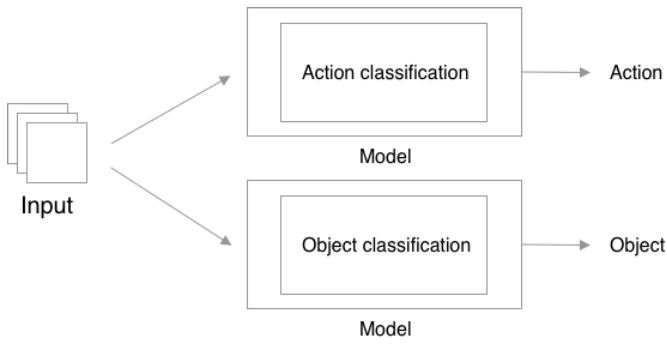


Figure 3. Illustration of using two sub-models

Each pair of action-object should be treated as a single class, however, we lose the ability of generalization by doing so. The separation of action and object not only increases generalization ability, but also helps to use data efficiently, since using action-object pair could make the number of classes explode rapidly.
This approach still encounters another issue when building the data set. Not all video frames of, e.g., ”taking an apple” can be used for object class ”apple”, because there are also moments at which the hand is not holding anything when it is visible or even moves. Some pre-processing techniques can solve this problem, which will be discussed in a later chapter.

2.2. Temporal fusion

In order to detect the performed movements as actions we use the concept of temporal fusion in our approach. The temporal fusion block can be simply a fully connected layer or a more complex block that consumes an image sequence and produces an encoded information. Many scientific papers have proposed various approaches for video understanding such as 3D Convolutional Network, Inflated 3D Convolutional Network [5], CNN-RNN and especially Temporal Relation Network (TRN) [6].
We applied the idea of TRN for temporal fusion block (see Figure 4), which can learn to reason relations between changes of entities along the time axis. Rather than using optical flow to learn movement patterns, it only recognizes state changes in time and anticipates what is happening or reasons what happened.
Spatial information can contribute a lot to many action recognition tasks. However, actions in this concrete use case of the beverage refrigerator can only be distinguished primarily using temporal information, namely, hand movements and changes in the presence of an object. Therefore, this use case can evoke the potential of TRN, helping to achieve good performance with low computational costs. Although TRN can be imple-

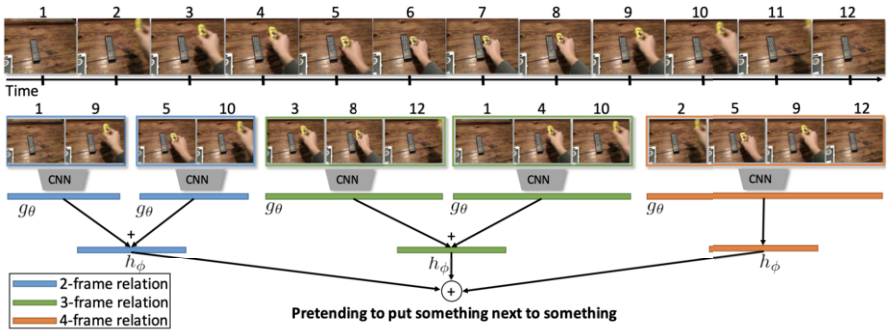


Figure 4. How Temporal Relation Network works [6]

mented with multi-scale time relations, in our approach single scale time relation is used. The reason for that is the fact that multi-scale time relations bring no significant gain for this use case after some experiments.
Instead of random sampling, as in the original paper, frames are sampled so that they are equally distributed across the time axis. This type of sampling ensures that the representative positions of movement are completely captured, because every hand position is

crucial for understanding actions. In this case, a skipped snippet can also lead to a change in the meaning of the action.
After extracting information from sampled frames, the model needs to perform temporal fusion to connect information from many positions in time. In this work, the acquired information is fed into a GRU (Gated Recurrent Unit), which works as an encoder. The final hidden state of the GRU is then passed for further processing. This approach is introduced in the paper "Temporal Reasoning in Videos using Convolutional Gated Recurrent Units" [7].

2.3. View fusion

Due to the limitations of camera perspectives, a camera may not always be able to see an object, leading to wrong classifications. Multiple cameras can co-operate with each other to produce the best result. The main idea of multi-view fusion is to find a consensus among all the cameras, see Figure 5. The ideal mapping of temporal fusion block (before

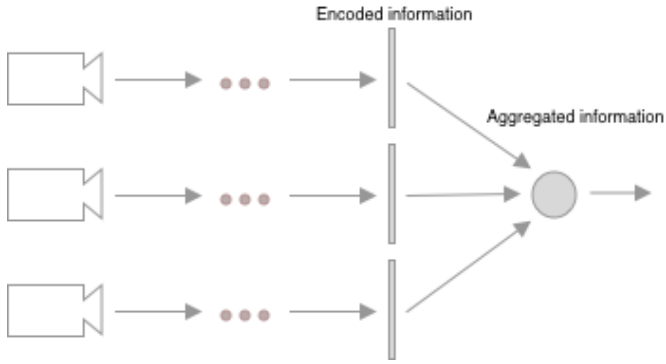


Figure 5. The illustration of the view fusion

the view fusion) is signal strength of informative movement patterns that each individual camera can observe despite their different view angles so that combining that information from all cameras can help to make the best decision. Therefore, instead of using another fully connected layer to merge all into one, the mean values of these signals from all cameras are calculated.

2.4. Architecture overview

The pre-processing blocks contain pre-processing operations such as inactivity removal, background subtraction, and density-based cropping, which clean the input and prepare it for further processing, as we will show in a later chapter. Video frames from the pre-processed input are then sampled and fed into the feature extractor. The extracted visual features are fused together by an encoder in the temporal fusion step. The encoded information from different cameras is aggregated into one, which is then finally passed to the classifier. The overview over the architecture is shown in the Figure 6.
Simultaneously, with action recognition, all pre-processed frames are also passed to the object classifier. The object classifier performs prediction on every frame that it receives from the pre-processing block. All object predictions, therefore, also need to reach a con-

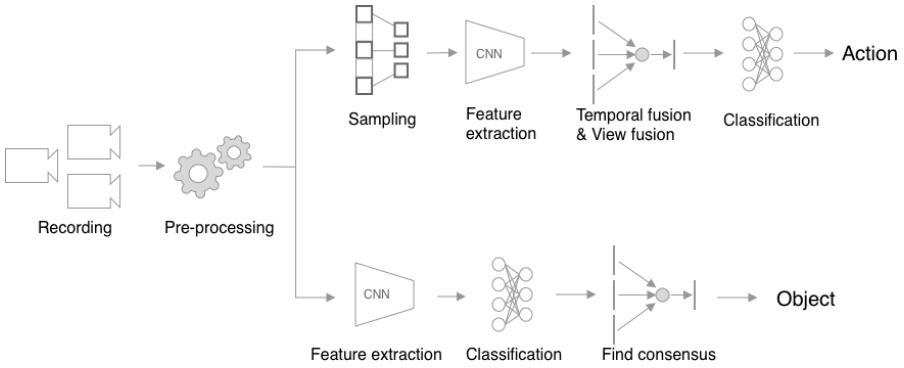


Figure 6. Architecture overview

sensus. Different policies can be applied to this action. The most frequently classified object (with a high confidence score) is chosen for this work.

2.5. Shared weights

Despite multiple input sources, a single feature extractor with shared weights is used for extracting visual features in each frame, so that it can learn to see an object from different points of view and also keep the number of parameters as low as possible.
The extracted features are then passed to corresponding temporal fusion blocks to recognize movement patterns. The GRU cell in temporal fusion block is shared since it should also learn to see actions from different angles. The hidden state of the GRU cell is reset after completing a computation task for a single camera.
All learnable parameters in feature extraction and temporal fusion stage are shared; only a mean calculation is performed in view fusion stage. Furthermore, the Multi-Fusion network architecture is scalable because increasing the number of cameras does not require an increase in the amount of parameters.

3. Data set

The data set for the beverage refrigerator contains seven object classes and three actions, captured by MQ013CG-E2 cameras (1.3 Megapixel) with wide-angle lenses at 20 FPS, see Figure 7. Some of the chosen objects possess different shapes and colors, which helps the model to distinguish between them easily. However, some pairs have the same shape or color in order to find out how well the model performs. An object can be taken or put, or nothing can also be done with the object (in each video sequence), see Table 1.
Four cameras are used to capture actions from different angles, preventing objects from being fully covered by hands or other objects. Two cameras are mounted at the front top left and right corners, pointing inwards. If an object is being taken by the left hand, the left camera may not be able to see that object, but the other front camera will. The third camera is mounted at the rear top left corner, pointing outwards, and the fourth is mounted on top of the refrigerator in order to provide more information.
No camera is mounted at the top center position (pointing downwards), because it would only be able to see bottle caps, and the model may have no chance to classify when

Figure 7. The camera and lens used to record the dataset

Objects	Actions
Apple	
Banana	
Lemonaid+ Blutorange (Bottle)	Nothing
Lemonaid+ Limette (Bottle)	Take
Club-Mate (Bottle)	Put
Granini Orange (Bottle)	
Veltins beer (Can)	

Table 1. Objects and actions in the dataset

working with low-resolution images correctly.
In the beginning, each camera recorded 1818 video frames. However, the model trained

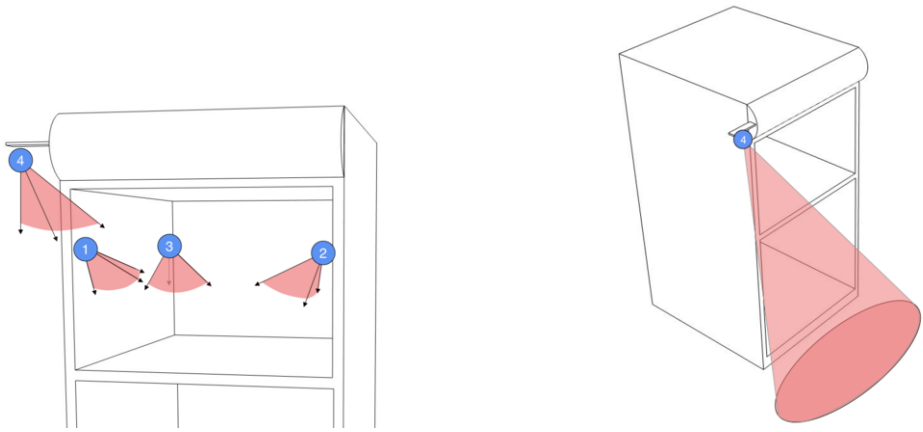


Figure 8. Camera positions

with those videos was strongly overfitted regardless of how hyperparameters were configured. The model performed very well during training; accuracy increased up to 99%, but made random guess during the validation phase. After some investigations, it turned out to be caused by the lack of diversity in the data set.
Only four persons participated in the recording phase. Although there was an effort of changing clothes as well as shuffling object positions frequently, it was still insufficient

for the model to generalize. Besides, the background in the third and fourth cameras rarely changes. The evaluation was performed on a model with base architecture, i.e.,

Figure 9. Images captured by camera 1 and 2

Figure 10. Images captured by camera 3 and 4

without additional pre-processing blocks. Due to the overfitting problem, the third and fourth cameras were removed. The model did show significant improvement by making reasonable classifications. Accuracy increased slowly over time and stopped at about 50% and about 30% for action classifier and object classifier, respectively.

4. Preprocessing techniques

The data sets were recorded several times, each time with minor changes in the camera's field of view. At first, various models were trained with different numbers of segments. The fact that the data sets are small and were recorded by four persons on the same day at the same location led to overfitting of all models, despite many regularization techniques such as reducing the number of hidden units, dropout, and adding regularization terms.
The root cause of this overfitting problem is highly likely to be the lack of variance among samples in the data set. Background, surrounding objects, and clothing were infrequently changed. Since it is very labor-intensive to record such a data set that has many different backgrounds and users, some techniques should be applied in order not only to overcome overfitting by ignoring irrelevant information in video frames, but also to keep the amount of training data needed as small as possible.

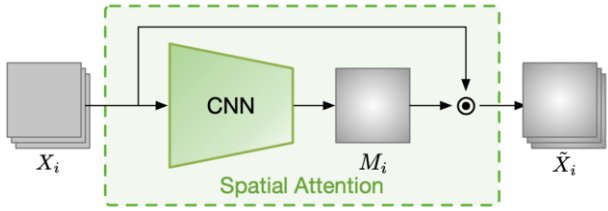


Figure 11. Spatial attention block [1]

4.1. Finding the region of interest

The fundamental idea of attention in video action recognition is that humans only need to look at parts of a video at each instance of time in order to understand which action is being taken. The ability to know where and when to pay attention to grasp the happening has been acquired since humans were born. Therefore, attention is nowadays widely applied in many machine learning fields in general and in action recognition tasks in particular. This work only makes use of spatial attention (where to look), see Figure 11; temporal attention is neglected to let the action classifier learn to understand the video by reasoning action sequences.

Spatial attention

A soft attention mechanism [1] was inserted before the feature extraction model and is responsible for blacking-out irrelevant parts of input to minimize their impacts on prediction. The spatial attention block consists of a convolutional network with the 'same' padding in all layers that learns to produce an importance mask for each input image, which is then multiplied element-wise with the original input image. Same padding means the width and height of output will be the same as those of input image. This block is designed to be plugged-in in any existing network easily. The input image will be fed to the spatial attention block, whose output will be then passed to the classifier.
However, the spatial attention block failed to produce reasonable importance masks in practice, and the training loss did not converge. Insufficient variation in images could be again the main reason for this failure.

Background subtraction as an attention mechanism

Background subtraction is a technique used for segmentation of the foreground objects from the background. In other words, background subtraction detects moving objects and is mostly used for traffic monitoring tasks such as detecting and tracking vehicles, pedestrians, etc.
Traditional methods, e.g., Frame Differencing, produce good results when the camera is stationary. Otherwise, every pixel in the image would change, and the background estimation algorithm fails. Many deep learning approaches developed in the last few years are, however, computationally too expensive for being used in a real-time application due to their convolutional encoder-decoder architecture.
Fortunately, in our approach, all of the cameras are stationarily mounted in the refrigerator in this specified use case. Thus, traditional algorithms could be applied to generate

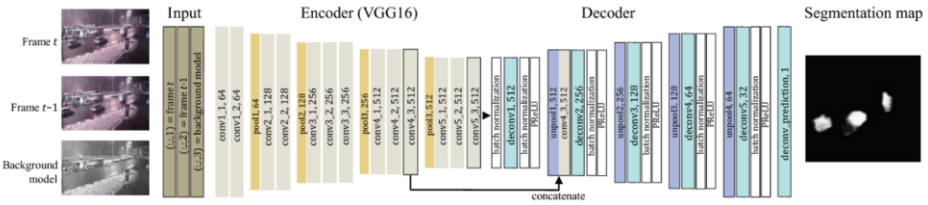


Figure 12. Encoder-decoder architecture for foreground detection [8]

foreground masks, which effectively works as a spatial attention mechanism.
Andrews Sobral implemented various algorithms and introduced BGSLibrary for foreground detection and background estimation in his work [9]. Although the library was written in C++, BGSLibrary can also be used in Python, Java, and MATLAB with specific wrappers.
BGSLibrary supports different OpenCV versions, but the number of available algorithms does differ at the time of this experiment. BGSLibrary compiled with OpenCV 3, OpenCV 4 has 41 and 15 algorithms, respectively. Therefore, BGSLibrary used for this work was compiled with OpenCV 3.4.1 on Ubuntu 16.04 LTS.
41 available algorithms were benchmarked to find which is the best-suited candidate for this problem. Benchmarking criteria are computational complexity, noise level in the mask, intersection over union (Jaccard-index) of the detected moving object and ground truth.
According to the criteria mentioned above, the algorithms fall into three main groups (see Figure 13):

1. Fast, low mask quality
2. Slow, high mask quality
3. Average speed, average mask quality

Frame Differencing, SuBSENSE [11], and Local Binary Pattern with Markov Random Field (LBP-MRF) [10] are the representative algorithms for group 1, 2, and 3, respectively.
There is also a trade-off between speed and quality. Slow algorithms are not suitable for real-time requirements. Fast algorithms produce low-quality masks with a high noise level, which makes the relevant features in images barely recognizable. Eventually, LBP-MRF is selected due to its sufficient speed to fulfill the real-time requirements, and due to its more than acceptable mask quality. More precisely, the density of positive pixels in the area of the moving objects is here much higher than that in the background area. The density difference between those two areas plays an important role in enhancing input quality, which will be further discussed in the next section.

4.2. Zooming to the region of interest

Input video frames usually contain black regions after multiplication with their masks. The average percentage of area that contains a moving object in the recorded data set is only about 10%, i.e., 90% of data the neural network processes contains no information. Furthermore, most convolutional neural networks for extracting features are built in such a way that small, simple patterns in small areas are detected by the first filters;

(a) Input image (b) Low quality (c) Average quality (d) High quality

Figure 13. Different mask qualities with a wide range of noise level and coverness

dimension of the input is step-wise reduced, complex patterns formed by simple ones are then detected by the last filters. Consequently, complex patterns in small areas are hardly detectable. Zooming to the area of the moving object helps to reduce wasted computation, to make patterns in images easier to detect by enlarging them, though it is also challenging to do so without affecting the real-time capability.

Spatial Transformer

Spatial Transformer [2] was introduced by Google DeepMind and became popular by its ability of learning rotation, translation, scale, etc., to help simplify classification tasks. The principle of STN is to use convolutional layers (or fully-connected layers) followed by a last fully-connected layer with 6 perceptrons to look at input image and produce an appropriate affine transformation matrix $\theta = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix}$. The weights of the last layer are zero-initialized and its biases are initialized so that its output is an identity transformation matrix $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$. The affine transformation is then applied to a sampling grid G. Because, we only need to zoom into the region of interest, only translation and

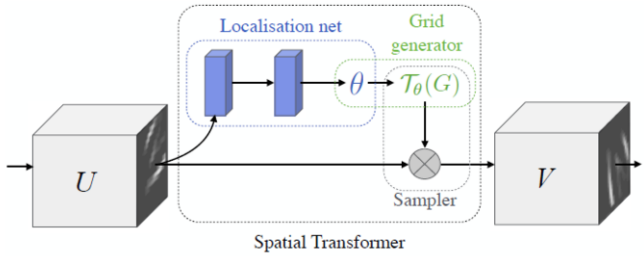
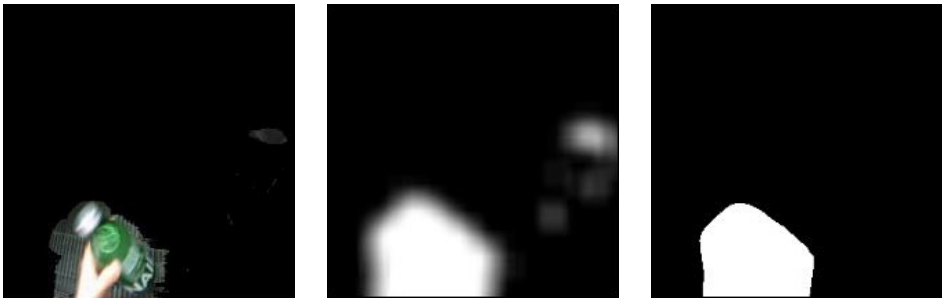


Figure 14. Spatial Transformer Network [2]

scale are needed for our approach. The number of parameters for affine transformation can be reduced from 6 to 4, meaning $\theta = \begin{bmatrix} \theta_{sx} & 0 & \theta_{tx} \\ 0 & \theta_{sy} & \theta_{ty} \end{bmatrix}$.
Spatial Transformer works as a plugin to many classification tasks on such data sets as MNIST, Street View House Numbers, German Traffic Signs, etc. However, the zooming effect of STN in this specific data set is not remarkable and reliable. Although it did zoom



(a) Input image (b) After mean filtering (c) After thresholding

Figure 15. Illustration of density-based cropping

into moving objects successfully, sometimes input images were squeezed or translated until they completely disappeared.

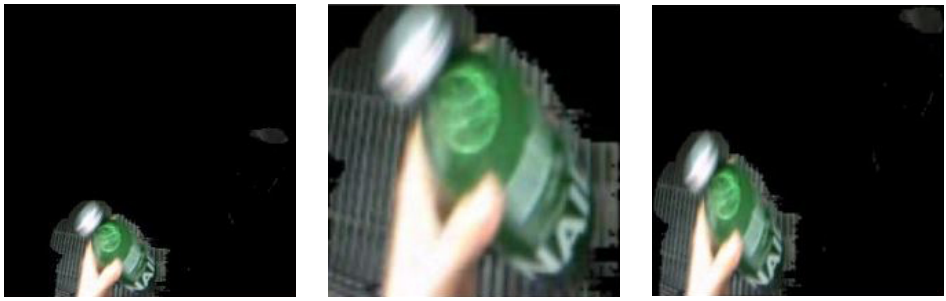
Density-based cropping

To utilize the fact that there is nothing else visible than the moving object in every frame, density-based cropping is introduced based on the following assumption. Density of positive pixels in an area that contains the moving object is significantly higher than that in an area solely containing background.
Firstly, a mean filter is utilized to reduce noise by computing the percentage of positive pixels in fixed-size areas, resulting in a blurry grayscale version of the input image. The area containing the moving object will still be white due to its high pixel density, while noise areas will fade to gray. Image thresholding is then applied to remove noise pixels completely, see Figure 15.
The cropping procedure is simple; all it needs to do is to slice the image, so that every row/column contains at least one white pixel. Hence, choosing the right threshold value plays a crucial role in producing reasonable results.
The image size for this use case is 96x96, the averaging kernel size is 5x5, and the threshold value is set to 0.7. Density-based cropping has proved its ability in practice by producing desired results in most cases. Sometimes, input image is not well cropped because noises were not well eliminated; even one tiny noise in a corner that passed the thresholding stage could affect the results seriously, see Figure 16.

Unlike spatial transformer, density-based cropping may also fail to output good results as expected due to incorrect parameters (kernel size, threshold value). Moving objects are still visible in failure cases.

5. Evaluation of Multi-Fusion Network

The models are trained on a single GTX 1080Ti for maximal 50 iterations (epochs) with SGD optimizer. The batch size for the action model and object model is 16 and 64, respectively. Both action and object classifiers have a pre-trained ImageNet ResNet-18 as the base feature extractor. All layers of the base model ResNet except the last convolutional block are frozen, allowing them to learn blurry features that appear due to hand



(a) Input image (b) Good result (c) Bad result

Figure 16. Success and failure of density-based cropping

and object motion. If this last block was frozen, the model could not achieve high accuracy.

Bottleneck features	Temporal units	Accuracy (%)
256	64	88.048
128	64	90.438
128	**32**	**91.235**
128	16	86.065
64	16	89.243

Table 2. The result table of the action model

The number of features in this work is much smaller than that of the ImageNet data set. Shapes and colors of hands, bottles, cans, etc. are the most important features for this specific use case. Logically, a 512-dimensional output of ResNet is superabundant. A dropout layer with a rate of 0.7 is applied, followed by a bottleneck layer for honing common visual features into specific ones for this approach. 16 frames are sparsely sampled from every video in the same manner. The action classification model achieved 91.235 % accuracy with 128 bottleneck features and 32 temporal units. The object classification model is trained for 41 epochs and achieved 98.954% accuracy.
A practical test is performed to ensure the accuracy of the trained models. The accuracy of both models in practice is just slightly lower than that in the validation step. The actions (put, take, nothing) are correctly classified in most cases, except some special cases where the object couldn't be seen or fully detected in the background subtraction stage. The action classifier performed well in the test, but it still can make false decisions, especially when unseen artifacts appear (e.g., different sleeve colors) or when other objects are also seen in the foreground (bad background subtraction).
The refrigerator is equipped with an external display for showing recognition results, see Figure 17. It was placed at the Event Solutions in Hamburg 2019 and Sommerfest 2019 for real-worl use case scenarios. The models performed well even at the event, where many other light sources could have had a negative impact on the results.
The experiments serve as proof-of-concept, and they have confirmed the feasibility and efficiency of the proposed architecture. The model performed its tasks correctly in practice, although it was trained on a tiny data set. However, more quantitative and qualita-

Figure 17. A photo of the beverage refrigerator

tive data, numerous classes recorded in various domains, are needed in order to fully and fairly evaluate the Multi-Fusion network architecture's effectiveness.

6. Conclusion

Supervised learning tasks are data-consuming giants, and it is extremely labor-intensive to make a large data set with a diversity of features. Nevertheless, it may be possible to minimize the amount of required data by analyzing project requirements, data set, then applying some techniques to reduce dimensionality. An important result of this work shows that decisions strongly depend on the presence/absence of an object currently in motion. Attention mechanism bring remarkable contribution to reducing the amount of data if it can focus on the moving object. Using background subtraction as a strong attention mechanism can also help to reduce overfitting. However, which background subtraction algorithm should be used depends on specific characteristics of the use case. Bad choice of background subtraction can lead to loss of relevant information and, therefore, to performance degradation.

Data recorded by a small organization could contain not enough variations for the model to be able to recognize and extract relevant features, which leads to the problem of overfitting. The presented Multi-Fusion network reduces the amount of training data and, therefore, also reduces the required effort and the risk of overfitting.

A lot of unexpected cases in real-world experiments are inevitable. Input standardization and noise removal are much more crucial in practice. This work utilizes density-based cropping and inactivity trimming in the pre-processing step for achieving the best results. Multi-Fusion network has proved its potential by achieving 91.235% accuracy on the test

data set with only hundreds of training samples.
Potential of Multi-Fusion network could be also indoor activity recognition for gaming experiences, smart home applications, or augmented / mixed reality devices. This architecture could be used in embedded systems with average computational capability or small toolkits such as Google Coral or NVIDIA Jetson due to its lightweight and scalable architecture.

References

[1] Meng, Lili, et al. "Interpretable spatio-temporal attention for video action recognition." Proceedings of the IEEE International Conference on Computer Vision Workshops. 2019.
[2] Jaderberg, Max, Karen Simonyan, and Andrew Zisserman. "Spatial transformer networks." Advances in neural information processing systems. 2015.
[3] Li, Yong-Lu, et al. "HAKE: Human Activity Knowledge Engine." arXiv preprint arXiv:1904.06539 (2019).
[4] Gkioxari, Georgia, et al. "Detecting and recognizing human-object interactions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.
[5] Carreira, Joao, and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset." proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.
[6] Zhou, Bolei, et al. "Temporal relational reasoning in videos." Proceedings of the European Conference on Computer Vision (ECCV). 2018.
[7] Dwibedi, Debidatta, Pierre Sermanet, and Jonathan Tompson. "Temporal reasoning in videos using convolutional gated recurrent units." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2018.
[8] Lim, Kyungsun, Won-Dong Jang, and Chang-Su Kim. "Background subtraction using encoder-decoder structured convolutional neural network." 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE, 2017.
[9] Sobral, Andrews. "BGSLibrary: An opencv c++ background subtraction library." IX Workshop de Visao Computacional. Vol. 27. 2013.
[10] Kertsz, Csaba, and Vincit Oy. "Texture-based foreground detection." (2011).
[11] St-Charles, Pierre-Luc, Guillaume-Alexandre Bilodeau, and Robert Bergevin. "Subsense: A universal change detection method with local adaptive sensitivity." IEEE Transactions on Image Processing 24.1 (2014): 359-373.